Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

H.264

(03/2005)

SÉRIE H: SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

Infrastructure des services audiovisuels – Codage des images vidéo animées

Codage vidéo évolué pour les services audiovisuels génériques

Recommandation UIT-T H.264

UIT-T

Union internationale des télécommunications

RECOMMANDATIONS UIT-T DE LA SÉRIE H

SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

CARACTÉRISTIQUES DES SYSTÈMES VISIOPHONIQUES	H.100–H.199
INFRASTRUCTURE DES SERVICES AUDIOVISUELS	
Généralités	H.200–H.219
Multiplexage et synchronisation en transmission	H.220–H.229
Aspects système	H.230–H.239
Procédures de communication	H.240–H.259
Codage des images vidéo animées	**H.260–H.279**
Aspects liés aux systèmes	H.280–H.299
Systèmes et équipements terminaux pour les services audiovisuels	H.300–H.349
Architecture des services d'annuaire pour les services audiovisuels et multimédias	H.350–H.359
Architecture de la qualité de service pour les services audiovisuels et multimédias	H.360–H.369
Services complémentaires en multimédia	H.450–H.499
PROCÉDURES DE MOBILITÉ ET DE COLLABORATION	
Aperçu général de la mobilité et de la collaboration, définitions, protocoles et procédures	H.500–H.509
Mobilité pour les systèmes et services multimédias de la série H	H.510–H.519
Applications et services de collaboration multimédia mobile	H.520–H.529
Sécurité pour les systèmes et services multimédias mobiles	H.530–H.539
Sécurité pour les applications et services de collaboration multimédia mobile	H.540–H.549
Procédures d'interfonctionnement de la mobilité	H.550–H.559
Procédures d'interfonctionnement de collaboration multimédia mobile	H.560–H.569
SERVICES À LARGE BANDE ET MULTIMÉDIAS TRI-SERVICES	
Services multimédias à large bande sur VDSL	H.610–H.619

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T H.264

Codage vidéo évolué pour les services audiovisuels génériques

Résumé

La présente Recommandation | Norme internationale représente une évolution des normes de codage vidéo existantes (H.261, H.262 et H.263). Elle a été développée afin de répondre à un besoin croissant de plus forte compression des images animées pour diverses applications telles que la visioconférence, les moyens de stockage numérique, la diffusion télévisuelle, le trafic Internet et la communication. Elle est aussi conçue afin de permettre une utilisation souple de la représentation vidéo codée dans une large gamme d'environnements de réseaux. L'utilisation de la présente Recommandation | Norme internationale permet de manipuler la vidéo animée comme des données informatiques et de la mémoriser sur des moyens de stockage variés, de la transmettre et de la recevoir sur les réseaux existants et futurs ainsi que de la distribuer sur les canaux de diffusion actuels et de l'avenir.

La révision approuvée en mars 2005 contient des modifications de la norme relative au codage vidéo afin d'y ajouter quatre nouveaux profils, désignés par les appellations "profil élevé", "profil élevé 10", "profil élevé 4:2:2" et "profil élevé 4:4:4". Ces profils sont ajoutés afin d'améliorer la capacité de qualité vidéo et d'augmenter l'étendue des applications visées par la norme (par exemple en incluant la prise en charge d'une plus grande gamme de précisions d'échantillon d'image et de formats chromatiques à résolution élevée). En outre, une définition de nouveaux types de données supplémentaires a été spécifiée afin d'élargir l'applicabilité de la norme de codage vidéo. Enfin, un certain nombre de corrections d'erreurs ont été incluses dans le texte publié. Cette révision, s'ajoutant à l'amélioration de la capacité de codage vidéo, contribuera à conserver l'alignement technique avec l'ISO/CEI 14496-10 correspondante, élaborée conjointement.

Le Corrigendum 1 à la Rec. UIT-T H.264 redressait et mettait à jour divers aspects mineurs afin de mettre la version UIT-T du texte en harmonie avec le statut de l'édition d'avril 2005, approuvée comme nouvelle édition du texte correspondant de l'ISO/CEI 14496-10, élaboré conjointement et techniquement aligné. Ce corrigendum supprime un certain nombre d'erreurs mineures et de besoins de clarification, tout en définissant trois indicateurs de format d'échantillon, qui étaient jusqu'ici réservés.

La présente édition comprend le texte approuvé en mars 2003 ainsi que son Corrigendum 1, approuvé en septembre 2005.

Source

La Recommandation UIT-T H.264 a été approuvée le 1 mars 2005 par la Commission d'études 16 (2005-2008) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8. Il contient les modifications apportées par la Rec. UIT-T H.264 (2005) Corrigendum 1 approuvé le 13 septembre 2005 par la Commission d'études 16 (2005-2008) de l'UIT selon la procédure définie dans la Recommandation UIT-T A.8.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT avait été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Introduction

L'Union internationale des télécommunications (UIT) est l'agence spécialisée des Nations Unies dans le domaine des télécommunications. Le Secteur de la normalisation des télécommunications de l'UIT (UIT-T) est un organe permanent de l'UIT. L'UIT-T est responsable de l'étude des questions techniques, de fonctionnement et de tarification et de la production de Recommandations sur ces questions afin de normaliser les télécommunications au plan mondial. L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, arrête les sujets d'étude des Commissions d'études de l'UIT-T, qui à leur tour produisent des Recommandations sur ces sujets. L'approbation des Recommandations de l'UIT-T est traitée selon les procédures établies dans la Résolution 1 de l'AMNT. Dans certains domaines des technologies de l'information qui tombent dans le champ de compétence de l'UIT-T, les normes nécessaires sont préparées en collaboration avec l'ISO et la CEI.

L'ISO (Organisation internationale de normalisation) et la CEI (Commission électrotechnique internationale) constituent le système spécialisé de la normalisation mondiale. Les entités nationales qui sont membres de l'ISO et de la CEI participent au développement des normes internationales dans les comités techniques établis par les organisations qui traitent respectivement des domaines particuliers d'activité technique. Les comités techniques et d'études de l'ISO et de la CEI collaborent dans les domaines d'intérêt mutuel. D'autres organisations internationales, gouvernementales et non gouvernementales, participent également aux travaux, en liaison avec l'ISO et la CEI. Dans le domaine des technologies de l'information, l'ISO et la CEI ont constitué un Comité technique mixte, l'ISO/CEI JTC1. Les projets de norme internationale adoptés par ce Comité technique mixte sont soumis à l'approbation des organismes nationaux. La publication comme Norme internationale exige l'approbation par au moins 75% des Organismes nationaux ayant exprimé leur voix.

La présente Recommandation | Norme internationale a été préparée conjointement par le Groupe de travail Q.6 de la Commission d'études 16 de l'UIT-T, connu aussi sous le nom de Groupe d'experts de codage vidéo (VCEG, *video coding experts group*), et par le Groupe de travail ISO/CEI JTC1/SC29/WG11, connu aussi sous le nom de Groupe d'experts pour les images animées (MPEG, *moving picture experts group*). Le VCEG a été constitué en 1997 afin d'assurer la maintenance des anciennes normes de codage vidéo de l'UIT-T et de développer une ou plusieurs nouvelles normes de codage vidéo convenables pour une large gamme de services conversationnels ou non conversationnels. Le groupe MPEG a été constitué en 1988 afin d'établir des normes de codage des images animées et des sons qui leur sont associés pour diverses applications telles que les moyens de stockage numériques, la distribution et la communication.

Dans la présente Recommandation | Norme internationale, les Annexes A à E contiennent des exigences normatives qui font partie intégrante de la présente Recommandation | Norme internationale.

Recommandation UIT-T H.264

Codage vidéo évolué pour les services audiovisuels génériques

0 Introduction

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

0.1 Prologue

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Alors qu'ont diminué les coûts aussi bien de la puissance de traitement que de la mémoire, que se sont diversifiés les réseaux de traitement des données vidéo codées, et qu'ont progressé les avancées de la technique du codage vidéo, est apparu le besoin d'une norme industrielle pour la représentation vidéo comprimée avec une efficacité de codage substantiellement augmentée et une robustesse améliorée aux environnements de réseau. A cette fin, le Groupe d'experts de codage vidéo de l'UIT-T (VCEG) et le Groupe d'experts pour les images animées de l'ISO/CEI (MPEG) ont formé une équipe vidéo mixte (JVT, *joint video team*) en 2001 pour le développement d'une nouvelle Recommandation | Norme internationale.

0.2 Objet

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale a été élaborée afin de répondre au besoin croissant d'une plus forte compression des images animées pour diverses applications comme la visioconférence, les moyens de stockage numériques, la radiodiffusion télévisuelle, le trafic Internet et la communication. Elle est aussi conçue afin de permettre l'utilisation de la représentation vidéo codée de façon souple dans une large variété d'environnements de réseaux. L'utilisation de la présente Recommandation | Norme internationale permet de manipuler la vidéo animée comme une forme de données informatiques et de la mémoriser sur des moyens de stockage variés, de la transmettre et de la recevoir sur les réseaux existants et futurs ainsi que de la distribuer sur les canaux de radiodiffusion existants et futurs.

0.3 Applications

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale est destinée à couvrir une large gamme d'applications contenant de la vidéo, y compris les suivantes, mais sans exclusive:

CATV	Télévision par câble sur réseaux optiques, cuivre, etc. (*cable television*)
DBS	Services vidéo de diffusion directe par satellite (*direct broadcast satellite*)
DSL	Services vidéo sur ligne d'abonné numérique (*digital subscriber line*)
DTTB	Radiodiffusion télévisuelle numérique par voie hertzienne de Terre (*digital terrestrial television broadcasting*)
ISM	Support d'enregistrement interactif (disque optique, etc.) (*interactive storage media*)
MMM	Messagerie multimédia
MSPN	Services multimédias sur réseaux par paquets (*multimedia services over packet networks*)
RTC	Services conversationnels en temps réel (visioconférence, visiophone, etc.) (*real-time conversational services*)
RVS	Télévidéosurveillance (*remote video surveillance*)
SSM	Support d'enregistrement en série (magnétoscope numérique, etc.) (*serial storage media*)

0.4 Publication et versions de la présente spécification

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Celle-ci a été élaborée conjointement par le Groupe d'experts de codage vidéo de l'UIT-T (VCEG) et par le Groupe d'experts pour les images animées de l'ISO/CEI (MPEG). Elle est publiée dans les deux organisations – l'UIT-T et l'ISO/CEI – en tant que texte jumelé techniquement aligné.

La version 1 de la Rec. UIT-T H.264 | ISO/CEI 14496-10 se rapporte à la première version approuvée (en 2003) de la présente Recommandation | Norme internationale.

La version 2 de la Rec. UIT-T H.264 | ISO/CEI 14496-10 se rapporte au texte intégré contenant les corrections spécifiées dans le premier corrigendum technique.

La version 3 de la Rec. UIT-T H.264 | ISO/CEI 14496-10 se rapporte au texte intégré contenant à la fois le premier corrigendum technique (2004) et le premier amendement, qui est désigné par l'expression "Extensions de l'étendue de fidélité".

La version 4 de la Rec. UIT-T H.264 | ISO/CEI 14496-10 (la présente spécification) se rapporte au texte intégré contenant le premier corrigendum technique (2004), le premier amendement (les "Extensions de l'étendue de fidélité") et un corrigendum technique additionnel (2005). Au sein de l'UIT-T, la version qui a été publiée après la version 2 a été la version 4 (parce que les travaux de rédaction de la version 4 avaient été effectués avant que l'occasion ait été donnée d'approuver le texte final de la version 3).

0.5 Profils et niveaux

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale est conçue pour être générique en ce sens qu'elle sert à une large gamme d'applications, de débits binaires, de résolutions, de qualités et de services. Les applications devraient couvrir, entre autres, les supports d'enregistrement numériques, la radiodiffusion télévisuelle et les communications en temps réel. En créant la présente Spécification, des exigences variées d'applications typiques ont été prises en compte, les éléments algorithmiques nécessaires ont été développés et ont été intégrés en une syntaxe unique. Ainsi, la présente Spécification facilitera les échanges de données vidéo entre les différentes applications.

Cependant, en considérant la mise en œuvre pratique de la totalité de la syntaxe de la présente Spécification, un nombre limité de sous-ensembles de la syntaxe sont également stipulés au moyen de "profils" et de "niveaux". Ceux-ci et d'autres termes s'y rapportant sont formellement définis au § 3.

Un "profil" est un sous-ensemble de la syntaxe entière du flux binaire qui est spécifiée dans la présente Recommandation | Norme internationale. Dans les limites imposées par la syntaxe d'un profil donné, il est toujours possible d'exiger une très grande variation des performances des codeurs et décodeurs selon les valeurs prises par les éléments de la syntaxe dans le flux binaire, tels que la taille spécifique des images décodées. Dans de nombreuses applications, il n'est habituellement ni pratique ni économique de mettre en œuvre un décodeur capable de traiter toutes les utilisations possibles de la syntaxe au sein d'un profil particulier.

Afin de régler ce problème, des "niveaux" sont spécifiés au sein de chaque profil. Un niveau est un ensemble spécifié de contraintes imposées aux valeurs des éléments syntaxiques dans le flux binaire. Ces contraintes peuvent être de simples limites de valeurs. Elles peuvent aussi prendre la forme de contraintes sur les combinaisons arithmétiques de valeurs (par exemple, une largeur d'image multipliée par une hauteur d'image multipliée par le nombre d'images décodées par seconde).

Les contenus vidéo codés conformément à la présente Recommandation | Norme internationale utilisent une syntaxe commune. Afin de réaliser un sous-ensemble de la syntaxe complète, des fanions, des paramètres et d'autres éléments syntaxiques sont inclus dans le flux binaire afin de signaler la présence ou l'absence d'éléments syntaxiques qui apparaîtront ultérieurement dans le flux binaire.

0.6 Aperçu général sur les caractéristiques de base

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La représentation codée spécifiée dans la syntaxe est conçue afin de permettre une forte capacité de compression pour une qualité d'image désirée. A l'exception du mode de fonctionnement à contournement de transformation pour le codage sans perte du profil élevé 4:4:4 et du mode de fonctionnement I_PCM dans tous les profils, l'algorithme n'est en principe pas sans perte, car les valeurs exactes d'échantillonnage de source ne sont pas, en principe, conservées pendant les processus de codage et de décodage. On peut utiliser un certain nombre de techniques afin de réaliser une compression de haute efficacité. Les algorithmes de codage (qui ne sont pas spécifiés dans la présente Recommandation | Norme internationale) peuvent choisir entre l'intercodage et l'intracodage pour les parties structurelles de chaque image. L'intercodage utilise des vecteurs cinétiques pour la prédiction interne fondée sur les blocs afin d'exploiter les dépendances temporelles statistiques entre différentes images. L'intracodage utilise différents modes de prédiction spatiale afin d'exploiter les dépendances statistiques spatiales dans le signal source pour une image

unique. Les vecteurs cinétiques et les modes d'intraprédiction peuvent être spécifiés pour des tailles de bloc variées dans l'image. La prédiction résiduelle est alors encore comprimée au moyen d'une transformée afin d'éliminer la corrélation spatiale à l'intérieur du bloc transformé avant qu'il soit quantifié, ce qui produit un processus irréversible qui élimine en principe les informations visuelles les moins importantes lorsqu'est formée une approximation fidèle des échantillons de la source. Finalement, les vecteurs cinétiques ou les modes d'intraprédiction sont combinés avec les informations de coefficient de transformée quantifiées et codées au moyen de codes à longueur variable ou d'un codage arithmétique.

0.6.1 Codage prédictif

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

A cause d'exigences contradictoires d'accès aléatoire et de compression à haute performance, deux types principaux de codage sont spécifiés. L'intracodage est fait sans référence aux autres images. L'intracodage peut fournir des points d'accès à la séquence codée lorsque le décodage peut commencer et se poursuivre correctement mais, en principe, il ne donne qu'une efficacité de compression modérée. L'intercodage (prédictif ou biprédictif) est plus efficace car il utilise l'interprédiction pour chaque bloc de valeurs d'échantillon à partir de certaines images précédemment décodées, choisies par le codeur. A la différence de certaines autres normes de codage vidéo, les images codées à l'aide de l'interprédiction biprédictive peuvent aussi être utilisées comme références pour l'intercodage d'autres images.

L'application des trois types de codage aux images d'une séquence est souple, et l'ordre du processus de décodage n'est généralement pas le même que le processus de capture de l'image source dans le codeur ou l'ordre de sortie du décodeur pour l'affichage. Le choix est laissé au codeur et dépendra des exigences de l'application. L'ordre de décodage est spécifié de telle façon que le décodage des images qui utilise la prédiction interimage suive plus tard dans l'ordre de décodage les autres images qui sont référencées dans le processus de décodage.

0.6.2 Codage de vidéos progressives et entrelacées

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale spécifie une syntaxe et un processus de décodage pour les vidéos qui ont été produites par balayage progressif ou entrelacé, qui peuvent être entremêlées dans la même séquence. Les deux sous-trames d'une trame entrelacée sont séparées au moment de la capture, tandis que les deux sous-trames d'une trame progressive partagent le même temps de capture. Chaque sous-trame peut être codée séparément ou bien les deux sous-trames peuvent être codées ensemble comme une trame. Les trames progressives sont en général codées comme une trame. Pour la vidéo entrelacée, le codeur peut choisir entre le codage de trame et le codage de sous-trame. Le codage de trame ou le codage de sous-trame peuvent être choisis de façon adaptative, image par image, et aussi sur une base plus locale au sein d'une trame codée. Le codage de trame est généralement préféré lorsque la scène vidéo contient des détails significatifs avec un mouvement limité. Le codage de sous-trame fonctionne généralement mieux lorsqu'il y a un mouvement rapide d'image à image.

0.6.3 Partage d'image en macroblocs et en partitions plus petites

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Comme dans les Recommandations et Normes internationales antérieures sur le codage vidéo, un macrobloc, consistant en un bloc d'échantillons luma de 16x16 et deux blocs correspondants d'échantillons chroma, est utilisé comme unité de base de traitement du processus de décodage vidéo.

Un macrobloc peut encore être partagé pour l'interprédiction. Le choix de la taille des partitions d'interprédiction est le résultat d'un compromis entre le gain du codage procuré par l'utilisation de la compensation du mouvement avec de plus petits blocs et la quantité de données nécessaires afin de représenter les données pour la compensation du mouvement. Dans la présente Recommandation | Norme internationale, le processus d'interprédiction peut former des segmentations pour une représentation du mouvement de taille aussi petite que des échantillons luma de 4x4, en utilisant une précision de vecteur cinétique d'un quart du déplacement spatial de la grille de l'échantillon luma. Le processus d'interprédiction d'un bloc d'échantillon peut aussi impliquer le choix d'une image à utiliser comme image de référence pour un certain nombre d'images mémorisées précédemment décodées. Les vecteurs cinétiques sont codés de façon différentielle par rapport aux valeurs prévues, formées à partir des vecteurs cinétiques codés proches.

Généralement, le codeur calcule les vecteurs de mouvement appropriés et les autres éléments de données représentés dans le flux de données vidéo. Ce processus d'estimation du mouvement dans le codeur et le choix entre l'utilisation de l'interprédiction pour la représentation de chaque région du contenu de la vidéo ne sont pas spécifiés dans la présente Recommandation | Norme internationale.

0.6.4 Réduction de la redondance spatiale

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Les images sources et les résidus de prédiction ont tous deux une forte redondance spatiale. La présente Recommandation | Norme internationale se fonde sur l'utilisation de la méthode de la transformée à base de bloc afin de supprimer la redondance spatiale. Après l'interprédiction à partir d'échantillons précédemment décodés dans d'autres images ou après la prédiction fondée sur l'espace à partir d'échantillons précédemment décodés au sein de l'image en cours, le résidu de prédiction résultant est partagé en blocs de 4x4. Ceux-ci sont convertis dans le domaine de transformation où ils sont quantifiés. Après quantification, de nombreux coefficients de transformation sont des zéros et ont une faible amplitude. Ils peuvent alors être représentés par un faible nombre de données codées. Le processus de transformation et de quantification dans le codeur n'est pas spécifié dans la présente Recommandation | Norme internationale.

0.7 Mode de lecture de la présente spécification

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

On suggère que le lecteur commence au paragraphe 1 (Domaine d'application) et passe au paragraphe 3 (Définitions). Le paragraphe 6 comporte les relations géométriques de la source, de l'entrée et de la sortie du décodeur. Le paragraphe 7 (Syntaxe et sémantique) spécifie l'ordre de l'analyse syntaxique des éléments syntaxiques dans le flux de données. Voir les § 7.1 à 7.3 pour l'ordre syntaxique et voir le § 7.4 pour la sémantique; c'est-à-dire, la portée, les interdictions et les conditions qui sont exigées pour les éléments syntaxiques. L'analyse syntaxique réelle pour la plupart des éléments syntaxiques est spécifiée au paragraphe 9 (Processus d'analyse syntaxique). Finalement, le paragraphe 8 (Processus de décodage) spécifie comment les éléments syntaxiques sont mappés dans les échantillons décodés. Tout au long de la lecture de la présente spécification, le lecteur devrait se référer aux paragraphes 2 (Références normatives), 4 (Abréviations), et 5 (Conventions) en tant que de besoin. Les Annexes A à E font également partie intégrante de la présente Recommandation | Norme internationale.

L'Annexe A spécifie sept profils (de base, principal, étendu, élevé, élévé 10, élevé 4:2:2 et élevé 4:4:4), chacun d'eux étant adapté à certains domaines d'application. Cette Annexe définit ce qu'on appelle les niveaux des profils. L'Annexe B spécifie la syntaxe et la sémantique d'un format de flux d'octets pour la fourniture de la vidéo codée en flux ordonné d'octets. L'Annexe C spécifie le décodeur de référence type et son utilisation afin de vérifier la conformité du flux binaire et du décodeur. L'Annexe D spécifie la syntaxe et la sémantique pour les charges utilesde messages d'informations apportant des améliorations supplémentaires. Finalement, l'Annexe E spécifie la syntaxe et la sémantique des paramètres d'information d'utilisation de la vidéo, contenus dans l'ensemble des paramètres de la séquence.

Tout au long de la présente spécification, les remarques précédées de la mention "NOTE –" sont informatives et ne font pas partie intégrante de la présente Recommandation | Norme internationale.

1 Domaine d'application

La présente Recommandation UIT-T H.264 | Norme internationale ISO/CEI 14496-10 spécifie le codage vidéo.

2 Références normatives

Les Recommandations et autres références suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | Norme internationale. Au moment de la publication, les éditions indiquées étaient en vigueur. Toute Recommandation et normes sont sujettes à révision; et tous les utilisateurs de la présente Recommandation | Norme internationale sont donc invités à rechercher la possibilité d'appliquer les éditions les plus récentes des Recommandations et normes indiquées ci-après. Les membres de la CEI et de l'ISO tiennent des registres des normes internationales en vigueur. Le Bureau de normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT-T en vigueur.

- Recommandation UIT-T T.35 (2000), *Procédure d'attribution des codes définis par l'UIT-T pour les facilités non normalisées*.
- ISO/CEI 11578:1996, Annexe A, *Identificateur unique universel*.
- ISO/CIE 10527:1991, *Observateurs de référence colorimétriques CIE*.

3 Définitions

Pour les besoins de la présente Recommandation | Norme internationale, les définitions suivantes s'appliquent.

3.1 **unité d'accès**: ensemble d'unités *NAL* qui contient toujours exactement 1 *image codée primaire*. En plus de l'*image codée primaire*, une unité d'accès peut aussi contenir une ou plusieurs *images codées redondantes* ou

d'autres *unités NAL* ne contenant pas de *tranches* ou de *division de données par tranche* d'une *image codée*. Le décodage d'une unité d'accès a toujours pour résultat une *image décodée*.

3.2 **coefficient de transformée AC *(AC transform coefficient)***: tout *coefficient de transformée* pour lequel l'*indice de fréquence* dans une des deux ou dans les deux dimensions est différent de zéro.

3.3 **processus de décodage arithmétique binaire adaptatif**: *processus de décodage* entropique qui calcule les valeurs des *segments* d'un *flux binaire* produits par un *processus de codage arithmétique binaire adaptatif*.

3.4 **processus de codage arithmétique binaire adaptatif**: *processus de codage* entropique, non spécifié de façon normative dans la présente Recommandation | Norme internationale, qui code une séquence de *segments* et produit un *flux binaire* qui peut être décodé au moyen du *processus de décodage arithmétique binaire adaptatif*.

3.5 **fusion alpha; superposition de texture**: processus non spécifié par la présente Recommandation | Norme internationale, dans lequel une *image codée auxiliaire* est utilisée en combinaison avec une *image codée primaire* et avec d'autres données non spécifiées dans la présente Recommandation | Norme internationale, au cours du *processus d'affichage*. Dans un processus de fusion alpha, les échantillons d'une *image codée auxiliaire* sont interprétés comme étant des indications du degré d'opacité (ou, ce qui est équivalent, comme des indications des degrés de transparence) associé aux échantillons *luma* correspondants de l'*image codée primaire*.

3.6 **ordre arbitraire par tranches**: *ordre de décodage* en *tranches* dans lequel l'*adresse de macrobloc* du premier *macrobloc* de certaines *tranches* d'une *image* peut être inférieure à l'*adresse de macrobloc* du premier *macrobloc* de certaines autres *tranches* précédentes de la même *image codée*.

3.7 **image codée auxiliaire**: *image* qui s'ajoute à l'*image codée primaire* et qui peut être utilisée en combinaison avec d'autres données non spécifiées par la présente Recommandation | Norme internationale, au cours du *processus d'affichage*. Une image codée auxiliaire est soumise aux mêmes restrictions syntaxiques et sémantiques qu'une *image codée redondante* monochrome. Une image codée auxiliaire doit toujours contenir le même nombre de *macroblocs* que l'*image codée primaire*. Les images codées auxiliaires n'exercent aucun effet normatif sur le *processus de décodage*. Voir également les termes *image codée primaire* et *image codée redondante.*

3.8 **tranche B**: *tranche* qui peut être codée au moyen de l'*intraprédiction* à partir d'échantillons décodés au sein de la même *tranche* ou au moyen de l'*interprédiction* à partir d'*images de référence* précédemment décodées, en utilisant au plus deux *vecteurs cinétiques* et des *indices de référence* afin de *prédire* les valeurs d'échantillon de chaque *bloc*.

3.9 **segment**: bit unique d'une *chaîne de segments*.

3.10 **binarisation**: ensemble de *chaînes de segments* pour toutes les valeurs possibles d'un *élément syntaxique*.

3.11 **processus de binarisation**: processus de mise en correspondance (mappage) unique de toutes les valeurs possibles d'un élément *syntaxique* avec un ensemble de *chaînes de segments*.

3.12 **chaîne de segments**: représentation binaire intermédiaire des valeurs des *éléments syntaxiques* obtenues par la *binarisation* de l'*élément syntaxique*.

3.13 **tranche biprédictive**: voir *tranche B.*

3.14 **flux binaire**: séquence de bits qui forme la représentation des *images codées* et des données associées qui constituent une ou plusieurs *séquences vidéo codées*. Le flux binaire est un terme collectif utilisé afin de désigner soit un *flux d'unités NAL* soit un *flux d'octets*.

3.15 **bloc**: matrice MxN (M colonnes sur N rangées) d'échantillons, ou matrice MxN de *coefficients de transformée*.

3.16 **sous-trame inférieure**: l'une des deux *sous-trames (ou monotrames)* que comporte une *trame (ou bi-trame)*. Chaque rangée d'une sous-trame inférieure est localisée spatialement immédiatement en dessous de la rangée correspondante d'une *sous-trame supérieure*.

3.17 **macrobloc inférieur (d'une paire de macroblocs)**: *macrobloc* qui contient, au sein d'une *paire de macroblocs,* les échantillons de la rangée inférieure d'échantillons pour cette *paire de macroblocs.* Pour une *paire de macroblocs de sous-trame*, le macrobloc inférieur représente les échantillons provenant de la région de la *sous-trame inférieure* de la *trame* qui s'inscrit dans la région spatiale de la paire de *macroblocs.* Pour une *paire de macroblocs de trame*, le macrobloc inférieur représente les échantillons de la *trame* qui s'inscrit dans la moitié inférieure de la région spatiale de la *paire de macroblocs*.

3.18 **rupture de liaison**: localisation dans un flux binaire à laquelle il est indiqué que certaines *images* suivantes dans l'*ordre de décodage* peuvent contenir des distorsions visuelles importantes, dues à des opérations non spécifiées qui ont été effectuées au cours de la production du *flux binaire*.

3.19 **octet**: séquence de 8 bit, écrite et lue avec le bit de plus fort poids à gauche et le bit de plus faible poids à droite. Dans la représentation d'une séquence de bits de données, le bit de plus fort poids est le premier.

3.20 **aligné par les octets**: une position d'un *flux binaire* est alignée par les octets lorsque cette position est un multiple entier de 8 bit à partir de la position du premier bit dans le *flux binaire*. Un bit ou *octet* ou *élément syntaxique* est dit *aligné par les octets* lorsque la position où il apparaît dans un *flux binaire* est alignée par les octets.

3.21 **flux d'octets**: incorporation d'un *flux d'unités NAL* contenant des *préfixes de code de démarrage* et des *unités NAL* comme spécifié à l'Annexe B.

3.22 **peut (peuvent)**: forme verbale utilisée afin de désigner un comportement autorisé mais pas nécessairement obligatoire.

3.23 **catégorie**: nombre associé à chaque *élément syntaxique*. La catégorie sert à spécifier l'allocation des *éléments syntaxiques* aux *unités NAL* pour le *partage des données en tranches*. Elle peut aussi servir d'une façon déterminée par l'application afin de faire référence aux classes d'*éléments syntaxiques* d'une façon non spécifiée dans la présente Recommandation | Norme internationale.

3.24 **chroma**: adjectif spécifiant qu'un tableau d'échantillons ou un simple échantillon représente un des deux signaux de différence de couleur se rapportant aux couleurs primaires. Les symboles utilisés pour un tableau ou échantillon chroma sont Cb et Cr.

NOTE – Le terme *chroma* est utilisé de préférence au terme *chrominance* afin d'éviter l'implication de l'utilisation de caractéristiques linéaires de transfert de lumière qui est souvent associée au terme *chrominance*.

3.25 **sous-trame codée**: *représentation codée* d'une *sous-trame*.

3.26 **trame codée**: *représentation codée* d'une *trame*.

3.27 **image codée**: *représentation codée* d'une *image*. Une image codée peut être soit une *sous-trame codée* soit une *trame codée*. *Image codée* est un terme collectif se rapportant à une *image codée primaire* ou à une *image codée redondante*, mais pas aux deux à la fois.

3.28 **(mémoire) tampon des images codées (CPB, *coded picture buffer*)**: tampon de type premier entré premier sorti contenant des *unités d'accès* dans l'*ordre de décodage* spécifié dans le *décodeur fictif de référence* de l'Annexe C.

3.29 **représentation codée**: élément de données tel que représenté sous sa forme codée.

3.30 **séquence vidéo codée**: séquence d'*unités d'accès* qui consiste, en ordre de décodage, en une *unité d'accès IDR* suivie de zéro, une ou plusieurs *unités d'accès* non IDR y compris toutes les *unités d'accès* suivantes jusqu'à, mais non compris, toute *unité d'accès IDR* suivante.

3.31 **composante**: matrice ou simple échantillon de l'une des trois matrices (une *luma* et deux *chroma*) qui forment une *sous-trame* ou une *trame*.

3.32 **paire de sous-trames complémentaires**: terme collectif pour une *paire de sous-trames de référence complémentaires* ou une *paire de sous-trames non référentielles complémentaires*.

3.33 **paire de sous-trames non référentielles complémentaires**: ensemble de deux *sous-trames non référentielles* se trouvant dans des *unités d'accès* consécutives en *ordre de décodage* en tant que deux *sous-trames codées* de parité opposée lorsque la première *sous-trame* n'est pas déjà une *sous-trame* appariée.

3.34 **paire de sous-trames de référence complémentaires**: ensemble de deux *sous-trames de référence* se trouvant dans des *unités d'accès* consécutives en *ordre de décodage* en tant que deux *sous-trames codées* qui partagent la même valeur de l'*élément syntaxique* frame_num (numéro de trame), où la seconde *sous-trame* dans l'*ordre de décodage* n'est pas une*image à rafraîchissement IDR* et n'inclut pas un élément syntaxique memory_management_control_operation (opération de commande de gestion de la mémoire) égal à 5.

3.35 **variable de contexte**: variable spécifiée pour le *processus de décodage arithmétique binaire adaptatif* d'un *segment* par une équation contenant des *segments* récemment décodés.

3.36 **coefficient de transformée DC (apériodique)**: *coefficient de transformée* pour lequel l'*indice de fréquence* est zéro dans toutes les dimensions.

3.37 **image décodée**: *image* déduite en décodant une *image codée.* Une *image décodée* est soit une *trame* décodée, soit une *sous-trame* décodée. Une *sous-trame* décodée est soit une *sous-trame supérieure* décodée soit une *sous-trame inférieure* décodée.

3.38 **(mémoire) tampon d'image décodée (DPB, *decoded picture buffer*)**: mémoire tampon contenant des *images décodées* servant de référence, de réorganisation de sortie et de retard de sortie, qui est spécifiée pour le *décodeur fictif de référence* de l'Annexe C.

3.39 **décodeur**: matérialisation d'un *processus de décodage*.

3.40 **ordre de décodage**: ordre dans lequel les *éléments syntaxiques* sont traités par le *processus de décodage*.

3.41 **processus de décodage**: processus spécifié dans la présente Recommandation | Norme internationale, qui lit un *flux binaire* et calcule des *images décodées* à partir de celui-ci.

3.42 **prédiction directe**: *interprédiction* pour un *bloc* dans lequel il n'y a pas de décodage de *vecteur cinétique*. Deux modes de *prédiction* directe sont spécifiés, qui sont appelés mode de *prédiction* directe spatiale et mode de *prédiction* temporelle.

3.43 **processus d'affichage**: opération non spécifiée dans la présente Recommandation | Norme internationale, dont l'entrée est constituée par les *images* décodées et recadrées qui forment la sortie du *processus de décodage*.

3.44 **décodeur soumis aux essais (DUT, *decoder under test*)**: *décodeur* dont on vérifie la conformité à la présente Recommandation | Norme internationale en faisant délivrer à ce *décodeur* et au *décodeur fictif de référence* un *flux binaire* conforme par l'*ordonnanceur fictif de flux binaires* et en comparant les valeurs et la temporisation de sortie des deux *décodeurs*.

3.45 **octet de prévention d'émulation**: octet égal à 0x03 qui peut être présent au sein d'une *unité NAL*. La présence d'octets de prévention d'émulation garantit que, dans l'*unité NAL*, aucune séquence d'octets consécutifs alignés par les octets ne contient de *préfixe de code de début*.

3.46 **codeur**: matérialisation d'un *processus de codage*.

3.47 **processus de codage**: processus, non spécifié dans la présente Recommandation | Norme internationale, qui produit un *flux binaire* conforme à la présente Recommandation | Norme internationale.

3.48 **sous-trame**: assemblage de rangées alternées d'une *trame*. Une *trame* se compose de deux *sous-trames*: une *sous-trame supérieure* et une *sous-trame inférieure*.

3.49 **macrobloc de sous-trame**: macrobloc contenant des échantillons provenant d'une seule *sous-trame*. Tous les *macroblocs* d'une *sous-trame codée* sont des macroblocs de sous-trame. Lorsqu'on utilise le *décodage de trame/sous-trame de macrobloc adaptatif*, certains macroblocs d'une *trame codée* peuvent être des macroblocs de sous-trame.

3.50 **paire de macroblocs de sous-trame**: paire de *macroblocs* décodés comme deux *macroblocs de sous-trame*.

3.51 **balayage de sous-trame**: ordonnancement séquentiel spécifique des *coefficients de transformée* qui diffère du balayage en *zigzag* en balayant les colonnes plus rapidement que les rangées. Le balayage de sous-trame est utilisé pour les *coefficients de transformée* dans les *macroblocs de sous-trame*.

3.52 **fanion**: variable qui peut prendre une des deux valeurs possibles de 0 et 1.

3.53 **trame**: une *trame* contient une série matricielle d'échantillons luma et deux séries matricielles correspondantes d'échantillons chroma. Une *trame* consiste en deux *sous-trames*: une *sous-trame supérieure* et une *sous-trame inférieure*.

3.54 **macrobloc de trame**: *macrobloc* représentant des échantillons des deux *sous-trames* d'une *trame codée*. Lorsqu'on n'utilise pas le *décodage de trame/sous-trame de macrobloc adaptatif*, tous les macroblocs d'une trame codée sont des macroblocs de trame. Lorsqu'on utilise le *décodage de trame/sous-trame de macrobloc adaptatif*, certains macroblocs d'une *trame codée* peuvent être des macroblocs de trame.

3.55 **paire de macroblocs de trame**: *paire de macroblocs* décodés comme deux *macroblocs de trame*.

3.56 **indice de fréquence**: indice à une ou à deux dimensions, associé à un *coefficient de transformée* avant une partie de *transformation inverse* du *processus de décodage.*

3.57 **décodeur fictif de référence (HRD, *hypothetical reference decoder*)**: modèle de *décodeur* fictif qui spécifie les contraintes en matière de variation des *flux d'unités NAL* conformes ou de *flux d'octet*s conformes que peut produire le processus de codage.

3.58 **ordonnanceur fictif de flux binaires (HSS, *hypothetical stream scheduler*)**: mécanisme de fourniture fictive de la temporisation et du flux de données à l'entrée d'un *flux binaire* au *décodeur fictif de référence*. L'ordonnanceur HSS est utilisé afin de vérifier la conformité d'un *flux binaire* ou d'un *décodeur*.

3.59 **tranche I**: *tranche* autre qu'une *tranche SI*, qui est décodée en effectuant uniquement la *prédiction* à partir d'échantillons décodés au sein de la même *tranche*.

3.60 **informatif**: terme utilisé afin de désigner les éléments de la présente Recommandation | Norme internationale qui ne font pas partie intégrante de celle-ci. Les éléments informatifs ne créent aucune obligation de se conformer à la présente Recommandation | Norme internationale.

3.61 **unité d'accès à rafraîchissement de décodage instantané (IDR, *instantaneous decoding refresh*)**: *unité d'accès* dans laquelle l'*image codée primaire* est une*image à rafraîchissement IDR*.

3.62 **image à rafraîchissement de décodage instantané (IDR)**: *image codée* dans laquelle toutes les *tranches* sont des tranches *I* ou *SI*, qui amène le *processus de décodage* à marquer toutes les *images de référence* comme étant "inutilisée pour référence" immédiatement après le décodage de l'image à rafraîchissement IDR. Après le décodage d'uneimage à rafraîchissement IDR, toutes les images codées suivantes en ordre de décodage peuvent être décodées sans *interprédiction* à partir de toute image décodée avant l'image à rafraîchissement IDR. La première *image* de chaque *séquence vidéo codée* est uneimage à rafraîchissement IDR.

3.63 **intercodage**: codage d'un *bloc*, d'un *macrobloc*, d'une *tranche* ou d'une *image* qui utilise l'*interprédiction*.

3.64 **interprédiction**: *prédiction* déduite d'échantillons décodés d'*images de référence* autres que l'*image décodée* en cours.

3.65 **valeur d'échantillon d'interprétation**: valeur éventuellement altérée qui correspond à une valeur d'échantillon décodé d'une *image codée auxiliaire* pouvant être produite afin de servir au *processus d'affichage*. Des valeurs d'échantillon d'interprétation ne sont pas utilisées dans le *processus d'affichage* et n'ont aucun effet normatif sur le *processus de décodage*.

3.66 **intracodage**: codage d'un *bloc*, d'un *macrobloc*, d'une *tranche* ou d'une *image* qui utilise l'*intraprédiction*.

3.67 **intraprédiction**: *prédiction* déduite des échantillons décodés de la même *tranche décodée*.

3.68 **intratranche**: voir *tranche I*.

3.69 **transformation inverse**: partie du *processus de décodage* par laquelle un ensemble de *coefficients de transformée* sont convertis en valeurs de domaine spatial, ou par laquelle un ensemble de *coefficients de transformée* sont convertis en *coefficients de transformée DC*.

3.70 **couche**: élément d'un ensemble de structures syntaxiques dans une relation hiérarchique sans embranchement. Les couches supérieures contiennent les couches inférieures. Les couches de codage sont les couches de *séquence vidéo codée*, d'*image*, de *tranche* et de *macrobloc*.

3.71 **niveau**: ensemble défini de contraintes sur les valeurs que peuvent prendre les *éléments syntaxiques* et variables de la présente Recommandation | Norme internationale. Le même ensemble de niveaux est défini pour tous les *profils*, la plupart des aspects de la définition de chaque niveau étant communs à travers les différents *profils.* Les mises en œuvre individuelles peuvent, avec des contraintes spécifiques, accepter un niveau différent pour chaque *profil* géré. Dans un contexte différent, le niveau est la valeur d'un *coefficient de transformée* avant la *normalisation.*

3.72 **vecteur cinétique de liste 0 (liste 1)**: *vecteur cinétique* associé à un *indice de référence* portant sur une *liste 0* (*liste 1) d'images de référence*.

3.73 **prédiction de liste 0 (liste 1)**: *interprédiction* du contenu d'une *tranche* utilisant un *indice de référence* portant sur une *liste 0* (*liste 1) d'images de référence*.

3.74 **luma**: adjectif spécifiant qu'une série matricielle d'échantillons ou un simple échantillon représente le signal monochromatique qui se rapporte aux couleurs primaires. Le symbole ou l'indice utilisé pour luma est Y ou L.

NOTE – L'utilisation du terme *luma* est préférée à celle du terme *luminance* afin d'éviter les implications de l'utilisation des caractéristiques linéaires de transfert de la lumière qui sont souvent associées au terme *luminance*. Le symbole L est parfois utilisé au lieu du symbole Y afin d'éviter toute confusion avec le symbole Y utilisé sur l'axe des ordonnées.

3.75 **macrobloc**: *bloc* d'échantillons *luma* de 16x16 et deux *blocs* correspondants d'échantillons *chroma*. La subdivision d'une *tranche* ou d'une *paire de macroblocs* en macroblocs est une *subdivision*.

3.76 **décodage de trame/sous-trame de macrobloc adaptatif**: *processus de décodage* pour *trames codées* dans lequel certains *macroblocs* peuvent être décodés comme *macroblocs de trame* et d'autres peuvent être décodés comme *macroblocs de sous-trame.*

3.77 **adresse de macrobloc**: lorsqu'on n'utilise pas le *décodage de trame/sous-trame de macrobloc adaptatif*, une adresse de macrobloc est l'indice d'un macrobloc dans un *balayage matriciel de macrobloc* de l'*image* commençant par zéro pour le *macrobloc* supérieur gauche dans une *image*. Lorsqu'on utilise le *décodage de trame/sous-trame de macrobloc adaptatif*, l'adresse du *macrobloc supérieur* d'une *paire de macroblocs* est deux fois l'indice d'une *paire de macroblocs* dans un *balayage matriciel de paire de macroblocs* de l'*image*, et l'adresse du *macrobloc inférieur* d'une *paire de macroblocs* est l'adresse du *macrobloc supérieur* correspondant plus 1. L'adresse du *macrobloc supérieur* de chaque *paire de macroblocs* est un nombre pair et l'adresse du *macrobloc inférieur* de chaque *paire de macroblocs* est un nombre impair.

3.78 **localisation de macrobloc**: coordonnées bidimensionnelles d'un *macrobloc* dans une *image*, notées (x, y). Pour le *macrobloc* supérieur gauche de l'*image*, la localisation (x, y) est égale à (0, 0). La valeur de x est incrémentée de 1 pour chaque colonne de *macrobloc*, de gauche à droite. Lorsqu'on n'utilise pas le *décodage de trame/sous-trame de macrobloc adaptatif*, la valeur de y est incrémentée de 1 pour chaque rangée de *macrobloc*, de haut en bas. Lorsqu'on utilise le *décodage de trame/sous-trame de macrobloc adaptatif*, la valeur de y est incrémentée de 2 pour chaque rangée de *paire de macroblocs* de haut en bas, et est en plus incrémentée de 1 lorsqu'un macrobloc est un *macrobloc inférieur*.

3.79 **paire de macroblocs**: paire de *macroblocs* verticalement contigus dans une *trame* qui est couplée afin d'être utilisée dans le *décodage de trame/sous-trame de macrobloc adaptatif*. La division d'une *tranche* en paires de macroblocs est une *subdivision.*

3.80 **subdivision de macrobloc**: un *bloc* d'échantillons *luma* et deux *blocs* d'échantillons *chroma* correspondants résultant de la *subdivision* d'un *macrobloc* pour l'*interprédiction*.

3.81 **mappage de macrobloc à un groupe de tranches**: moyen de répartir les *macroblocs* d'une *image* en *groupes de tranches*. Le mappage de macrobloc à un groupe de tranches consiste en une liste de numéros, un pour chaque *macrobloc codé*, spécifiant le *groupe de tranches* auquel appartient chaque *macrobloc* codé.

3.82 **mappage d'unité de répartition à un groupe de tranches**: moyen de faire correspondre les *unités de répartition de groupe de tranches* d'une *image* avec des *groupes de tranches*. Le mappage d'unité de répartition à un groupe de tranches consiste en une liste de numéros, un pour chaque *unité de répartition de groupe de tranches*, spécifiant le *groupe de tranches* auquel appartient chaque *unité de répartition de groupe de tranches*.

3.83 **a (ont) la possibilité**: expression utilisée afin de désigner un comportement autorisé, mais pas nécessairement obligatoire. Parfois, lorsqu'on souhaite souligner le caractère facultatif du comportement décrit, on utilise à cet effet l'expression "a (ont) éventuellement la possibilité de".

3.84 **opération de commande de gestion de mémoire**: sept opérations qui commandent le *marquage d'image de référence*.

3.85 **vecteur cinétique**: vecteur bidimensionnel utilisé pour l'*interprédiction* qui donne le décalage à partir des coordonnées dans l'*image décodée* jusqu'aux coordonnées situées dans une *image de référence*.

3.86 **doit (doivent) toujours**: terme utilisé afin d'exprimer une observation relative à une obligation ou à une conséquence d'une obligation qui est spécifiée ailleurs dans la présente Recommandation | Norme internationale. Ce terme est utilisé exclusivement dans un contexte *informatif*.

3.87 **unité (de couche) (NAL*, network abstraction layer*)**: structure syntaxique contenant une indication du type des données qui suivent et des *octets* contenant ces données sous la forme d'une charge utile *RBSP* entrecoupée, en tant que de besoin, d'*octets de prévention d'émulation*.

3.88 **flux d'unités NAL**: séquence d'*unités NAL*.

3.89 **sous-trame non appariée**: terme collectif désignant une *sous-trame de référence non appariée* ou une *sous-trame non référentielle non appariée*.

3.90 **sous-trame non référentielle non appariée**: *sous-trame non référentielle* décodée qui ne fait pas partie d'une *paire de sous-trames non référentielles complémentaires.*

3.91 **sous-trame de référence non appariée**: *sous-trame de référence* décodée qui ne fait pas partie d'une paire de *sous-trames de référence complémentaires.*

3.92 **sous-trame non référentielle**: *sous-trame* codée avec nal_ref_idc égal à 0.

3.93 **trame non référentielle**: *trame* codée avec nal_ref_idc égal à 0.

3.94 **image non référentielle**: *image* codée avec nal_ref_idc égal à 0. On n'utilise pas d'*image non référentielle* pour l'*interprédiction* d'autres *images*.

3.95 **note**: terme utilisé afin d'introduire des observations *informatives*. Ce terme est utilisé exclusivement dans un contexte *informatif*.

3.96 **parité opposée**: la *parité opposée* du *haut* est le *bas* et vice versa.

3.97 **ordre de sortie**: ordre dans lequel les *images décodées* sont extraites de la *mémoire tampon d'images décodées*.

3.98 **tranche P**: *tranche* qui peut être décodée à l'aide de l'*intraprédiction* à partir d'échantillons décodés au sein de la même *tranche* ou à l'aide de l'*interprédiction* à partir d'*images de référence* précédemment décodées, au moyen d'un seul *vecteur cinétique* au plus et d'un seul *indice de référence* au plus afin de *prédire* les valeurs d'échantillon de chaque *bloc*.

3.99 **paramètre**: *élément syntaxique d'un ensemble de paramètres de séquence* ou d'un *ensemble de paramètres d'image*. Ce terme est aussi utilisé comme partie de l'expression définie de *paramètre de quantification*.

3.100 **parité**: la *parité* d'une *sous-trame* peut être *supérieure* ou *inférieure*.

3.101 **subdivision**: division d'un ensemble en sous-ensembles de telle sorte que chaque élément de l'ensemble soit exactement dans un seul des sous-ensembles.

3.102 **image**: terme collectif désignant une *sous-trame* ou une *trame*.

3.103 **ensemble de paramètres d'image**: *structure syntaxique* contenant des *éléments syntaxiques* s'appliquant à zéro, une ou plusieurs *images codées* entières, telles que déterminées par l'*élément syntaxique* pic_parameter_set_id (identification d'ensemble de paramètres d'image) figurant dans chaque *en-tête de tranche*.

3.104 **compte d'ordre d'image**: variable ayant une valeur non décroissante avec une position d'*image* croissante dans l'ordre de sortie, soit par rapport à l'*image à rafraîchissement IDR* précédente dans l'*ordre de décodage* ou par rapport à la précédente *image* contenant l'*opération de commande de gestion de mémoire* qui marque toutes les *images de référence* comme étant "inutilisée comme référence".

3.105 **prédiction**: matérialisation du *processus de prédiction*.

3.106 **processus de prédiction**: utilisation d'un *prédicteur* afin de fournir une estimation de la valeur d'échantillon ou de l'élément de données en cours de décodage.

3.107 **tranche prédictive**: voir *tranche P*.

3.108 **prédicteur**: combinaison de valeurs spécifiées ou de valeurs d'échantillons décodés antérieurement ou d'éléments de données utilisées dans le *processus de décodage* de valeurs d'échantillons ou d'éléments de données subséquents.

3.109 **image codée primaire**: représentation codée d'une *image* à utiliser dans le *processus de décodage* pour un flux binaire conforme à la présente Recommandation | Norme internationale. L'image codée primaire contient tous les *macroblocs* de l'*image.* Les seules *images* qui ont un effet normatif sur le *processus de décodage* sont des images codées primaires. Voir aussi *image codée redondante.*

3.110 **profil**: sous-ensemble défini de la syntaxe de la présente Recommandation | Norme internationale.

3.111 **paramètre de quantification**: variable utilisée par le *processus de décodage* pour la normalisation des *niveaux de coefficient de transformée*.

3.112 **accès aléatoire**: action de démarrage du processus de décodage pour un *flux binaire* à un point autre que le début de ce flux.

3.113 **balayage matriciel**: mise en correspondance (mappage) d'un canevas rectangulaire bidimensionnel avec un canevas unidimensionnel, de telle sorte que les premières entrées dans le canevas unidimensionnel proviennent de la première rangée supérieure du canevas bidimensionnel balayé de gauche à droite, suivi de la même façon par la seconde, la troisième, etc., rangée du canevas (en descendant), chacune d'elle étant balayée de gauche à droite.

3.114 **charge utile de séquence d'octets bruts (RBSP, *raw byte sequence payload*)**: structure syntaxique contenant un nombre entier d'octets, qui est encapsulée dans une *unité NAL*. Une charge utile RBSP est soit vide soit formée d'une *chaîne de bits de données* contenant des *éléments syntaxiques* suivis par un *bit d'arrêt de charge utile RBSP* et suivis par zéro, un ou plusieurs bits subséquents de valeur égale à 0.

3.115 **bit d'arrêt de charge utile de séquence d'octets bruts (RBSP)**: bit égal à 1, présent au sein d'une *charge utile de séquence d'octets bruts (RBSP)* après une *chaîne de bits de données*. La localisation de la fin de la *chaîne de bits de données* au sein d'une charge utile *RBSP* peut être identifiée en cherchant à partir de la fin de la charge utile *RBSP* le *bit d'arrêt RBSP*, qui est le dernier bit non égal à zéro dans la charge utile *RBSP*.

3.116 **point de rétablissement**: point dans le *flux binaire* auquel le rétablissement d'une représentation exacte ou approximative des *images décodées* représentées par le *flux binaire* est réalisé après un *accès aléatoire* ou une *rupture de liaison*.

3.117 **image codée redondante**: représentation codée d'une *image* ou d'une partie d'*image*. Le contenu d'une image codée redondante ne doit pas être utilisé par le *processus de décodage* pour un *flux binaire* conforme à la présente Recommandation | Norme internationale. Une *image codée redondante* ne doit pas nécessairement contenir tous les *macroblocs* de l'*image codée primaire*. Les images codées redondantes n'ont pas d'effet normatif sur le *processus de décodage*. Voir aussi *image codée primaire*.

3.118 **sous-trame de référence**: une *sous-trame de référence* peut être utilisée pour l'*interprédiction* lorsque des *tranches P*, *SP* et *B* d'une *sous-trame codée* ou des *macroblocs de sous-trame* d'une *trame codée* sont décodés. Voir aussi à *image de référence*.

3.119 **trame de référence**: on peut utiliser une *trame de référence* pour l'*interprédiction* lorsque des tranches *P*, *SP* et *B* d'une *trame codée* sont décodées. Voir aussi *image de référence*.

3.120 **indice de référence**: indice contenu dans une *liste d'images de référence*.

3.121 **image de référence**: *image* avec nal_ref_idc non égal à 0. Une *image de référence* contient des échantillons qui peuvent être utilisés pour l'*interprédiction* dans le *processus de décodage* des *images* suivantes dans l'*ordre de décodage*.

3.122 **liste d'images de référence**: liste d'*images de référence* utilisée pour l'*interprédiction* d'une *tranche P, B* ou *SP*. Dans le *processus de décodage* d'une *tranche P* ou *SP*, il existe une seule liste d'images de référence. Pour le *processus de décodage* d'une *tranche B*, il existe deux listes d'images de référence.

3.123 **liste 0 d'images de référence**: *liste d'images de référence* utilisée pour l'*interprédiction* d'une *tranche P*, *B* ou *SP*. Toute *interprédiction* utilisée pour des tranches *P* et *SP* utilisent une liste 0 d'images de référence. Une liste 0 d'images de référence est une des deux listes d'*images de référence* utilisées pour l'*interprédiction* pour une *tranche B*, l'autre étant une *liste 1 d'images de référence*.

3.124 **liste 1 d'images de référence**: *liste d'images de référence* utilisée pour l'*interprédiction* d'une *tranche B*. Une liste 1 d'images de référence est une des deux listes d'*images de référence* utilisées pour l'*interprédiction* pour une *tranche B*, l'autre étant la *liste 0 d'images de référence*.

3.125 **marquage d'image de référence**: spécifie, dans le flux binaire, comment sont marquées les *images décodées* pour l'*interprédiction*.

3.126 **réservé**: lorsqu'il est utilisé dans les clauses spécifiant des valeurs d'un *élément syntaxique* particulier, ce terme indique une utilisation future par l'UIT-T | ISO/CEI. Ces valeurs ne doivent pas être utilisées dans des *flux binaires* conformes à la présente Recommandation | Norme internationale, mais pourront être utilisées dans des extensions futures de la présente Recommandation | Norme internationale, élaborées par l'UIT-T | ISO/CEI.

3.127 **résidu**: différence décodée entre une *prédiction* d'un échantillon ou d'un élément de données et sa valeur décodée.

3.128 **exécution**: nombre d'éléments de données consécutifs représentés dans le processus de décodage. Dans un certain contexte, c'est le nombre de *niveaux de coefficient de transformée* de valeur zéro précédant un niveau de *coefficient de transformée* de valeur différente de zéro dans la liste de *niveaux de coefficient de transformée* produits par un *balayage en zigzag* ou par un *balayage de sous-trame*. Dans d'autres contextes, ce terme se rapporte à un certain nombre de *macroblocs*.

3.129 **format d'échantillon (*SAR, sample aspect ratio*)**: afin d'aider au processus d'affichage -qui n'est pas spécifié dans la présente Recommandation | Norme internationale, ce terme spécifie le rapport entre la distance horizontale recherchée entre les colonnes et la distance verticale recherchée entre les rangées de la matrice d'échantillons *luma* dans une *trame*. Le format d'échantillon est exprimé par *h*:*v*, où *h* est la largeur horizontale et *v* est la hauteur verticale (en unités arbitraires de distance spatiale).

3.130 **normalisation**: processus de multiplication des *niveaux de coefficient de transformée* par un facteur, résultant en *coefficients de transformée*.

3.131 **ensemble de paramètres de séquence**: *structure syntaxique* contenant des *éléments syntaxiques* s'appliquant à zéro, une ou plusieurs *séquences vidéo codées* entières, telles que déterminées par le contenu d'un *élément syntaxique* seq_parameter_set_id (identificateur d'ensemble de paramètres de séquence) figurant dans l'*ensemble de paramètres d'image* visé dans *l'élément syntaxique* pic_parameter_set_id (identificateur d'ensemble de paramètres d'image) figurant dans chaque *en-tête de tranche*.

3.132 **doit (doivent)**: terme utilisé afin d'exprimer une obligation impérative de conformité à la présente Recommandation | Norme internationale. Lorsqu'il est utilisé afin d'exprimer une contrainte obligatoire relative aux valeurs d'*éléments syntaxiques* ou aux résultats obtenus par actionnement du *processus de décodage* spécifié, il relève de la responsabilité du *codeur* de faire en sorte que la contrainte soit mise en œuvre. Lorsque ce terme se rapporte à des opérations exécutées par le *processus de décodage*, tout *processus de décodage* qui produit des résultats identiques au *processus de décodage* décrit ici est conforme aux prescriptions applicables au *processus de décodage* de la présente Recommandation | Norme internationale.

3.133 **devrait (devraient)**: terme utilisé afin de désigner le mode de fonctionnement d'une mise en œuvre qu'il est préconisé d'appliquer dans les conditions habituelles prévues, mais qui n'impose aucune obligation impérative de conformité à la présente Recommandation | Norme internationale.

3.134 **tranche SI**: *tranche* qui est codée en n'utilisant la *prédiction* qu'à partir d'échantillons décodés au sein de la même *tranche* et au moyen de la quantification des échantillons de la *prédiction.* Une tranche SI peut être codée de telle sorte que ses échantillons décodés puissent être construits de façon identique à une *tranche SP*.

3.135 **macrobloc sauté**: *macrobloc* pour lequel il n'y a pas de données codées autres qu'une indication que le *macrobloc* doit être décodé comme "sauté". Cette indication peut être commune à plusieurs *macroblocs*.

3.136 **tranche**: nombre entier de *macroblocs* ou de *paires de macroblocs* ordonnés consécutivement au cours du *balayage matriciel* au sein d'un *groupe de tranches* particulier. Pour l'*image codée primaire*, la division de chaque *groupe de tranches* en tranches est une *subdivision*. Bien qu'une tranche contienne des *macroblocs* ou des *paires de macroblocs* qui sont consécutifs dans le balayage matriciel au sein d'un groupe de tranches, ces *macroblocs* ou *paires de macroblocs* ne sont pas nécessairement consécutifs dans le balayage matriciel au sein de l'*image*. Les adresses des *macroblocs* sont déduites de l'adresse du premier *macrobloc* contenu dans une tranche (comme représenté dans l'*en-tête de tranche*) et du *mappage de macrobloc à groupe de tranches.*

3.137 **subdivision de données en tranches**: méthode de *subdivision* d'*éléments syntaxiques* choisis en *structures syntaxiques* fondées sur une *catégorie* associée à chaque *élément syntaxique*.

3.138 **groupe de tranches**: sous-ensemble des *macroblocs* ou *paires de macroblocs* d'une *image*. La division de l'*image* en groupes de tranches est une *subdivision* de l'*image.* La subdivision est spécifiée par le *mappage de macrobloc à groupe de tranches*.

3.139 **unités de mappage de groupe de tranches**: unités du *mappage d'unité de répartition à groupe de tranches.*

3.140 **en-tête de tranche**: partie d'une *tranche codée* contenant les éléments de données appartenant au premier ou à tous les *macroblocs* représentés dans la tranche.

3.141 **source**: terme utilisé afin de décrire le matériel vidéo ou certains de ses attributs avant le codage.

3.142 **tranche SP**: *tranche* qui est codée au moyen de l'*interprédiction* à partir d'*images de référence* précédemment décodées, au moyen d'un seul *vecteur cinétique* et d'un seul *indice de référence* afin de *prédire* les valeurs d'échantillon de chaque *bloc.* Une tranche SP peut être codée de telle sorte que ses échantillons décodés puissent être construits de façon identique à une autre tranche SP ou *tranche SI*.

3.143 **préfixe de code de déclenchement**: séquence unique de trois *octets* égale à 0x000001 incorporée dans le *flux d'octets* comme préfixe pour chaque *unité NAL.* La localisation d'un préfixe de code de déclenchement peut être utilisée par un *décodeur* afin d'identifier le début d'une nouvelle *unité NAL* et la fin d'une *unité NAL* antérieure. On peut empêcher l'émulation des préfixes de code de déclenchement au sein des *unités NAL* en incluant des *octets de prévention d'émulation*.

3.144 **chaîne de bits de données (SODB*, string of data bits*)**: séquence d'un certain nombre de bits représentant des *éléments syntaxiques* présents au sein d'une *charge utile de séquence d'octets bruts* avant le *bit d'arrêt de charge utile de séquence d'octets bruts.* Au sein d'une chaîne *SODB*, le bit le plus à gauche est considéré comme le premier bit et celui de plus fort poids, et le bit le plus à droite est considéré comme le dernier et le bit de plus faible poids.

3.145 **sous-macrobloc**: un quart des échantillons d'un *macrobloc*, c'est-à-dire un *bloc luma* de 8x8 et deux *blocs chroma* correspondants dont un coin est situé dans un coin du *macrobloc*.

3.146 **subdivision de sous-macrobloc**: un *bloc* d'échantillons *luma* et les deux *blocs* correspondants d'échantillons *chroma* résultant d'une *subdivision* d'un *sous-macrobloc* pour l'*interprédiction*.

3.147 commutation de tranche I: voir *tranche SI*.

3.148 commutation de tranche P: voir *tranche SP*.

3.149 élément syntaxique: élément de données représenté dans le *flux binaire*.

3.150 structure syntaxique: zéro, un ou plusieurs *éléments syntaxiques* présents ensemble dans le *flux binaire* dans un ordre spécifié.

3.151 sous-trame supérieure: une des deux *sous-trames* que comporte une *trame*. Chaque rangée d'une *sous-trame supérieure* est située dans l'espace immédiatement au-dessus de la rangée correspondante de la *sous-trame inférieure*.

3.152 macrobloc supérieur (d'une paire de macroblocs): *macrobloc* au sein d'une *paire de macroblocs* qui contient les échantillons de la rangée supérieure d'échantillons pour la *paire de macroblocs*. Pour une *paire de macroblocs de sous-trame*, le macrobloc supérieur représente les échantillons provenant de la région de la *sous-trame supérieure* de la *trame* qui se tient dans la région de l'espace de la *paire de macroblocs*. Pour une *paire de macroblocs de trame*, le macrobloc supérieur représente les échantillons de la *trame* qui est dans la moitié supérieure de la région de l'espace de la *paire de macroblocs*.

3.153 coefficient de transformée: grandeur scalaire, considérée comme faisant partie du domaine de la fréquence, qui est associée à un *indice de fréquence* unidimensionnel ou bidimensionnel particulier dans une partie de *transformation inverse* du *processus de décodage*.

3.154 niveau de coefficient de transformée: grandeur entière représentant la valeur associée à un indice de fréquence bidimensionnel particulier dans le *processus de décodage* avant la *normalisation* pour le calcul d'une valeur de *coefficient de transformée*.

3.155 identificateur unique universel (UUID): identificateur qui est unique par rapport à l'espace de tous les identificateurs uniques universels.

3.156 non spécifié: lorsqu'il est utilisé dans les paragraphes spécifiant des valeurs d'un *élément syntaxique* particulier, ce terme indique que ces valeurs n'ont pas de signification spécifiée dans la présente Recommandation | Norme internationale et n'auront pas de signification spécifiée à l'avenir en tant que partie intégrante de la présente Recommandation | Norme internationale.

3.157 codage à longueur variable (VLC, *variable length coding*): procédure réversible pour le codage entropique qui alloue de plus courtes chaînes binaires aux *symboles* dont on s'attend qu'ils seront plus fréquents et qui alloue des chaînes binaires plus longues aux *symboles* dont on s'attend qu'ils seront moins fréquents.

3.158 balayage en zigzag: ordre séquentiel spécifique des *niveaux de coefficient de transformée* depuis (approximativement) la plus basse fréquence spatiale jusqu'à la plus élevée. Le balayage en zigzag est utilisé pour les *niveaux de coefficient de transformée* dans les *macroblocs de trame*.

4 Abréviations

Pour les besoins de la présente Recommandation | Norme internationale, les abréviations suivantes s'appliquent.

CABAC codage arithmétique binaire adaptatif selon le contexte (*context-based adaptive binary arithmetic coding*)

CAVLC codage à longueur variable adaptatif selon le contexte (*context-based adaptive variable length coding*)

CBR débit binaire constant (*constant bit rate*)

CPB mémoire tampon d'images codées (*coded picture buffer*)

DPB mémoire tampon d'images décodées (*decoded picture buffer*)

DUT décodeur soumis aux essais (*decoder under test*)

FIFO premier entré, premier sorti (*first-in, first-out*)

HRD décodeur fictif de référence (*hypothetical reference decoder*)

HSS ordonnanceur fictif de flux binaires (*hypothetical stream scheduler*)

IDR rafraîchissement de décodage instantané (*instantaneous decoding refresh*)

LSB bit de plus faible poids (*least significant bit*)

MB macrobloc

MBAFF	codage de trame/sous-trame de macrobloc adaptatif (*macrobloch-adaptive frame-field coding*)
MSB	bit de plus fort poids (*most significant bit*)
NAL	couche d'abstraction du réseau (*network abstraction layer*)
RBSP	charge utile de séquence d'octets bruts (*raw byte sequence payload*)
SEI	informations d'amélioration supplémentaires (*supplemental enhancement information*)
SODB	chaîne de bits de données (*string of data bits*)
UUID	identificateur unique universel (*universal unique identifier*)
VBR	débit binaire variable (*variable bit rate*)
VCL	couche de codage vidéo (*video coding layer*)
VLC	codage à longueur variable (*variable length coding*)
VUI	informations de facilité d'utilisation de la vidéo (*video usability information*)

5 Conventions

NOTE – Les opérateurs mathématiques utilisés dans la présente Spécification sont similaires à ceux qui sont utilisés dans le langage de programmation C. Cependant, les opérations de division entière et de décalage arithmétique sont définies de façon spécifique. Les conventions de numérotage et de comptage commencent généralement à partir de 0.

5.1 Opérateurs arithmétiques

Les opérateurs arithmétiques ci-après sont définis comme suit.

+	Addition
–	Soustraction (dans le cas d'un opérateur binaire) ou négation (dans le cas d'un opérateur unaire de préfixe)
*	Multiplication
x^y	Exponentiation. Spécifie x à la puissance y. Dans d'autres contextes, une telle notation est utilisée afin de mettre un exposant qui n'est pas destiné à être interprété comme une puissance.
/	Division entière avec troncature du résultat tendant vers zéro. Par exemple, les valeurs 7/4 et –7/–4 sont tronquées à 1 et les valeurs –7/4 et 7/–4 sont tronquées à –1.
÷	Sert à marquer la division dans des équations mathématiques non destinées à faire l'objet d'une troncature ou d'un arrondi.
$\frac{x}{y}$	Sert à marquer la division dans des équations mathématiques non destinées à faire l'objet d'une troncature ou d'un arrondi.
$\sum_{i=x}^{y} f(i)$	Sommation des f(i) où i prend toutes les valeurs entières de x jusqu'à et y compris y.
x% y	Modulo. Reste de x divisé par y, défini seulement pour des entiers x et y avec x >= 0 et y > 0.

Lorsque l'ordre de préséance n'est pas indiqué explicitement par l'utilisation de parenthèses, on appliquera les règles suivantes:

– les opérations de multiplication et division sont censées prendre place avant l'addition et la soustraction;

– les opérations de multiplication et division en séquence sont effectuées à la suite de gauche à droite;

– les opérations d'addition et de soustraction en séquence sont effectuées à la suite de gauche à droite.

5.2 Opérateurs logiques

Les opérateurs logiques ci-après sont définis comme suit:

x && y	est le "ET" de la logique booléenne de x et y
x || y	est le "OU" de la logique booléenne de x ou y
!	est le "NON" de la logique booléenne

x ? y: z si x est "TRUE" (vrai) ou non égal à 0, il se rapporte à la valeur de y; sinon, il se rapporte à la valeur de z.

5.3 Opérateurs relationnels

Les opérateurs relationnels ci-après sont définis comme suit:

> supérieur à

>= supérieur ou égal à

< inférieur à

<= inférieur ou égal à

= = égal à

!= différent de

Lorsqu'un opérateur relationnel est appliqué à un élément syntaxique ou à une variable dont la valeur assignée est "na" (non applicable), cette valeur "na" est traitée comme une valeur distincte pour cet élément syntaxique ou cette variable. La valeur "na" est considérée comme inégale à toute autre valeur.

5.4 Opérateurs binaires

Les opérateurs binaires ci-après sont définis comme suit:

& binaire "et". Lorsqu'il agit sur des arguments entiers, il agit sur une représentation du complément à deux de la valeur entière. Lorsqu'il agit sur un argument binaire qui contient moins de bits qu'un autre argument, l'argument plus court est étendu en ajoutant plus de bits significatifs égaux à 0.

| binaire "ou". Lorsqu'il agit sur des arguments entiers, il agit sur une représentation du complément à deux de la valeur entière. Lorsqu'il agit sur un argument binaire qui contient moins de bits qu'un autre argument, l'argument plus court est étendu en ajoutant plus de bits significatifs égaux à 0.

x >> y décalage arithmétique à droite de la représentation entière du complément à deux de x par y chiffres binaires. Cette fonction n'est définie que pour des valeurs entières positives de y. Les bits décalés dans les MSB à la suite du décalage à droite ont une valeur égale au MSB de x avant l'intervention du décalage.

x << y décalage arithmétique à gauche de la représentation entière du complément à deux de x par y chiffres binaires. Cette fonction n'est définie que pour des valeurs entières positives de y. Les bits décalés dans les LSB à la suite du décalage à gauche doivent avoir une valeur égale à 0.

5.5 Opérateurs d'affectation

Les opérateurs arithmétiques ci-après sont définis comme suit:

= opérateur d'affectation.

++ Incrément, c'est-à-dire, x++ est équivalent à $x = x + 1$; lorsqu'il est utilisé dans un indice matriciel, il se rapporte à la valeur de la variable avant l'opération d'incrémentation.

–– Décrément, c'est-à-dire, x–– est équivalent à $x = x – 1$; lorsqu'il est utilisé dans un indice matriciel, il se rapporte à la valeur de la variable avant l'opération de décrémentation.

+= Incrément d'un montant spécifié, c'est-à-dire, x += 3 est équivalent à x = x + 3, et x += (-3) est équivalent à x = x + (-3).

–= Décrément d'un montant spécifié, c'est-à-dire, x –= 3 est équivalent à x = x – 3, et x –= (-3) est équivalent à x = x – (-3).

5.6 Notation de gamme

La notation suivante est utilisée afin de spécifier une gamme de valeurs:

x = y .. z x prend des valeurs entières de y à z inclus avec x, y et z entiers.

5.7 Fonctions mathématiques

Les fonctions mathématiques ci-après sont définies comme suit:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x >= 0 \\ -x & ; \quad x < 0 \end{cases} \tag{5-1}$$

Ceil(x) le plus petit entier supérieur ou égal à x. (5-2)

$$\text{Clip1}_Y(x) = \text{Clip3}(0, (1 << \text{BitDepth}_Y) - 1, x) \tag{5-3}$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 << \text{BitDepth}_C) - 1, x) \tag{5-4}$$

$$\text{Clips3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{autrement} \end{cases} \tag{5-5}$$

Floor(x) le plus grand entier inférieur ou égal à x. (5-6)

$$\text{InverseRasterScan}(a, b, c, d, e) = \begin{cases} (a\%(d/b)) * b; & e == 0 \\ (a/(d/b)) * c; & e == 1 \end{cases} \tag{5-7}$$

Log2(x) renvoie au logarithme à base 2 de x. (5-8)

Log10(x) renvoie au logarithme à base 10 de x. (5-9)

$$\text{Median}(x, y, z) = x + y + z - \text{Min}(x, \text{Min}(y, z)) - \text{Max}(x, \text{Max}(y, z)) \tag{5-10}$$

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases} \tag{5-11}$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases} \tag{5-12}$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0{,}5) \tag{5-13}$$

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x >= 0 \\ -1 & ; \quad x < 0 \end{cases} \tag{5-14}$$

$$\text{Sqrt}(x) = \sqrt{x} \tag{5-15}$$

5.8 Variables, éléments syntaxiques et tableaux

Les éléments syntaxiques contenus dans le flux binaire sont représentés en caractères **gras**. Chaque élément syntaxique est décrit par son nom (toujours en minuscules avec des liaisons en caractères de soulignement), par ses catégories syntaxiques (au nombre de 1 ou de 2) et par un ou deux descripteurs pour sa méthode de représentation codée. Le processus de décodage se comporte conformément à la valeur de l'élément syntaxique et aux valeurs des éléments syntaxiques précédemment décodés. Lorsqu'une valeur d'un élément syntaxique est utilisée dans les tableaux syntaxiques ou dans le texte, cette valeur apparaît en caractères normaux (c'est-à-dire, pas en gras).

Dans certains cas, les tableaux syntaxiques peuvent utiliser les valeurs d'autres variables déduites de valeurs d'éléments syntaxiques. De telles variables apparaissent dans les tableaux syntaxiques, ou dans le texte, et sont désignées par un mélange de majuscules et de minuscules sans aucun caractère de soulignement. Les variables commençant par une majuscule sont tirées, pour le décodage, de la structure syntaxique courante et de toutes celles qui en dépendent. Les variables commençant par une majuscule peuvent être utilisées dans le processus de décodage pour des structures syntaxiques ultérieures en mentionnant la structure syntaxique d'origine de la variable. Les variables commençant par des minuscules ne sont utilisées qu'à l'intérieur du paragraphe dont elles sont tirées.

Dans certains cas, on utilise de façon interchangeable avec leurs valeurs numériques des noms "mnémoniques" pour des valeurs d'élément syntaxique ou de variable. Parfois, les noms "mnémoniques" sont utilisés sans être associés à une quelconque valeur numérique. L'association entre valeurs et noms est spécifiée dans le texte. Les noms sont construits à partir d'un ou de plusieurs groupes de lettres séparées par un caractère de soulignement. Chaque groupe commence par une majuscule et peut contenir plusieurs majuscules.

NOTE – La syntaxe est décrite d'une façon qui suit de près la construction syntaxique du langage C.

Les fonctions sont décrites par leur nom, qui est construit comme un nom d'élément syntaxique, avec des parenthèses à droite et à gauche incluant zéro, un ou plusieurs noms de variables (pour les définitions) ou de valeurs (pour l'utilisation), séparés par des virgules (s'il y a plus d'une seule variable).

Une série matricielle unidimensionnelle est considérée comme étant une liste. Une série matricielle bidimensionnelle est considérée comme étant une matrice.

Les séries matricielles peuvent être soit des éléments syntaxiques soit des variables. Des indices inférieurs ou des crochets sont utilisés afin d'indexer des séries matricielles. Dans le cas d'une description visuelle d'une matrice, le premier indice inférieur est utilisé comme indice (vertical) de rangée et le deuxième indice inférieur est utilisé comme indice (horizontal) de colonne. L'ordre d'indexation est inversé lors de l'utilisation de crochets au lieu d'indices inférieurs pour l'indexation. Un élément d'une matrice s situé à la position horizontale x et à la position verticale y pourra donc être désigné soit par s[x,y] ou par s_{yx}.

La notation binaire est indiquée par l'inclusion de la chaîne de valeurs binaires entre des apostrophes simples. Par exemple, '01000001' représente une chaîne de 8 bit ayant seulement ses second et dernier bits égaux à 1.

La notation hexadécimale, indiquée par le préfixe "0x" devant le nombre hexadécimal, peut être utilisée à la place de la notation binaire lorsque le nombre de bits est un multiple entier de 4. Par exemple, 0x41 représente une chaîne binaire de 8 bit ayant seulement ses second et dernier bits égaux à 1.

Les valeurs numériques non incluses entre des apostrophes simples et sans le préfixe "0x" sont des valeurs décimales.

Une valeur égale à 0 représente une condition "FALSE" dans une instruction d'essai. La valeur "TRUE" est représentée par toute autre valeur que zéro.

5.9 Description textuelle des opérations logiques

Dans le texte, une instruction d'opérations logiques qui serait décrite en pseudo-code comme:

```
si( condition 0 )
    instruction 0
    sinon si ( condition 1 )
      instruction 1
    …
    sinon /* remarque informative sur la condition restante */
      instruction n
```

peut être décrite de la manière suivante:

... comme suit / ... on applique ce qui suit:

– si condition 0, instruction 0

– Sinon, si condition 1, instruction 1
– …
– Sinon (si remarque informative sur la condition restante), instruction n.

Chaque instruction "si...sinon, si...sinon, ..." dans le texte est introduite par "... comme suit" ou "... on applique ce qui suit" immédiatement suivi de "si ... ". La dernière condition du "si...sinon, si...sinon, ..." est toujours un "sinon, ...". L'entrelacement des instructions "si...sinon, si...sinon, ..." peut se reconnaître en recherchant les "... comme suit" ou "... on applique ce qui suit" avec la terminaison "sinon, ...".

Dans le texte, une instruction d'opérations logiques qui serait décrite en pseudo-code comme:

```
si( condition 0a  &&  condition 0b )
    instruction 0
  sinon si ( condition 1a  ||  condition 1b )
    instruction 1
  …
  sinon
    instruction n
```

peut être décrite de la façon suivante:

... comme suit / ... on applique ce qui suit.

– Si toutes les conditions suivantes sont vraies, instruction 0
  – condition 0a
  – condition 0b
– Sinon, si une des conditions suivantes est vraie, instruction 1
  – condition 1a
  – condition 1b
– …
– Sinon, instruction n

Dans le texte, une instruction d'opérations logiques qui serait décrite en pseudo-code comme

```
si( condition 0 )
    instruction 0
  si( condition 1 )
    instruction 1
```

peut être décrite de la façon suivante:

Lorsqu'il y a la condition 0, instruction 0

Lorsqu'il y a la condition 1, instruction 1

5.10 Processus

Les processus sont utilisés afin de décrire le décodage des éléments syntaxiques. La spécification et l'invocation d'un processus sont séparées. Tous les éléments syntaxiques et les variables en majuscules qui appartiennent à la structure syntaxique courante et aux structures syntaxiques qui en dépendent sont disponibles dans la spécification et l'invocation d'un processus. La spécification d'un processus peut aussi avoir une variable en minuscule explicitement spécifiée comme entrée. Chaque spécification de processus spécifie explicitement une sortie. La sortie est une variable qui peut être aussi bien en majuscules qu'en minuscules.

L'allocation des variables est spécifiée comme suit:

– si on invoque un processus, les variables sont explicitement affectées à des lettres minuscules représentant des variables d'entrée ou de sortie de la spécification du processus au cas où elles n'auraient pas le même nom;
– sinon (lorsque les variables ont le même nom à l'invocation et à la spécification), l'affectation est implicite.

Dans la spécification d'un processus, le nom de la variable peut faire référence à un macrobloc spécifique en ayant une valeur égale à l'adresse de ce macrobloc spécifique.

6 Format des données de source, des données codées, des données décodées et des données de sortie, processus de balayage et relations de voisinage

6.1 Formats de flux binaires

Le présent paragraphe spécifie la relation entre le flux d'unités (de couche) NAL et le flux d'octets, l'un ou l'autre étant appelé *flux binaire*.

Le flux binaire peut avoir un des deux formats suivants: le format de flux d'unités NAL ou le format de flux d'octets. Le format de flux d'unités NAL est théoriquement de type plus "basique". Il consiste en une séquence de structures syntaxiques appelées *unités NAL*. Cette séquence est ordonnée en ordre de décodage. Il y a des contraintes imposées à l'ordre de décodage (et au contenu) des unités NAL dans le flux d'unités NAL.

Le format de flux d'octets peut être construit à partir du format de flux d'unités NAL par ordonnancement des unités NAL dans l'ordre de décodage et par adjonction à chaque unité NAL d'un préfixe de code de déclenchement et de zéro, un ou plusieurs octets de valeur zéro afin de former un flux d'octets. Le format de flux d'unités NAL peut être extrait du format de flux d'octets par recherche de la localisation du schéma unique de préfixe de code de déclenchement au sein de ce flux d'octets. Les méthodes de tramage des unités NAL d'une autre façon que par l'utilisation du format de flux d'octet sont en dehors du domaine d'application de la présente Recommandation | Norme internationale. Le format de flux d'octets est spécifié à l'Annexe B.

6.2 Formats d'image source, d'image décodée et d'image de sortie

Le présent paragraphe spécifie la relation entre trames et sous-trames de source et décodées. Cette relation est donnée via le flux binaire.

La source vidéo qui est représentée par le flux binaire est une séquence de trames et/ou de sous-trames (appelées collectivement *images*) en ordre de décodage.

Les images source et décodées (trames ou sous-trames) sont chacune composées d'une ou de plusieurs série(s) matricielle(s) d'échantillons:

- échantillons luma (Y) (monochromes), avec ou sans table auxiliaire;
- échantillons luma et deux échantillons chroma (YCbCr ou YCgCo), avec ou sans table auxiliaire;
- échantillons verts, bleus et rouges (composantes VBR, également désignées par *RGB*), avec ou sans table auxiliaire;
- séries matricielles représentant d'autres échantillonnages monochromes ou trichromatiques non spécifiés (par exemple composantes YZX, également désignées par *XYZ*), avec ou sans table auxiliaire.

Afin de faciliter la notation et la terminologie dans la présente Spécification, les variables et les termes associés à ces séries matricielles sont désignés par luma (ou L ou Y) et par chroma, les deux séries matricielles d'échantillons chromatiques étant désignées par les abréviations Cb et Cr, quelle que soit la méthode de représentation des couleurs qui a été choisie. Cette méthode peut être indiquée dans la syntaxe spécifiée par l'Annexe E. Les tables auxiliaires d'échantillons monochromes, qui peuvent, le cas échéant, être présentes sous forme d'images auxiliaires dans une séquence vidéo codée, sont facultatives au décodage et peuvent servir à des fins telles que la fusion alpha (superposition de textures).

Les variables SubWidthC et SubHeightC sont spécifiées dans le Tableau 6-1, selon le mode d'échantillonnage du format chromatique, lequel est spécifié au moyen de la variable de format chromatique chroma_format_idc. Une entrée marquée par un tiret ("-") dans le Tableau 6-1 désigne une valeur indéfinie des variables SubWidthC et SubHeightC. D'autres valeurs pourront être spécifiées ultérieurement par l'UIT-T | ISO/CEI pour chroma_format_idc, pour SubWidthC et pour SubHeightC.

Tableau 6-1 – Valeurs de SubWidthC et de SubHeightC déduites de chroma_format_idc

chroma_format_idc	Format chromatique	SubWidthC	SubHeightC
0	monochrome	–	–
1	4:2:0	2	2
2	4:2:2	2	1
3	4:4:4	1	1

En échantillonnage monochrome, il n'y a qu'une seule série matricielle d'échantillons, qui est considérée par défaut comme étant la série luma.

En échantillonnage 4:2:0, chacune des deux séries matricielles chromatiques contient la moitité de la hauteur et la moitié de la largeur de la série luma.

En échantillonnage 4:2:2, chacune des deux séries matricielles chromatiques a la même hauteur et la moitié de la largeur de la série luma.

En échantillonnage 4:4:4, chacune des deux séries matricielles chromatiques a la même hauteur et la même largeur que la série luma.

La largeur et la hauteur des séries matricielles d'échantillons luma sont chacune un multiple entier de 16. Dans les flux binaires utilisant l'échantillonnage au format chromatique 4:2:0, la largeur et la hauteur des séries matricielles d'échantillons chroma sont chacune un multiple entier de 8. Dans les flux binaires utilisant l'échantillonnage 4:2:2, la largeur des séries matricielles d'échantillons chroma est un multiple entier de 8 et leur hauteur est un multiple entier de 16. La hauteur d'une série matricielle luma qui est codée sous forme de deux sous-trames distinctes ou qui est codée par trame/sous-trame de macrobloc adaptatif (voir ci-dessous) est un multiple entier de 32. Dans les flux binaires utilisant l'échantillonnage 4:2:0, la hauteur de chaque série matricielle chroma qui est codée sous forme de deux sous-trames distinctes ou qui est codée par trame/sous-trame de macrobloc adaptatif (voir ci-dessous) est un multiple entier de 16. La largeur ou la hauteur des images issues du processus de décodage n'ont pas besoin d'être un multiple de 16 et peuvent être spécifiées au moyen d'une mire de recadrage.

La syntaxe des séries matricielles d'échantillons luma et (lorsqu'ils sont présents) d'échantillons chroma est ordonnancée de telle sorte que, lorsque les données pour les trois composantes chromatiques sont présentes, les données relatives à la série d'échantillons luma sont les premières, suivies des données éventuellement présentes concernant la série d'échantillons Cb, suivies des données éventuellement présentes concernant la série d'échantillons Cr, sauf spécification contraire.

La largeur des sous-trames codées se rapportant à un ensemble de paramètres de séquence spécifiques est la même que celle des trames codées se rapportant au même ensemble de paramètres de séquence (voir ci-dessous). La hauteur des sous-trames codées se rapportant à un ensemble de paramètres de séquence spécifiques est la moitié de celle des trames codées se rapportant au même ensemble de paramètres de séquence (voir ci-dessous).

Le nombre de bits nécessaires pour la représentation de chacun des échantillons contenus dans les séries matricielles luma et chroma d'une séquence vidéo est compris entre 8 et 12. Le nombre de bits utilisés dans la série matricielle luma peut différer du nombre de bits utilisés dans les séries matricielles chroma.

Lorsque la valeur de la variable chroma_format_idc est égale à 1, les localisations verticales et horizontales relatives nominales des échantillons luma et chroma dans les trames sont indiquées à la Figure 6-1. D'autres localisations relatives d'échantillons chroma peuvent être indiquées dans les informations de facilité d'utilisation de la vidéo (voir l'Annexe E).

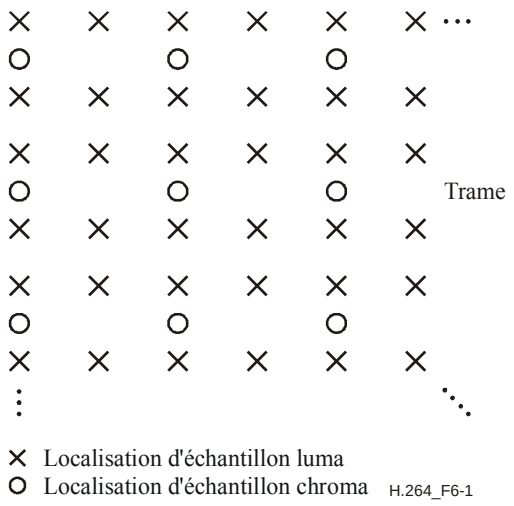


Figure 6-1 – Localisations nominales verticales et horizontales des échantillons 4:2:0 luma et chroma dans une trame

Une trame se compose de deux sous-trames comme décrit ci-dessous. Une image codée peut représenter une trame codée ou une sous-trame codée individuelle. Une séquence vidéo codée se conformant à la présente Recommandation | Norme internationale peut contenir des combinaisons arbitraires de trames codées et de sous-trames codées. Le processus de décodage est aussi spécifié d'une façon qui permet que de très petites régions d'une trame codée soient codées comme des régions de trame ou de sous-trame, au moyen du codage de trame/sous-trame par macrobloc adaptatif.

Les sous-trames de source et décodées sont d'un des deux types suivants: sous-trame supérieure ou sous-trame inférieure. Lorsqu'il y a deux sous-trames en sortie au même moment, ou qu'elles sont combinées de façon à être utilisées comme trame de référence (voir ci-dessous), les deux sous-trames (qui doivent être de parité opposée) sont entrelacées. La première (c'est-à-dire, celle du haut), la troisième, la cinquième, etc., rangée d'une trame décodée sont les rangées de sous-trame supérieure. La seconde, la quatrième, la sixième, etc., rangée d'une trame décodée sont les rangées de sous-trame inférieure. Une sous-trame supérieure ne se compose que des rangées de sous-trame supérieure d'une trame décodée, et une sous-trame inférieure ne se compose que des rangées de sous-trame inférieure d'une trame. Lorsque la sous-trame supérieure ou la sous-trame inférieure d'une trame décodée est utilisée comme sous-trame de référence (voir ci-dessous), seules sont utilisées les rangées paires (pour une sous-trame supérieure) ou les rangées impaires (pour une sous-trame inférieure) de la trame décodée.

Lorsque la valeur de la variable chroma_format_idc est égale à 1, les localisations relatives nominales verticales et horizontales des échantillons luma et chroma contenus dans les sous-trames inférieures et supérieures sont indiquées dans la Figure 6-2. Les localisations relatives nominales verticales des échantillons contenus dans une sous-trame supérieure sont spécifiées comme étant décalées vers le haut d'un quart de la hauteur d'un échantillon luma par rapport à la grille d'échantillonnage de la sous-trame. Les localisations d'échantillonnage verticales des échantillons chroma contenus dans une sous-trame inférieure sont spécifiées comme étant décalées vers le bas d'un quart de la hauteur d'un échantillon luma par rapport à la grille d'échantillonnage de la sous-trame. D'autres localisations relatives d'échantillon chroma peuvent être indiquées dans les informations de facilité d'utilisation de la vidéo (voir l'Annexe E).

NOTE – Le décalage des échantillons chroma est destiné à aligner verticalement ces échantillons sur la localisation habituelle par rapport à la grille d'échantillonnage de trame entière, comme indiqué dans la Figure 6-1.

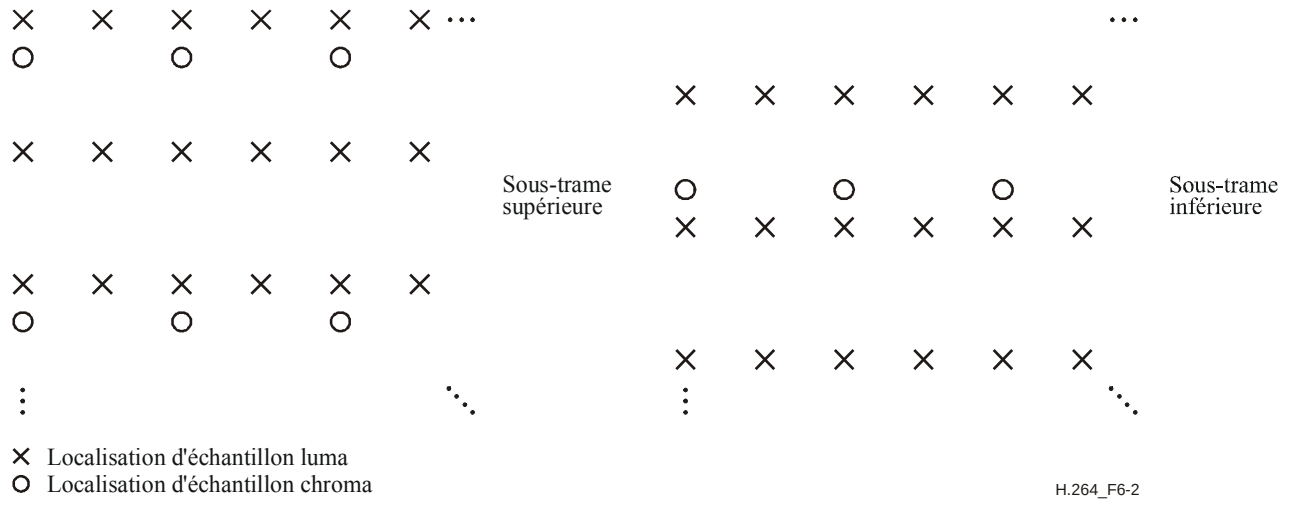


Figure 6-2 – Localisations nominales verticales et horizontales des échantillons 4:2:0 dans les sous-trames supérieure et inférieure

Lorsque la valeur de la variable chroma_format_idc est égale à 2, les échantillons chroma sont copositionnés avec les échantillons luma correspondants et les localisations nominales dans une trame et dans les sous-trames sont respectivement représentées dans la Figure 6-3 – Localisations nominales verticales et horizontales dese échantillons 4:2:0 luma et chroma dans une trame et dans la Figure 6-4 – Localisations nominales verticales et horizontales des échantillons 4:2:2 luma et chroma dans une trame.

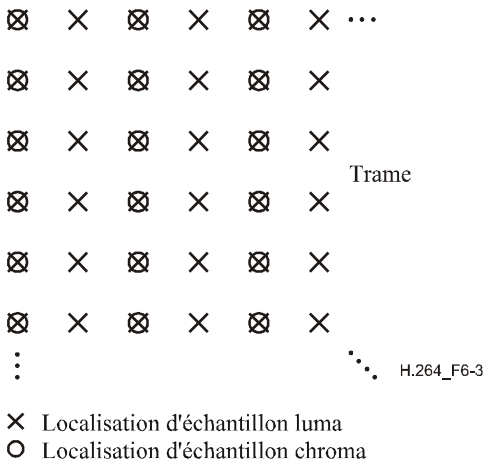


Figure 6-3 – Localisations nominales verticales et horizontales des échantillons 4:2:2 luma et chroma dans une trame

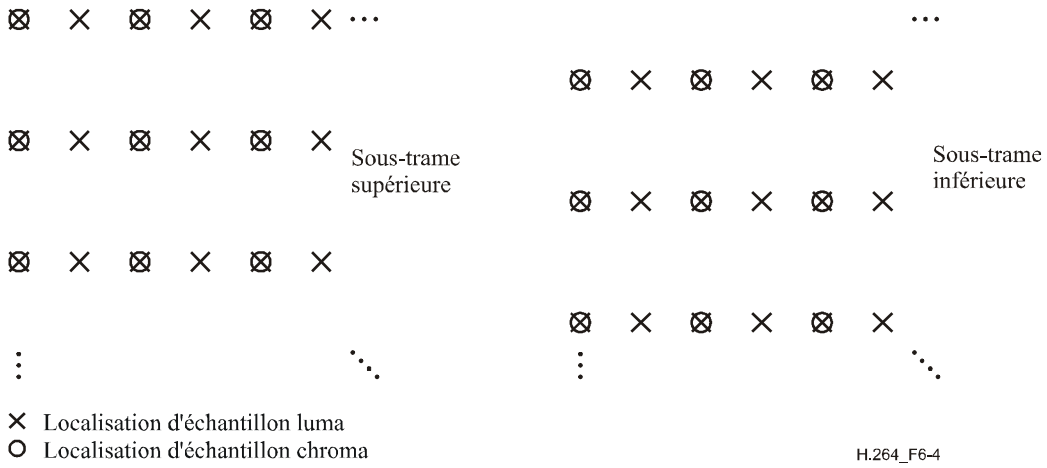


Figure 6-4 – Localisations nominales verticales et horizontales des échantillons 4:2:2 dans les sous-trames supérieure et inférieure

Lorsque la valeur de la variable chroma_format_idc est égale à 3, tous les échantillons des séries matricielles sont copositionnés avec toutes les occurrences de trames et de sous-trames. Les localisations nominales dans une trame et dans les sous-trames sont respectivement représentées dans les Figures 6-5 et 6-6.

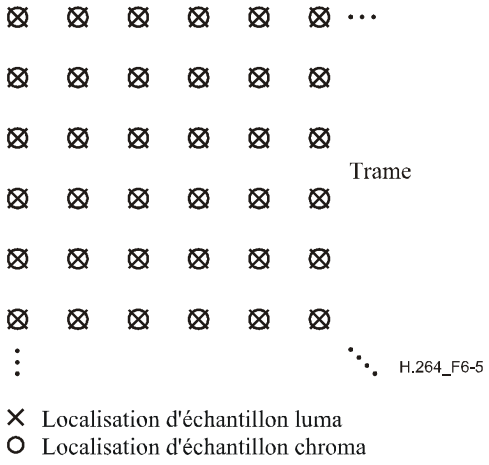


Figure 6-5 – Localisations nominales verticales et horizontales des échantillons 4:4:4 luma et chroma dans une trame

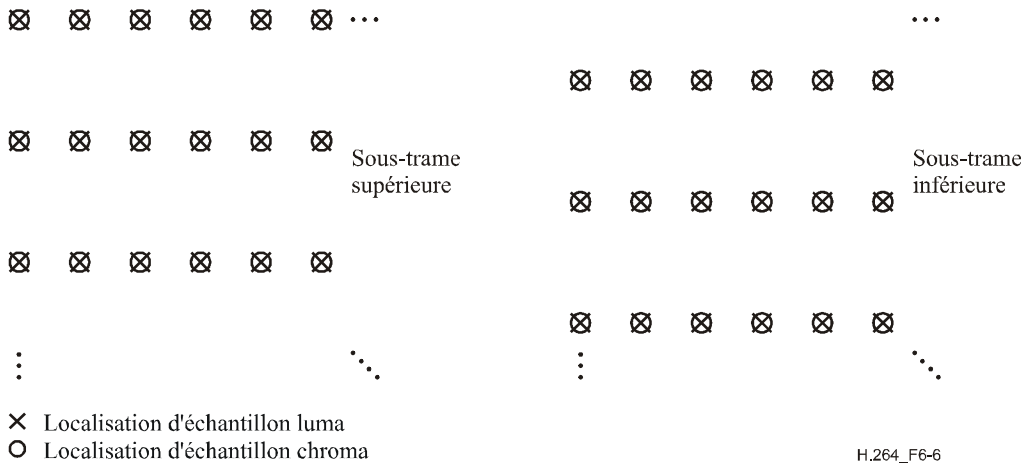


Figure 6-6 – Localisations nominales verticales et horizontales des échantillons 4:4:4 dans les sous-trames supérieure et inférieure

Les échantillons sont traités en unités de macrobloc. La série matricielle d'échantillons luma pour chaque macrobloc compte 16 échantillons en largeur comme en hauteur. Les variables MbWidthC et MbHeightC, qui spécifient respectivement la largeur et la hauteur des séries matricielles d'échantillons chroma pour chaque macrobloc, sont calculées comme suit:

– si la variable chroma_format_idc est égale à 0 (format monochrome), les variables MbWidthC et MbHeightC sont toutes les deux égales à 0 (car aucune série matricielle d'échantillons chroma n'est spécifiée pour la vidéo monochrome);

– sinon, les variables MbWidthC et MbHeightC sont calculées comme suit:

$$MbWidthC = 16 / SubWidthC \tag{6-1}$$

$$MbHeightC = 16 / SubHeightC \tag{6-2}$$

6.3 Subdivision spatiale des images et des tranches

Le présent paragraphe spécifie la façon dont une image est divisée en tranches et macroblocs. Les images sont divisées en tranches. Une tranche est une séquence de macroblocs ou, lorsque le décodage de trame/sous-trame de macrobloc adaptatif est utilisé, une tranche est une séquence de paires de macroblocs.

Chaque macrobloc est composé d'une série matricielle d'échantillons luma 16x16 et de deux séries matricielles d'échantillons chroma 8x8. Lorsque le décodage de trame/sous-trame de macrobloc adaptatif n'est pas utilisé, chaque macrobloc représente une région spatiale rectangulaire de l'image. Par exemple, une image peut être divisée en deux tranches, comme indiqué à la Figure 6-7.

Figure 6-7 – Image avec 11 sur 9 macroblocs qui est subdivisée en deux tranches

Lorsque le décodage de trame/sous-trame de macrobloc adaptatif est utilisé, l'image est subdivisée en tranches contenant un nombre entier de paires de macroblocs, comme indiqué à la Figure 6-8. Chaque paire de macroblocs comporte deux macroblocs.

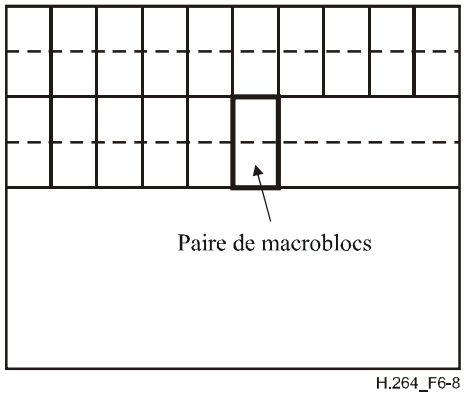


Figure 6-8 – Subdivision de la trame décodée en paires de macroblocs

6.4 Processus de balayage inverse et processus de déduction pour le voisinage

Le présent paragraphe spécifie les processus de balayage inverse; c'est-à-dire le mappage des indices sur les localisations, et les processus de déduction pour le voisinage.

6.4.1 Processus de balayage inverse de macrobloc

L'entrée dans ce processus est une adresse de macrobloc mbAddr.

La sortie de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour le macrobloc ayant l'adresse mbAddr relative à l'échantillon supérieur gauche de l'image.

Le processus de balayage inverse de macrobloc est spécifié comme suit:

– Si MbaffFrameFlag est égal à 0,

$$x = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 0) \quad (6\text{-}3)$$

$$y = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 1) \quad (6\text{-}4)$$

– Sinon (si MbaffFrameFlag est égal à 1), on applique ce qui suit:

$$xO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 0) \quad (6\text{-}5)$$

$$yO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 1) \quad (6\text{-}6)$$

Selon le macrobloc en cours, on applique ce qui suit.

– Si le macrobloc en cours est un macrobloc de trame

$$x = xO \quad (6\text{-}7)$$

$$y = yO + (\text{mbAddr} \% 2) * 16 \quad (6\text{-}8)$$

– Sinon (si le macrobloc en cours est un macrobloc de sous-trame)

$$x = xO \quad (6\text{-}9)$$

$$y = yO + (\text{mbAddr} \% 2) \quad (6\text{-}10)$$

6.4.2 Processus de balayage inverse de subdivision de macrobloc et de sous-macrobloc

Les macroblocs ou sous-macroblocs peuvent être divisés, et les subdivisions sont balayées pour l'interprédiction comme indiqué à la Figure 6-9. Les rectangles extérieurs se rapportent, selon le cas, aux échantillons contenus dans un macrobloc ou dans un sous-macrobloc. Les rectangles se rapportent aux subdivisions. Le nombre indiqué dans chaque rectangle spécifie l'indice du balayage inverse de division de macrobloc ou du balayage inverse de division de sous-macrobloc.

Les fonctions MbPartWidth( ), MbPartHeight( ), SubMbPartWidth( ) et SubMbPartHeight( ) décrivant la largeur et la hauteur des divisions de macrobloc et des divisions de sous-macrobloc sont spécifiées dans les Tableaux 7-13, 7-14, 7-17 et 7-18. Les fonctions MbPartWidth( ) et MbPartHeight( ) sont mises aux valeurs appropriées pour chaque macrobloc, en fonction du type de macrobloc. Les fonctions SubMbPartWidth( ) et SubMbPartHeight( ) sont mises aux valeurs appropriées pour chaque sous-macrobloc d'un macrobloc avec la variable mb_type égale à P_8x8, P_8x8ref0, ou B_8x8, selon le type du sous-macrobloc.

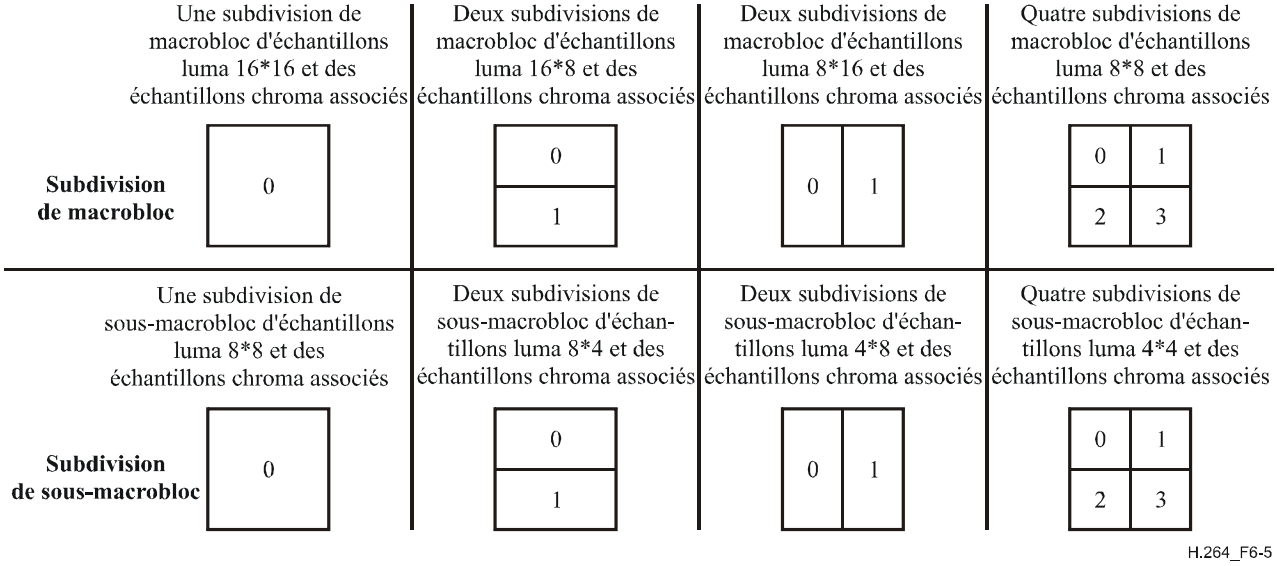


Figure 6-9 – Divisions de macrobloc, divisions de sous-macrobloc, balayages de division de macrobloc et balayages de division de sous-macrobloc

6.4.2.1 Processus de balayage inverse de division de macrobloc

L'entrée dans ce processus est l'indice mbPartIdx d'une division d'un macrobloc.

Le résultat de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour la division de macrobloc mbPartIdx se rapportant à l'échantillon supérieur gauche du macrobloc.

Le processus de balayage inverse de la division de macrobloc est spécifié par:

$$x = \text{InverseRasterScan}(\text{mbPartIdx}, \text{MbPartWidth}(\text{mb\_type}), \text{MbPartHeight}(\text{mb\_type}), 16, 0) \quad (6\text{-}11)$$

$$y = \text{InverseRasterScan}(\text{mbPartIdx}, \text{MbPartWidth}(\text{mb\_type}), \text{MbPartHeight}(\text{mb\_type}), 16, 1) \quad (6\text{-}12)$$

6.4.2.2 Processus de balayage inverse de division de sous-macrobloc

Les entrées dans ce processus sont l'indice mbPartIdx d'une division de macrobloc et l'indice subMbPartIdx d'une division de sous-macrobloc.

Le résultat de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour la division de sous-macrobloc subMbPartIdx relative à l'échantillon supérieur gauche du sous-macrobloc.

Le processus de balayage de division de sous-macrobloc inverse est spécifié comme suit.

– Si mb_type est égal à P_8x8, P_8x8ref0, ou B_8x8,

$$x = \text{InverseRasterScan}(\text{subMbPartIdx}, \text{SubMbPartWidth}(\text{sub\_mb\_type}[\text{mbPartIdx}]), \text{SubMbPartHeight}(\text{sub\_mb\_type}[\text{mbPartIdx}]), 8, 0) \quad (6\text{-}13)$$

y = InverseRasterScan(subMbPartIdx, SubMbPartWidth(sub_mb_type[mbPartIdx]),
SubMbPartHeight(sub_mb_type[mbPartIdx]), 8, 1) (6-14)

– Sinon,

x = InverseRasterScan(subMbPartIdx, 4, 4, 8, 0) (6-15)

y = InverseRasterScan(subMbPartIdx, 4, 4, 8, 1) (6-16)

6.4.3 Processus de balayage inverse de bloc luma 4x4

L'entrée dans ce processus est l'indice luma4x4BlkIdx d'un bloc luma 4x4.

Le résultat de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour le bloc luma 4x4 avec l'indice luma4x4BlkIdx relatif à l'échantillon luma supérieur gauche du macrobloc.

La Figure 6-10 montre le balayage pour les blocs luma 4x4.

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Figure 6-10 – Balayage pour les blocs luma 4x4

Le processus de balayage inverse de bloc luma 4x4 est spécifié par

x = InverseRasterScan(luma4x4BlkIdx / 4, 8, 8, 16, 0) + InverseRasterScan(luma4x4BlkIdx % 4, 4, 4, 8, 0) (6-17)

y = InverseRasterScan(luma4x4BlkIdx / 4, 8, 8, 16, 1) + InverseRasterScan(luma4x4BlkIdx % 4, 4, 4, 8, 1) (6-18)

6.4.4 Processus de balayage inverse de bloc luma 8x8

L'entrée dans ce processus est l'indice luma8x8BlkIdx de bloc lum 8x8.

Le résultat de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour le bloc luma 8x8 avec l'indice luma8x8BlkIdx relatif à l'échantillon luma supérieur gauche du macrobloc.

La Figure 6-11 montre le balayage pour les blocs luma 8x8.

0	1
2	3

Figure 6-11 – Balayage pour les blocs luma 8x8

Le processus de balayage inverse de bloc luma 8x8 est spécifié par

x = InverseRasterScan(luma8x8BlkIdx / 4, 8, 8, 16, 0) (6-19)

y = InverseRasterScan(luma8x8BlkIdx / 4, 8, 8, 16, 1) (6-20)

6.4.5 Processus de déduction de la disponibilité pour les adresses de macrobloc

L'entrée dans ce processus est une adresse de macrobloc mbAddr.

Le résultat de ce processus est la disponibilité du macrobloc mbAddr.

NOTE – La signification de la disponibilité est déterminée lorsque ce processus est invoqué.

Le macrobloc est marqué comme disponible, à moins qu'une des conditions suivantes ne soit vraie, auquel cas le macrobloc est marqué comme indisponible:

– mbAddr < 0

– mbAddr > CurrMbAddr

– le macrobloc avec l'adresse mbAddr appartient à une tranche différente du macrobloc ayant l'adresse CurrMbAddr.

6.4.6 Processus de déduction pour les adresses de macrobloc du voisinage et leur disponibilité

Ce processus ne peut être invoqué que lorsque MbaffFrameFlag est égal à 0.

Les résultats de ce processus sont:

– mbAddrA: l'adresse et l'état de disponibilité du macrobloc à gauche du macrobloc en cours;

– mbAddrB: l'adresse et l'état de disponibilité du macrobloc au-dessus du macrobloc en cours;

– mbAddrC: l'adresse et l'état de disponibilité du macrobloc droit au-dessus du macrobloc en cours;

– mbAddrD: l'adresse et l'état de disponibilité du macrobloc gauche au-dessus du macrobloc en cours.

La Figure 6-12 montre les localisations spatiales relatives des macroblocs avec mbAddrA, mbAddrB, mbAddrC et mbAddrD par rapport au macrobloc en cours avec CurrMbAddr.

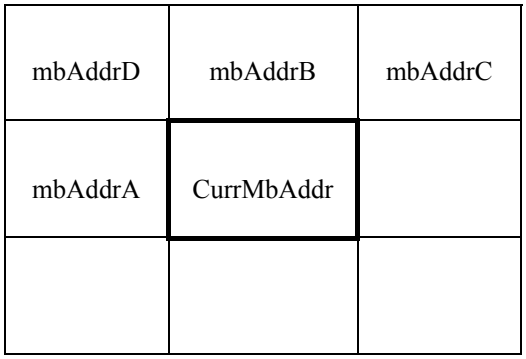


Figure 6-12 – Macroblocs voisins pour un macrobloc donné

L'entrée du processus au § 6.4.5 est mbAddrA = CurrMbAddr – 1 et le résultat dépend de la disponibilité du macrobloc mbAddrA. De plus, mbAddrA est marqué comme non disponible lorsque CurrMbAddr % PicWidthInMbs est égal à 0.

L'entrée du processus au § 6.4.5 est mbAddrB = CurrMbAddr – PicWidthInMbs et le résultat dépend de la disponibilité du macrobloc mbAddrB.

L'entrée du processus dans le § 6.4.5 est mbAddrC = CurrMbAddr – PicWidthInMbs + 1 et le résultat dépend de la disponibilité du macrobloc mbAddrC. De plus, mbAddrC est marqué comme non disponible lorsque (CurrMbAddr + 1) % PicWidthInMbs est égal à 0.

L'entrée du processus au § 6.4.5 est mbAddrD = CurrMbAddr – PicWidthInMbs – 1 et le résultat dépend de la disponibilité du macrobloc mbAddrD. De plus, mbAddrD est marqué comme non disponible lorsque CurrMbAddr % PicWidthInMbs est égal à 0.

6.4.7 Processus de déduction des adresses de macrobloc du voisinage et de leur disponibilité dans les trames MBAFF

Ce processus ne peut être invoqué que lorsque MbaffFrameFlag est égal à 1.

Les résultats de ce processus sont:

– mbAddrA: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs à gauche de la paire de macroblocs en cours;

– mbAddrB: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs au-dessus de la paire de macroblocs en cours;

– mbAddrC: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs à droite au-dessus de la paire de macroblocs en cours;

– mbAddrD: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs à gauche au-dessus de la paire de macroblocs en cours.

La Figure 6-13 montre les localisations spatiales relatives des macroblocs avec mbAddrA, mbAddrB, mbAddrC et mbAddrD par rapport au macrobloc en cours avec CurrMbAddr.

mbAddrA, mbAddrB, mbAddrC et mbAddrD ont des valeurs identiques sans considération de savoir si le macrobloc en cours est le macrobloc supérieur ou inférieur d'une paire de macroblocs.

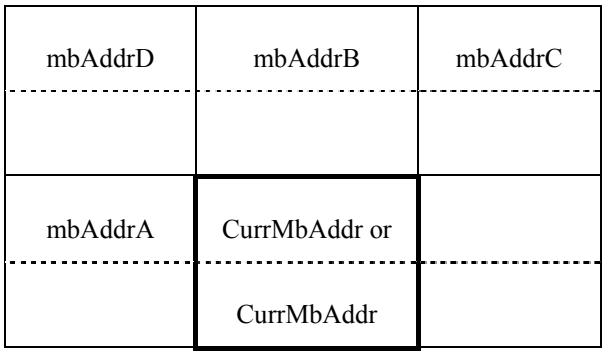


Figure 6-13 – Macroblocs du voisinage pour un macrobloc donné dans des trames MBAFF

L'entrée du processus au § 6.4.5 est mbAddrA =2 * (CurrMbAddr / 2 – 1) et le résultat dépend de la disponibilité du macrobloc mbAddrA. De plus, mbAddrA est marqué comme non disponible lorsque (CurrMbAddr / 2) % PicWidthInMbs est égal à 0.

L'entrée du processus au § 6.4.5 est mbAddrB =2 * (CurrMbAddr / 2 – PicWidthInMbs) et le résultat dépend de la disponibilité du macrobloc mbAddrB.

L'entrée du processus au § 6.4.5 est mbAddrC = 2 * (CurrMbAddr / 2 – PicWidthInMbs + 1) et le résultat dépend de la disponibilité du macrobloc mbAddrC. De plus, mbAddrC est marqué comme non disponible lorsque (CurrMbAddr / 2 + 1) % PicWidthInMbs est égal à 0.

L'entrée du processus au § 6.4.5 est mbAddrD = 2 * (CurrMbAddr / 2 – PicWidthInMbs – 1) et le résultat dépend de la disponibilité du macrobloc mbAddrD. De plus, mbAddrD est marqué comme non disponible lorsque (CurrMbAddr / 2) % PicWidthInMbs est égal à 0.

6.4.8 Processus de déduction pour les macroblocs, blocs et subdivisions du voisinage

Le paragraphe 6.4.8.1 spécifie le processus de déduction pour les macroblocs du voisinage.

Le paragraphe 6.4.8.2 spécifie le processus de déduction pour les blocs luma 8x8 du voisinage.

Le paragraphe 6.4.8.3 spécifie le processus de déduction pour les blocs luma 4x4 du voisinage.

Le paragraphe 6.4.8.4 spécifie le processus de déduction pour les blocs chroma 4x4 du voisinage.

Le paragraphe 6.4.8.5 spécifie le processus de déduction pour les subdivisions du voisinage.

Le Tableau 6-2 spécifie les valeurs pour la différence de localisation luma (xD, yD) pour l'entrée et le remplacement de N dans mbAddrN, mbPartIdxN, subMbPartIdxN, luma8x8BlkIdxN, luma4x4BlkIdxN et chroma4x4BlkIdxN pour la sortie. Ces allocations d'entrée et de sortie sont utilisées aux § 6.4.8.1 à 6.4.8.5. La variable predPartWidth est spécifiée lorsqu'on se rapporte au Tableau 6-2.

Tableau 6-2 – Spécification des allocations d'entrée et de sortie pour les § 6.4.8.1 à 6.4.8.5

N	xD	yD
A	–1	0
B	0	–1
C	predPartWidth	–1
D	–1	–1

La Figure 6-14 illustre la localisation relative des macroblocs, blocs ou subdivisions A, B, C et D du voisinage par rapport au macrobloc, subdivision ou bloc en cours lorsque le macrobloc, subdivision, ou bloc en cours est en mode de codage de trame.

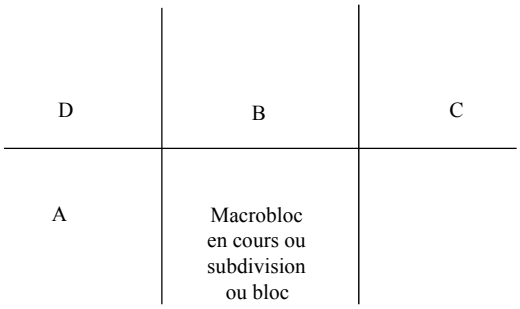


Figure 6-14 – Détermination des macroblocs, blocs et subdivisions du voisinage (pour information)

6.4.8.1 Processus de déduction pour les macroblocs du voisinage

Les résultats de ce processus sont:

– mbAddrA: l'adresse des macroblocs à la gauche du macrobloc en cours et son état de disponibilité; et

– mbAddrB: l'adresse du macroblocs au-dessus du macrobloc en cours et son état de disponibilité.

mbAddrN (N étant A ou B) est déduit comme suit.

– La différence de localisation luma (xD, yD) est établie conformément au Tableau 6-2.

– Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.9, est invoqué pour les localisations luma avec (xN, yN) égal à (xD, yD), et le résultat est alloué à mbAddrN.

6.4.8.2 Processus de déduction pour les blocs luma 8x8 du voisinage

L'entrée dans ce processus est un indice luma8x8BlkIdx de bloc luma 8x8.

L'indice luma8x8BlkIdx spécifie les blocs luma 8x8 d'un macrobloc dans un balayage matriciel.

Les résultats de ce processus sont:

– mbAddrA: égal à CurrMbAddr ou à l'adresse des macroblocs à gauche du macrobloc en cours et son état de disponibilité;

– luma8x8BlkIdxA: indice du bloc luma 8x8 à gauche du bloc 8x8 avec l'indice luma8x8BlkIdx et son état de disponibilité;

– mbAddrB: égal à CurrMbAddr ou à l'adresse des macroblocs au-dessus du macrobloc en cours et son état de disponibilité;

– luma8x8BlkIdxB: indice du bloc luma 8x8 au-dessus du bloc 8x8 avec l'indice luma8x8BlkIdx et son état de disponibilité.

mbAddrN et luma8x8BlkIdxN (N étant A ou B) sont déduits comme suit.

– La différence de localisation luma (xD, yD) est établie conformément au Tableau 6-2.

– La localisation luma (xN, yN) est spécifiée par:

$$xN = (luma8x8BlkIdx \% 2) * 8 + xD \quad (6\text{-}21)$$

$$yN = (luma8x8BlkIdx / 2) * 8 + yD \quad (6\text{-}22)$$

– Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.9, est invoqué pour les localisations luma avec (xN, yN) comme entrée et le résultat est alloué à mbAddrN et (xW, yW).

– La variable luma8x8BlkIdxN est déduite comme suit.

  – Si mbAddrN n'est pas disponible, luma8x8BlkIdxN est marqué comme non disponible.

  – Sinon (si mbAddrN est disponible), le bloc luma 8x8 dans le macrobloc mbAddrN couvrant la localisation luma (xW, yW) est alloué à luma8x8BlkIdxN.

6.4.8.3 Processus de déduction pour les blocs luma 4x4 du voisinage

L'entrée dans ce processus est un indice luma4x4BlkIdx de bloc luma 4x4.

Les résultats de ce processus sont:

– mbAddrA: égal à CurrMbAddr ou à l'adresse du macrobloc à gauche du macrobloc en cours et son état de disponibilité;

– luma4x4BlkIdxA: indice du bloc luma 4x4 à gauche du bloc 4x4 avec l'indice luma4x4BlkIdx et son état de disponibilité;

– mbAddrB: égal à CurrMbAddr ou à l'adresse du macrobloc au-dessus du macrobloc en cours et son état de disponibilité;

– luma4x4BlkIdxB: indice du bloc luma 4x4 au-dessus du bloc 4x4 avec l'indice luma4x4BlkIdx et son état de disponibilité.

mbAddrN et luma4x4BlkIdxN (N étant A ou B) sont déduits comme suit.

– La différence de localisation luma (xD, yD) est établie conformément au Tableau 6-2.

– Le processus de balayage inverse de bloc luma 4x4, comme spécifié au § 6.4.3, est invoqué avec luma4x4BlkIdx comme étant l'entrée et (x, y) comme résultat.

– La localisation luma (xN, yN) est spécifiée par:

$$xN = x + xD \quad (6\text{-}23)$$

$$yN = y + yD \quad (6\text{-}24)$$

– Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.9, est invoqué pour les localisations luma avec (xN, yN) comme entrée et le résultat est alloué à mbAddrN et (xW, yW).

– La variable luma4x4BlkIdxN est déduite comme suit.

  – Si mbAddrN n'est pas disponible, luma4x4BlkIdxN est marqué comme non disponible.

  – Sinon (si mbAddrN est disponible), le bloc luma 4x4 dans le macrobloc mbAddrN couvrant la localisation luma (xW, yW) est alloué à luma4x4BlkIdxN.

6.4.8.4 Processus de déduction pour les blocs chroma 4x4 du voisinage

L'entrée dans ce processus est un indice chroma4x4BlkIdx de bloc chroma 4x4.

Les résultats de ce processus sont:

– mbAddrA (égal à CurrMbAddr ou à l'adresse du macrobloc à gauche du macrobloc en cours) et son état de disponibilité;

– chroma4x4BlkIdxA (indice du bloc chroma 4x4 à gauche du bloc chroma 4x4 avec l'indice chroma4x4BlkIdx) et son état de disponibilité;

– mbAddrB (égal à CurrMbAddr ou à l'adresse du macrobloc au-dessus du macrobloc en cours) et son état de disponibilité;

– chroma4x4BlkIdxB (indice du bloc chroma 4x4 situé au-dessus du bloc chroma 4x4 ayant l'indice chroma4x4BlkIdx) et son état de disponibilité.

mbAddrN et chroma4x4BlkIdxN (N étant A ou B) sont calculés comme suit:

– La différence de localisation des échantillons chroma (xD, yD) est réglée conformément au Tableau 6-2.

– Selon chroma_format_idc, la position (x, y) de l'échantillon supérieur gauche du bloc chroma 4x4 ayant l'indice chroma4x4BlkIdx est calculée comme suit:

  – Si chroma_format_idc est égal à 1 ou 2, ce qui suit est applicable:

    x = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (6-25)

    y = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (6-26)

  – Sinon (si chroma_format_idc est égal à 3), ce qui suit est applicable:

    x = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) +
    InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 0) (6-27)

    y = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) +
    InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 1) (6-28)

– La localisation des échantillons chroma (xN, yN) est spécifiée par:

  xN = x + xD (6-29)

  yN = y + yD (6-30)

– Le processus de calcul pour les localisations voisines tel que spécifié dans le § 6.4.9 est invoqué pour la localisation des échantillons chromas avec (xN, yN) comme entrée et la sortie est attribuée à mbAddrN et (xW, yW).

– La variable chroma4x4BlkIdxN est calculée comme suit:

  – Si mbAddrN est indisponible, chroma4x4BlkIdxN est marqué comme étant indisponible.

  – Sinon (si mbAddrN est disponible), le bloc chroma 4x4 contenu dans le macrobloc mbAddrN recouvrant la localisation des échantillons chroma (xW, yW) est attribué à chroma4x4BlkIdxN.

6.4.8.5 Processus de déduction pour les subdivisions du voisinage

Les entrées dans ce processus sont:

– un indice de subdivision de macrobloc mbPartIdx;

– un type de sous-macrobloc en cours currSubMbType;

– un indice de subdivision de sous-macrobloc subMbPartIdx.

Les résultats de ce processus sont:

– mbAddrA\mbPartIdxA\subMbPartIdxA: spécifiant la subdivision de macrobloc ou sous-macrobloc à gauche du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc CurrMbAddr\mbPartIdx\subMbPartIdx et son état de disponibilité;

– mbAddrB\mbPartIdxB\subMbPartIdxB: spécifiant la subdivision de macrobloc ou sous-macrobloc au-dessus du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc CurrMbAddr\mbPartIdx\subMbPartIdx et son état de disponibilité;

– mbAddrC\mbPartIdxC\subMbPartIdxC: spécifiant la subdivision du macrobloc ou sous-macrobloc au-dessus à droite du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc CurrMbAddr\mbPartIdx\subMbPartIdx et son état de disponibilité;

– mbAddrD\mbPartIdxD\subMbPartIdxD: spécifiant la subdivision de macrobloc ou sous-macrobloc au-dessus à gauche du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc CurrMbAddr\mbPartIdx\subMbPartIdx et son état de disponibilité.

mbAddrN, mbPartIdxN et subMbPartIdx (N étant A, B, C ou D) sont déduits comme suit.

– Le processus de balayage inverse de subdivision de macrobloc, comme décrit au § 6.4.2.1, est invoqué avec mbPartIdx comme entrée et (x, y) comme résultat.

– La localisation de l'échantillon luma supérieur gauche à l'intérieur d'une subdivision de macrobloc (xS, yS) est déduite comme suit.

  – Si mb_type est égal à P_8x8, P_8x8ref0 ou B_8x8, le processus de balayage inverse de subdivision de sous-macrobloc, comme décrit au § 6.4.2.2, est invoqué avec subMbPartIdx comme entrée et (xS, yS) comme résultat.

  – Sinon, (xS, yS) sont établis à (0, 0).

– La variable predPartWidth du Tableau 6-2 est spécifiée comme suit.

  – Si mb_type est égal à P_Skip, B_Skip, ou B_Direct_16x16, predPartWidth = 16.

  – Sinon, si mb_type est égal à B_8x8, on applique ce qui suit.

    – Si currSubMbType est égal à B_Direct_8x8, predPartWidth = 16.

      NOTE 1 – Lorsque currSubMbType est égal à B_Direct_8x8 et que direct_spatial_mv_pred_flag est égal à 1, le vecteur cinétique prédit est le vecteur cinétique prédit pour le macrobloc tout entier.

    – Sinon, predPartWidth = SubMbPartWidth(sub_mb_type[mbPartIdx]).

  – Sinon, si mb_type est égal à P_8x8 ou P_8x8ref0, predPartWidth = SubMbPartWidth(sub_mb_type[mbPartIdx]).

  – Sinon, predPartWidth = MbPartWidth(mb_type).

– La différence de localisation luma (xD, yD) est établie conformément au Tableau 6-2.

– La localisation luma voisine (xN, yN) est spécifiée par:

$$xN = x + xS + xD \tag{6-31}$$

$$yN = y + yS + yD \tag{6-32}$$

– Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.9, est invoqué pour les localisations luma avec (xN, yN) comme entrée et le résultat est alloué à mbAddrN et (xW, yW).

– Selon mbAddrN, on applique ce qui suit.

  – Si mbAddrN n'est pas disponible, la subdivision de macrobloc ou sous-macrobloc mbAddrN\mbPartIdxN\subMbPartIdxN est marquée comme non disponible.

  – Sinon (si mbAddrN est disponible), on applique ce qui suit.

    – La subdivision de macrobloc dans le macrobloc mbAddrN couvrant la localisation luma (xW, yW) est allouée à mbPartIdxN et la sous-subdivision de macrobloc à l'intérieur de la subdivision de macrobloc mbPartIdxN couvrant l'échantillon (xW, yW) dans le macrobloc mbAddrN est allouée à subMbPartIdxN.

    – Lorsque la subdivision donnée par mbPartIdxN et subMbPartIdxN n'est pas encore décodée, la subdivision de macrobloc mbPartIdxN et la sous-subdivision de macrobloc subMbPartIdxN sont marquées comme non disponibles.

      NOTE 2 – C'est le cas de cette dernière condition lorsque, par exemple, mbPartIdx = 2, subMbPartIdx = 3, xD = 4, yD = –1, c'est-à-dire, lorsqu'est exigé C voisin du dernier bloc luma 4x4 du troisième sous-macrobloc.

6.4.9 Processus de déduction pour les localisations du voisinage

L'entrée dans ce processus est une localisation luma ou chroma (xN, yN) exprimée par rapport au coin supérieur gauche du macrobloc en cours.

Les résultats de ce processus sont:

– mbAddrN: égal à CurrMbAddr ou à l'adresse du macrobloc voisin qui contient (xN, yN) et son état de disponibilité;

– (xW, yW): la localisation (xN, yN) exprimée par rapport au coin supérieur gauche du macrobloc mbAddrN (plutôt que par rapport au coin supérieur gauche du macrobloc en cours).

Soit maxW et maxH deux variables spécifiant respectivement la valeur maximale des composantes de la localisation xN, xW et yN, yW. Les variables maxW et maxH sont calculés comme suit.

– Si ce processus est invoqué pour les localisations luma du voisinage,

$$maxW = maxH = 16 \tag{6-33}$$

– Sinon (si ce processus est invoqué pour les localisations chroma du voisinage),

$$maxW = MbWidthC \tag{6-34}$$

$$maxH = MbHeightC \tag{6-35}$$

Selon la variable MbaffFrameFlag, les localisations du voisinage sont déduites comme suit.

– Si MbaffFrameFlag est égal à 0, on applique la spécification pour les localisations du voisinage dans les sous-trames et dans les trames non MBAFF, comme décrit au § 6.4.9.1.

– Sinon (si MbaffFrameFlag est égal à 1), on applique la spécification pour les localisations luma du voisinage dans les trames MBAFF comme décrit au § 6.4.9.2.

6.4.9.1 Spécification pour les localisations de voisinage dans les sous-trames et les trames non MBAFF

Les spécifications du présent paragraphe sont appliquées lorsque MbaffFrameFlag est égal à 0.

Le processus de déduction pour les adresses de macroblocs de voisinage, et pour leur disponibilité dans le § 6.4.6 est invoqué avec mbAddrA, mbAddrB, mbAddrC, et mbAddrD et avec leur état de disponibilité comme résultat.

Le Tableau 6-3 spécifie mbAddrN en fonction de (xN, yN).

Tableau 6-3 – Spécification de mbAddrN

xN	**yN**	**mbAddrN**
< 0	< 0	mbAddrD
< 0	0 .. maxH – 1	mbAddrA
0 .. maxW – 1	< 0	mbAddrB
0 .. maxW – 1	0 .. maxH – 1	CurrMbAddr
> maxW – 1	< 0	mbAddrC
> maxW – 1	0 .. maxH – 1	Indisponible
	> maxH – 1	Indisponible

La localisation de voisinage (xW, yW) se rapportant au coin supérieur gauche du macrobloc mbAddrN est déduite comme suit:

$$xW = (xN + maxW) \% maxW \tag{6-36}$$

$$yW = (yN + maxH) \% maxH \tag{6-37}$$

6.4.9.2 Spécification pour les localisations de voisinage dans les trames MBAFF

Les spécifications du présent paragraphe s'appliquent lorsque MbaffFrameFlag est égal à 1.

Le processus de déduction pour les adresses de macroblocs du voisinage et pour leur disponibilité dans le § 6.4.7 est invoqué avec mbAddrA, mbAddrB, mbAddrC, et mbAddrD et avec leur état de disponibilité comme résultat.

Le Tableau 6-4 spécifie les adresses de macroblocs mbAddrN et yM en deux étapes ordonnées:

1. Spécification d'une adresse de macrobloc mbAddrX en fonction de (xN, yN) et des variables suivantes:
   - La variable currMbFrameFlag est déduite comme suit:
     - Si le macrobloc avec l'adresse CurrMbAddr est un macrobloc de trame, currMbFrameFlag est mis égal à 1.
     - Sinon (si le macrobloc avec l'adresse CurrMbAddr est un macrobloc de sous-trame), currMbFrameFlag est mis égal à 0.
   - La variable mbIsTopMbFlag est déduite comme suit.
     - Si le macrobloc avec l'adresse CurrMbAddr est un macrobloc supérieur (CurrMbAddr % 2 est égal à 0), mbIsTopMbFlag est mis égal à 1;
     - Sinon (si le macrobloc avec l'adresse CurrMbAddr est un macrobloc inférieur, CurrMbAddr % 2 est égal à 1), mbIsTopMbFlag est mis égal à 0.
2. Selon la disponibilité de mbAddrX, on applique ce qui suit.
   - Si mbAddrX n'est pas disponible, mbAddrN est marqué comme indisponible.
   - Sinon (si mbAddrX est disponible), mbAddrN est marqué comme disponible et le Tableau 6-4 spécifie mbAddrN et yM en fonction de (xN, yN), currMbFrameFlag, mbIsTopMbFlag, et la variable mbAddrXFrameFlag, qui est déduite comme suit:
     - si le macrobloc mbAddrX est un macrobloc de trame, mbAddrXFrameFlag est mis égal à 1;
     - Sinon (si le macrobloc mbAddrX est un macrobloc de sous-trame), mbAddrXFrameFlag est mis égal à 0.

Les valeurs non spécifiées (na) des fanions ci-dessus dans le Tableau 6-4 indiquent que la valeur du fanion correspondant n'est pas pertinente pour les rangées en cours du tableau.

Tableau 6-4 – Spécification de mbAddrN et yM

xN	yN	currMbFrameFlag	mbIsTopMbFlag	mbAddrX	mbAddrXFrameFlag	condition supplémentaire	mbAddrN	yM
< 0	< 0	1	1	mbAddrD			mbAddrD + 1	yN
			0	mbAddrA	1		mbAddrA	yN
					0		mbAddrA + 1	(yN + maxH) >> 1
		0	1	mbAddrD	1		mbAddrD + 1	2*yN
					0		mbAddrD	yN
			0	mbAddrD			mbAddrD + 1	yN
< 0	0 .. maxH – 1	1	1	mbAddrA	1		mbAddrA	yN
					0	yN% 2 = = 0	mbAddrA	yN >> 1
						yN% 2 != 0	mbAddrA + 1	yN >> 1
			0	mbAddrA	1		mbAddrA + 1	yN
					0	yN% 2 = = 0	mbAddrA	(yN + maxH) >> 1
						yN% 2 != 0	mbAddrA + 1	(yN + maxH) >> 1
		0	1	mbAddrA	1	yN < (maxH / 2)	mbAddrA	yN <<1
						yN ≥ (maxH / 2)	mbAddrA + 1	(yN <<1) – maxH
					0		mbAddrA	yN
			0	mbAddrA	1	yN < (maxH / 2)	mbAddrA	(yN <<1) + 1
						yN ≥ (maxH / 2)	mbAddrA + 1	(yN <<1) + 1 – maxH
					0		mbAddrA + 1	yN
0 .. maxW – 1	< 0	1	1	mbAddrB			mbAddrB + 1	yN
			0	CurrMbAddr			CurrMbAddr – 1	yN
		0	1	mbAddrB	1		mbAddrB + 1	2 * yN
					0		mbAddrB	yN
			0	mbAddrB			mbAddrB + 1	yN
0 .. maxW – 1	0 .. maxH – 1			CurrMbAddr			CurrMbAddr	yN
> maxW – 1	<0	1	1	mbAddrC			mbAddrC + 1	yN
			0	Indisponible			Indisponible	na
		0	1	mbAddrC	1		mbAddrC + 1	2 * yN
					0		mbAddrC	yN
			0	mbAddrC			mbAddrC + 1	yN
> maxW – 1	0 .. maxH – 1			Indisponible			Indisponible	na
	> maxH – 1			Indisponible			Indisponible	na

La localisation luma voisine (xW, yW) se rapportant au coin supérieur gauche du macrobloc mbAddrN est déduite par:

$$xW = (xN + maxW) \% maxW \quad (6\text{-}38)$$

$$yW = (yM + maxH) \% maxH \quad (6\text{-}39)$$

7 Syntaxe et sémantique

7.1 Méthode de spécification de la syntaxe en forme matricielle

Les tableaux de syntaxe spécifient un surensemble de la syntaxe de tous les flux binaires admis. Des contraintes syntaxiques supplémentaires peuvent être spécifiées dans d'autres paragraphes, soit directement ou indirectement.

NOTE – Un décodeur réel devrait mettre en œuvre des moyens d'identification des points d'entrée dans le flux binaire et des moyens d'identification et de traitement des flux binaires non conformes. Les méthodes d'identification et de traitement des erreurs et d'autres situations similaires ne sont pas spécifiées ici.

Le tableau suivant donne une liste d'exemples de pseudo-code utilisé afin de décrire la syntaxe. Lorsque le code **syntax_element** apparaît, il spécifie qu'un élément syntaxique est analysé syntaxiquement à partir du flux binaire et le pointeur de flux binaire est avancé jusqu'à la prochaine position située au-delà de l'élément syntaxique dans le processus d'analyse syntaxique/sémantique du flux binaire.

	C	Descripteur
/* Une instruction peut être un élément syntaxique avec une catégorie de syntaxe et un descripteur associés ou peut être une expression utilisée afin de spécifier les conditions pour l'existence, le type, et la quantité des éléments syntaxiques, comme dans les deux exemples suivants */		
syntax_element	3	ue(v)
instruction posant une condition		
/* Un groupe d'instructions entre accolades est une instruction composée qui est traitée fonctionnellement comme une instruction unique */		
{		
instruction		
instruction		
…		
}		
/* Une structure "tant que" spécifie un essai visant à déterminer si une condition est vraie. Si c'est le cas, elle spécifie l'évaluation répétitive d'une instruction (ou instruction composée) jusqu'à ce que la condition ne soit plus vraie */		
tant que(condition)		
instruction		
/* Une structure "faire … tant que" spécifie l'évaluation d'une instruction une seule fois, suivie par un essai visant à déterminer si une condition est vraie. Si c'est le cas, elle spécifie une évaluation répétée de l'instruction jusqu'à ce que la condition ne soit plus vraie */		
faire		
instruction		
tant que(condition)		
/* Une structure "si … sinon" spécifie un essai de savoir si une condition est vraie, et si la condition est vraie, elle spécifie l'évaluation d'une instruction primaire, et sinon, elle spécifie l'évaluation d'une instruction de remplacement. La partie "sinon" de la structure et l'instruction de remplacement associée est omise s'il n'est pas besoin d'une évaluation d'instruction de remplacement */		
si(condition)		
instruction primaire		
sinon		
instruction de remplacement		
/* Une structure "pour" spécifie l'évaluation d'une instruction initiale, suivie par l'essai d'une condition, et si la condition est vraie, elle spécifie des évaluations répétées d'une instruction primaire suivie d'une instruction ultérieure jusqu'à ce que la condition ne soit plus vraie */		
pour(instruction initiale; condition; instruction ultérieure)		
instruction primaire		

7.2 Spécification de fonctions, catégories et descripteurs syntaxiques

Les fonctions présentées ici sont utilisées dans la description syntaxique. Ces fonctions supposent l'existence d'un pointeur de flux binaire avec une indication de la position du prochain bit qui sera lu par le processus de décodage à partir du flux binaire.

byte_aligned( ) (*aligné en octets*) spécifié comme suit.

- Si la position actuelle dans le flux binaire est à la frontière d'un octet, c'est-à-dire si le prochain bit dans le flux binaire est le premier bit d'un octet, la valeur résultante de byte_aligned( ) est égale à "TRUE".
- Sinon, la valeur résultante de byte_aligned( ) est égale à "FALSE".

more_data_in_byte_stream( ) (*plus de données dans le flux d'octets*), qui n'est utilisé que dans la structure syntaxique de flux d'octets d'unité NAL spécifiée à l'Annexe B, est spécifié comme suit.

- Si plus de données suivent dans le flux d'octets, la valeur résultante de more_data_in_byte_stream( ) est égale à "TRUE".
- Sinon, la valeur résultante de more_data_in_byte_stream( ) est égale à "FALSE".

more_rbsp_data( ) (*plus de données de charge utile RBSP*) est spécifié comme suit.

- S'il y a plus de données dans une charge utile RBSP avant rbsp_trailing_bits( ), la valeur résultante de more_rbsp_data( ) est égale à "TRUE".
- Sinon, la valeur résultante de more_rbsp_data( ) est égale à "FALSE".

La méthode permettant de déterminer s'il y a plus de données dans la charge utile RBSP est spécifiée par l'application (ou à l'Annexe B pour les applications qui utilisent le format de flux d'octets).

more_rbsp_trailing_data( ) (*plus de données de traîne de charge utile RBSP*) est spécifié comme suit.

- S'il y a plus de données dans une charge utile RBSP, la valeur résultante de more_rbsp_trailing_data( ) est égale à "TRUE".
- Sinon, la valeur résultante de more_rbsp_trailing_data( ) est égale à "FALSE".

next_bits(n) fournit les bits suivants dans le flux binaire pour des besoins de comparaison, sans faire avancer le pointeur de flux binaire. Il permet de jeter un coup d'œil aux n prochains bits dans le flux binaire avec n comme argument. Lorsqu'il est utilisé au sein du flux d'octets, comme spécifié à l'Annexe B, next_bits(n) retourne une valeur de 0 si moins de n bits restent dans le flux d'octets.

read_bits(n) lit les n prochains bits du flux binaire et avance le pointeur de flux binaire de n positions de bit. Lorsque n est égal à 0, read_bits(n) est spécifié afin de retourner une valeur égale à 0 et ne pas avancer le pointeur de flux binaire.

Les catégories (indiquées dans le tableau par **C**) spécifient la subdivision de données de tranche en au plus trois subdivisions de données de tranche. La subdivision de données de tranche A contient tous les éléments syntaxiques de catégorie 2. La subdivision de données de tranche B contient tous les éléments syntaxiques de catégorie 3. La subdivision de données de tranche C contient tous les éléments syntaxiques de catégorie 4. La signification des autres valeurs de catégorie n'est pas spécifiée. Pour certains éléments syntaxiques, deux valeurs de catégorie, séparées par une barre verticale, sont utilisées. Dans ces cas, la valeur de catégorie à appliquer est spécifiée plus loin dans le texte. Pour les structures syntaxiques utilisées au sein d'autres structures syntaxiques, la liste des catégories de tous les éléments syntaxiques trouvés au sein de la structure syntaxique incluse est séparée par une barre verticale. Un élément syntaxique ou structure syntaxique avec la catégorie marquée "Toutes" est présent au sein de toutes les structures syntaxiques qui incluent cet élément ou structure syntaxique. Pour les structures syntaxiques utilisées au sein d'autres structures syntaxiques, on considère qu'une valeur numérique de catégorie fournie dans un tableau syntaxique à l'endroit de l'inclusion d'une structure syntaxique contenant un élément syntaxique dont la catégorie est marquée "Toutes" s'applique aux éléments syntaxiques de catégorie "Toutes".

Les descripteurs suivants spécifient le processus d'analyse syntaxique (et sémantique) de chaque élément syntaxique. Pour certains éléments syntaxiques, deux descripteurs, séparés par une barre verticale, sont utilisés. Dans ces cas, le descripteur de gauche s'applique lorsque entropy_coding_mode_flag (*fanion de mode de codage entropique*) est égal à 0 et le descripteur de droite s'applique lorsque entropy_coding_mode_flag est égal à 1.

- ae(v): élément syntaxique arithmétique à codage entropique adaptable au contexte. Le processus d'analyse syntaxique pour ce descripteur est spécifié au § 9.3.
- b(8): octet ayant un schéma de chaîne binaire (8 bit). Le processus d'analyse syntaxique pour ce descripteur est spécifié par la valeur résultante de la fonction read_bits(8).

– ce(v): élément syntaxique de longueur variable à codage entropique adaptable au contexte avec le bit de gauche en premier. Le processus d'analyse syntaxique pour ce descripteur est spécifié au § 9.2.

– f(n): chaîne binaire à schéma fixe utilisant n bits écrits (de gauche à droite) avec le bit de gauche en premier. Le processus d'analyse syntaxique pour ce descripteur est spécifié par la valeur résultante de la fonction read_bits(n).

– i(n): entier signé utilisant n bits. Lorsque n est "v" dans le tableau syntaxique, le nombre de bits varie en fonction de la valeur des autres éléments syntaxiques. Le processus d'analyse syntaxique pour ce descripteur est spécifié par la valeur résultante de la fonction read_bits(n) interprétée comme la représentation entière d'un complément à deux avec le bit de plus fort poids écrit en premier.

– me(v): élément syntaxique mappé et codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse syntaxique pour ce descripteur est spécifié au § 9.1.

– se(v): élément syntaxique entier signé et codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse syntaxique pour ce descripteur est spécifié au § 9.1.

– te(v): élément syntaxique tronqué et codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse syntaxique pour ce descripteur est spécifié au § 9.1.

– u(n): entier non signé utilisant n bits. Lorsque n est "v" dans le tableau syntaxique, le nombre de bits varie en fonction de la valeur des autres éléments syntaxiques. Le processus d'analyse syntaxique pour ce descripteur est spécifié par la valeur résultante de la fonction read_bits(n) interprétée comme une représentation binaire d'un entier non signé avec le bit de plus fort poids écrit en premier.

– ue(v): élément syntaxique entier non signé codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse syntaxique pour ce descripteur est spécifié au § 9.1.

7.3 Syntaxe en forme tabulaire

7.3.1 Syntaxe d'unité NAL

nal_unit(NumBytesInNALunit) {	C	Descripteur
forbidden_zero_bit	Toutes	f(1)
nal_ref_idc	Toutes	u(2)
nal_unit_type	Toutes	u(5)
NumBytesInRBSP = 0		
pour(i = 1; i < NumBytesInNALunit; i++) {		
si(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {		
rbsp_byte[NumBytesInRBSP++ **]**	Toutes	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	Toutes	b(8)
i += 2		
emulation_prevention_three_byte /* égal à 0x03 */	Toutes	f(8)
} sinon		
rbsp_byte[NumBytesInRBSP++ **]**	Toutes	b(8)
}		
}		

7.3.2 Syntaxe de charges utiles de séquence d'octets bruts et de bits de traîne de charge utile RBSP

7.3.2.1 Syntaxe de charge utile RBSP d'ensemble de paramètres de séquence

seq_parameter_set_rbsp( ) {	C	Descripteur
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
constraint_set3_flag	0	u(1)
reserved_zero_4bits /* égal à 0 */	0	u(4)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
si(profile_idc == 100 || profile_idc == 110 || profile_idc == 122 || profile_idc == 144) {		
chroma_format_idc	0	ue(v)

si(chroma_format_idc == 3)		
residual_colour_transform_flag	0	u(1)
bit_depth_luma_minus8	0	ue(v)
bit_depth_chroma_minus8	0	ue(v)
qpprime_y_zero_transform_bypass_flag	0	u(1)
seq_scaling_matrix_present_flag	0	u(1)
si(seq_scaling_matrix_present_flag)		
pour(i = 0; i < 8; i++) {		
seq_scaling_list_present_flag[i **]**	0	u(1)
si(seq_scaling_list_present_flag[i])		
si(i < 6)		
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	0	
sinon		
scaling_list(ScalingList8x8[i – 6], 64, UseDefaultScalingMatrix8x8Flag[i – 6])	0	
}		
}		
log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
si(pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
sinon si(pic_order_cnt_type == 1) {		
delta_pic_order_always_zero_flag	0	u(1)
offset_for_non_ref_pic	0	se(v)
offset_for_top_to_bottom_field	0	se(v)
num_ref_frames_in_pic_order_cnt_cycle	0	ue(v)
pour(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)		
offset_for_ref_frame[i **]**	0	se(v)
}		
num_ref_frames	0	ue(v)
gaps_in_frame_num_value_allowed_flag	0	u(1)
pic_width_in_mbs_minus1	0	ue(v)
pic_height_in_map_units_minus1	0	ue(v)
frame_mbs_only_flag	0	u(1)
si(!frame_mbs_only_flag)		
mb_adaptive_frame_field_flag	0	u(1)
direct_8x8_inference_flag	0	u(1)
frame_cropping_flag	0	u(1)
si(frame_cropping_flag) {		
frame_crop_left_offset	0	ue(v)
frame_crop_right_offset	0	ue(v)
frame_crop_top_offset	0	ue(v)
frame_crop_bottom_offset	0	ue(v)
}		
vui_parameters_present_flag	0	u(1)
si(vui_parameters_present_flag)		
vui_parameters( )	0	
rbsp_trailing_bits( )	0	
}		

7.3.2.1.1 Syntaxe de liste de normalisation

scaling_list(scalingList, sizeOfScalingList, useDefaultScalingMatrixFlag) {	C	Descripteur
lastScale = 8		
nextScale = 8		
pour(j = 0; j < sizeOfScalingList; j++) {		
si(nextScale != 0) {		
delta_scale	0 | 1	se(v)
nextScale = (lastScale + delta_scale + 256) % 256		
useDefaultScalingMatrixFlag = (j == 0 && nextScale == 0)		
}		
scalingList[j] = (nextScale == 0) ? lastScale : nextScale		
lastScale = scalingList[j]		
}		
}		

7.3.2.1.2 Syntaxe de charge utile RBSP d'ensemble de paramètres de séquence

seq_parameter_set_extension_rbsp( ) {	C	Descripteur
seq_parameter_set_id	10	ue(v)
aux_format_idc	10	ue(v)
si(aux_format_idc != 0) {		
bit_depth_aux_minus8	10	ue(v)
alpha_incr_flag	10	u(1)
alpha_opaque_value	10	u(v)
alpha_transparent_value	10	u(v)
}		
additional_extension_flag	10	u(1)
rbsp_trailing_bits()	10	
}		

7.3.2.2 Syntaxe de charge utile RBSP d'ensemble de paramètres d'image

pic_parameter_set_rbsp( ) {	C	Descripteur
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
entropy_coding_mode_flag	1	u(1)
pic_order_present_flag	1	u(1)
num_slice_groups_minus1	1	ue(v)
si(num_slice_groups_minus1 > 0) {		
slice_group_map_type	1	ue(v)
si(slice_group_map_type == 0)		
pour(iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++)		
run_length_minus1[iGroup **]**	1	ue(v)
sinon si(slice_group_map_type == 2)		
pour(iGroup = 0; iGroup < num_slice_groups_minus1; iGroup++) {		
top_left[iGroup **]**	1	ue(v)
bottom_right[iGroup **]**	1	ue(v)
}		
sinon si(slice_group_map_type == 3 || slice_group_map_type == 4 || slice_group_map_type == 5) {		
slice_group_change_direction_flag	1	u(1)
slice_group_change_rate_minus1	1	ue(v)
} sinon si(slice_group_map_type == 6) {		
pic_size_in_map_units_minus1	1	ue(v)
pour(i = 0; i <= pic_size_in_map_units_minus1; i++)		
slice_group_id[i **]**	1	u(v)

}		
}		
num_ref_idx_l0_active_minus1	1	ue(v)
num_ref_idx_l1_active_minus1	1	ue(v)
weighted_pred_flag	1	u(1)
weighted_bipred_idc	1	u(2)
pic_init_qp_minus26 /* par rapport à 26 */	1	se(v)
pic_init_qs_minus26 /* par rapport à 26 */	1	se(v)
chroma_qp_index_offset	1	se(v)
deblocking_filter_control_present_flag	1	u(1)
constrained_intra_pred_flag	1	u(1)
redundant_pic_cnt_present_flag	1	u(1)
si(more_rbsp_data( )) {		
transform_8x8_mode_flag	1	u(1)
pic_scaling_matrix_present_flag	1	u(1)
si(pic_scaling_matrix_present_flag)		
pour(i = 0; i < 6 + 2* transform_8x8_mode_flag; i++) {		
pic_scaling_list_present_flag[i **]**	1	u(1)
si(pic_scaling_list_present_flag[i])		
si(i < 6)		
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	1	
sinon		
scaling_list(ScalingList8x8[i – 6], 64, UseDefaultScalingMatrix8x8Flag[i – 6])	1	
}		
second_chroma_qp_index_offset	1	se(v)
}		
rbsp_trailing_bits( )	1	
}		

7.3.2.3 Syntaxe de charge utile RBSP d'informations d'amélioration supplémentaires

sei_rbsp( ) {	**C**	**Descripteur**
faire		
sei_message( )	5	
tant que (more_rbsp_data( ))		
rbsp_trailing_bits( )	5	
}		

7.3.2.3.1 Syntaxe de message d'informations d'amélioration supplémentaires

sei_message( ) {	**C**	**Descriptor**
payloadType = 0		
while(next_bits(8) = = 0xFF) {		
ff_byte /* equal to 0xFF */	5	f(8)
payloadType += 255		
}		
last_payload_type_byte	5	u(8)
payloadType += last_payload_type_byte		
payloadSize = 0		
while(next_bits(8) = = 0xFF) {		
ff_byte /* equal to 0xFF */	5	f(8)
payloadSize += 255		
}		
last_payload_size_byte	5	u(8)
payloadSize += last_payload_size_byte		
sei_payload(payloadType, payloadSize)	5	
}		

7.3.2.4 Syntaxe de charge utile RBSP de délimiteur d'unité d'accès

access_unit_delimiter_rbsp( ) {	C	Descripteur
primary_pic_type	6	u(3)
rbsp_trailing_bits( )	6	
}		

7.3.2.5 Syntaxe de charge utile RBSP de fin de séquence

end_of_seq_rbsp( ) {	C	Descripteur
}		

7.3.2.6 Syntaxe de charge utile RBSP de fin de flux

end_of_stream_rbsp( ) {	C	Descripteur
}		

7.3.2.7 Syntaxe de charge utile RBSP de données de remplissage

filler_data_rbsp( ) {	C	Descripteur
tant que(next_bits(8) == 0xFF)		
ff_byte /* égal à 0xFF */	9	f(8)
rbsp_trailing_bits( )	9	
}		

7.3.2.8 Syntaxe de charge utile RBSP de couche de tranche sans subdivision

slice_layer_without_partitioning_rbsp( ) {	C	Descripteur
slice_header( )	2	
slice_data( ) /* toutes les catégories de syntaxe de slice_data( ) */	2 | 3 | 4	
rbsp_slice_trailing_bits( )	2	
}		

7.3.2.9 Syntaxe de charge utile RBSP de subdivision de données de tranche

7.3.2.9.1 Syntaxe de charge utile RBSP de subdivision de données de tranche A

slice_data_partition_a_layer_rbsp( ) {	C	Descripteur
slice_header( )	2	
slice_id	Toutes	ue(v)
slice_data( ) /* seulement les parties de catégorie 2 de la syntaxe de slice_data( ) */	2	
rbsp_slice_trailing_bits( )	2	
}		

7.3.2.9.2 Syntaxe de charge utile RBSP de subdivision de données de tranche B

slice_data_partition_b_layer_rbsp( ) {	C	Descripteur
slice_id	Toutes	ue(v)
si(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	Toutes	ue(v)
slice_data( ) /* seulement les parties de catégorie 3 de la syntaxe de slice_data( ) */	3	
rbsp_slice_trailing_bits( )	3	
}		

7.3.2.9.3 Syntaxe de charge utile RBSP de subdivision de données de tranche C

slice_data_partition_c_layer_rbsp( ) {	C	Descripteur
slice_id	Toutes	ue(v)
si(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	Toutes	ue(v)
slice_data( ) /* seulement les parties de catégorie 4 de la syntaxe de slice_data( ) */	4	
rbsp_slice_trailing_bits( )	4	
}		

7.3.2.10 Syntaxe des bits de traîne de tranche de charge utile RBSP

rbsp_slice_trailing_bits( ) {	C	Descripteur
rbsp_trailing_bits( )	Toutes	
si(entropy_coding_mode_flag)		
tant que(more_rbsp_trailing_data( ))		
cabac_zero_word /* égal à 0x0000 */	Toutes	f(16)
}		

7.3.2.11 Syntaxe des bits de traîne de charge utile RBSP

rbsp_trailing_bits( ) {	C	Descripteur
rbsp_stop_one_bit /* égal à 1 */	Toutes	f(1)
tant que(!byte_aligned( ))		
rbsp_alignment_zero_bit /* égal à 0 */	Toutes	f(1)
}		

7.3.3 Syntaxe d'en-tête de tranche

slice_header( ) {	C	Descripteur
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
si(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
si(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
si(nal_unit_type = = 5)		
idr_pic_id	2	ue(v)
si(pic_order_cnt_type = = 0) {		
pic_order_cnt_lsb	2	u(v)
si(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
si(pic_order_cnt_type = = 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0 **]**	2	se(v)
si(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1 **]**	2	se(v)
}		
si(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
si(slice_type = = B)		
direct_spatial_mv_pred_flag	2	u(1)
si(slice_type = = P | | slice_type = = SP | | slice_type = = B) {		
num_ref_idx_active_override_flag	2	u(1)

si(num_ref_idx_active_override_flag) {		
num_ref_idx_l0_active_minus1	2	ue(v)
si(slice_type == B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
ref_pic_list_reordering( )	2	
si((weighted_pred_flag && (slice_type == P || slice_type == SP)) || (weighted_bipred_idc == 1 && slice_type == B))		
pred_weight_table( )	2	
si(nal_ref_idc != 0)		
dec_ref_pic_marking( )	2	
si(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
si(slice_type == SP || slice_type == SI) {		
si (slice_type == SP)		
sp_for_switch_flag	2	u(1)
slice_qs_delta	2	se(v)
}		
si(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
si (disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
si(num_slice_groups_minus1 > 0 && slice_group_map_type >= 3 && slice_group_map_type <= 5)		
slice_group_change_cycle	2	u(v)
}		

7.3.3.1 Syntaxe de réorganisation de la liste d'images de référence

ref_pic_list_reordering( ) {	C	Descripteur
si(slice_type != I && slice_type != SI) {		
ref_pic_list_reordering_flag_l0	2	u(1)
si(ref_pic_list_reordering_flag_l0)		
faire {		
reordering_of_pic_nums_idc	2	ue(v)
si(reordering_of_pic_nums_idc == 0 || reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
sinon si(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
} tant que(reordering_of_pic_nums_idc != 3)		
}		
si(slice_type == B) {		
ref_pic_list_reordering_flag_l1	2	u(1)
si(ref_pic_list_reordering_flag_l1)		
faire{		
reordering_of_pic_nums_idc	2	ue(v)
si(reordering_of_pic_nums_idc == 0 || reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
sinon si(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
} tant que(reordering_of_pic_nums_idc != 3)		
}		
}		

7.3.3.2 Syntaxe de tableau de pondération de prédiction

pred_weight_table( ) {	C	Descripteur
luma_log2_weight_denom	2	ue(v)
si(chroma_format_idc != 0)		
chroma_log2_weight_denom	2	ue(v)
pour(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {		
luma_weight_l0_flag	2	u(1)
si(luma_weight_l0_flag) {		
luma_weight_l0[i **]**	2	se(v)
luma_offset_l0[i **]**	2	se(v)
}		
si(chroma_format_idc != 0) {		
chroma_weight_l0_flag	2	u(1)
si(chroma_weight_l0_flag)		
pour(j =0; j < 2; j++) {		
chroma_weight_l0[i **][** j **]**	2	se(v)
chroma_offset_l0[i **][** j **]**	2	se(v)
}		
}		
}		
si(slice_type == B)		
pour(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {		
luma_weight_l1_flag	2	u(1)
si(luma_weight_l1_flag) {		
luma_weight_l1[i **]**	2	se(v)
luma_offset_l1[i **]**	2	se(v)
}		
si(chroma_format_idc != 0) {		
chroma_weight_l1_flag	2	u(1)
si(chroma_weight_l1_flag)		
pour(j = 0; j < 2; j++) {		
chroma_weight_l1[i **][** j **]**	2	se(v)
chroma_offset_l1[i **][** j **]**	2	se(v)
}		
}		
}		
}		

7.3.3.3 Syntaxe de marquage d'image de référence décodée

dec_ref_pic_marking( ) {	C	Descripteur
si(nal_unit_type == 5) {		
no_output_of_prior_pics_flag	2 | 5	u(1)
long_term_reference_flag	2 | 5	u(1)
} sinon {		
adaptive_ref_pic_marking_mode_flag	2 | 5	u(1)
si(adaptive_ref_pic_marking_mode_flag)		
faire{		
memory_management_control_operation	2 | 5	ue(v)
si(memory_management_control_operation == 1 || memory_management_control_operation == 3)		
difference_of_pic_nums_minus1	2 | 5	ue(v)
si(memory_management_control_operation == 2)		
Long_term_pic_num	2 | 5	ue(v)
si(memory_management_control_operation == 3 || memory_management_control_operation == 6)		
Long_term_frame_idx	2 | 5	ue(v)
si(memory_management_control_operation == 4)		

Max_long_term_frame_idx_plus1	2 | 5	ue(v)
} tant que(memory_management_control_operation != 0)		
}		
}		

7.3.4 Syntaxe de données de tranche

slice_data( ) {	C	Descripteur
si(entropy_coding_mode_flag)		
tant que(!byte_aligned( ))		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
faire{		
si(slice_type != I && slice_type != SI)		
si(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
pour(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data( )		
} sinon {		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
si(moreDataFlag) {		
si(MbaffFrameFlag && (CurrMbAddr % 2 == 0 || (CurrMbAddr % 2 == 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) | ae(v)
Macroblock_layer( )	2 | 3 | 4	
}		
si(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data( )		
sinon {		
si(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
si(MbaffFrameFlag && CurrMbAddr % 2 == 0)		
moreDataFlag = 1		
sinon {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} tant que(moreDataFlag)		
}		

7.3.5 Syntaxe de couche de macrobloc

macroblock_layer( ) {	C	Descripteur
mb_type	2	ue(v) | ae(v)
si(mb_type == I_PCM) {		
tant que(!byte_aligned( ))		
pcm_alignment_zero_bit	2	f(1)
pour(i = 0; i < 256; i++)		
pcm_sample_luma[i **]**	2	u(v)
pour(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i **]**	2	u(v)
} sinon {		

	noSubMbPartSizeLessThan8x8Flag = 1		
	si(mb_type != I_NxN && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) == 4) {		
	sub_mb_pred(mb_type)	2	
	pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
	si(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
	si(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
	noSubMbPartSizeLessThan8x8Flag = 0		
	} sinon si(!direct_8x8_inference_flag)		
	noSubMbPartSizeLessThan8x8Flag = 0		
	} sinon {		
	si(transform_8x8_mode_flag && mb_type == I_NxN)		
	transform_size_8x8_flag	2	u(1) | ae(v)
	mb_pred(mb_type)	2	
	}		
	si(MbPartPredMode(mb_type, 0) != Intra_16x16) {		
	coded_block_pattern	2	me(v) | ae(v)
	si(CodedBlocPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (mb_type != B_Direct_16x16 || direct_8x8_inference_flag))		
	transform_size_8x8_flag	2	u(1) | ae(v)
	}		
	si(CodedBlocPatternLuma > 0 || CodedBlocPatternChroma > 0 || MbPartPredMode(mb_type, 0) == Intra_16x16) {		
	mb_qp_delta	2	se(v) | ae(v)
fv	residual( )	3 | 4	
	}		
	}		
}			

7.3.5.1 Syntaxe de la prédiction de macrobloc

mb_pred(mb_type) {	C	Descripteur
si(MbPartPredMode(mb_type, 0) == Intra_4x4 || MbPartPredMode(mb_type, 0) == Intra_8x8 || MbPartPredMode(mb_type, 0) == Intra_16x16) {		
si(MbPartPredMode(mb_type, 0) == Intra_4x4)		
pour(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
si(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
si(MbPartPredMode(mb_type, 0) == Intra_8x8)		
pour(luma8x8BlkIdx=0; luma8x8BlkIdx<4; luma8x8BlkIdx++) {		
prev_intra8x8_pred_mode_flag[luma8x8BlkIdx **]**	2	u(1) | ae(v)
si(!prev_intra8x8_pred_mode_flag[luma8x8BlkIdx])		
rem_intra8x8_pred_mode[luma8x8BlkIdx **]**	2	u(3) | ae(v)
}		
si(chroma_format_idc != 0)		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} sinon si(MbPartPredMode(mb_type, 0) != Direct) {		
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		

si((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si(MbPartPredMode (mb_type, mbPartIdx) != Pred_L1)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si(MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

7.3.5.2 Syntaxe de la prédiction de sous-macrobloc

sub_mb_pred(mb_type) {	C	Descripteur
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type[mbPartIdx **]**	2	ue(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
pour(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
pour(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	2	se(v) | ae(v)
}		

7.3.5.3 Syntaxe des données résiduelles

residual( ) {	C	Descriptor
si(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
sinon		
residual_block = residual_block_cabac		
si(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16DCLevel, 16)	3	
pour(i8x8 = 0; i8x8 < 4; i8x8++) /* chaque bloc luma 8x8 */		
si(!transform_size_8x8_flag || !entropy_coding_mode_flag)		

pour(i4x4 = 0; i4x4 < 4; i4x4++) { /* chaque sous-bloc 4x4 de bloc */		
si(CodedBlockPatternLuma & (1 << i8x8))		
si(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
sinon		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 | 4	
sinon si(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
pour(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
sinon		
pour(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
si(!entropy_coding_mode_flag && transform_size_8x8_flag)		
pour(i = 0; i < 16; i++)		
LumaLevel8x8[i8x8][4 * i + i4x4] = LumaLevel[i8x8 * 4 + i4x4][i]		
}		
sinon si(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(LumaLevel8x8[i8x8], 64)	3 | 4	
else		
pour(i = 0; i < 64; i++)		
LumaLevel8x8[i8x8][i] = 0		
(chroma_format_idc != 0) {		
NumC8x8 = 4 / (SubWidthC * SubHeightC)		
pour(iCbCr = 0; iCbCr < 2; iCbCr++)		
si(CodedBlockPatternChroma & 3) /* constante chromatique résiduelle présente */		
residual_block(ChromaDCLevel[iCbCr], 4 * NumC8x8)	3 | 4	
sinon		
pour(i = 0; i < 4 * NumC8x8; i++)		
ChromaDCLevel[iCbCr][i] = 0		
pour(iCbCr = 0; iCbCr < 2; iCbCr++)		
pour(i8x8 = 0; i8x8 < NumC8x8; i8x8++)		
pour(i4x4 = 0; i4x4 < 4; i4x4++)		
si(CodedBlockPatternChroma & 2) /* variable chromatique résiduelle présente */		
residual_block(ChromaACLevel[iCbCr][i8x8*4+i4x4], 15)	3 | 4	
sinon		
pour(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i8x8*4+i4x4][i] = 0		
}		

7.3.5.3.1 Syntaxe du codage CAVLC de bloc résiduel

residual_block_cavlc(coeffLevel, maxNumCoeff) {	C	Descripteur
pour(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
coeff_token	3 | 4	ce(v)
si(TotalCoeff(coeff_token) > 0) {		
si(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)		
suffixLength = 1		
sinon		
suffixLength = 0		
pour(i = 0; i < TotalCoeff(coeff_token); i++)		
si(i < TrailingOnes(coeff_token)) {		
trailing_ones_sign_flag	3 | 4	u(1)
level[i] = 1 – 2 * trailing_ones_sign_flag		
} sinon {		
level_prefix	3 | 4	ce(v)
levelCode = (Min(15, level_prefix << suffixLength)		
si(suffixLength > 0 || level_prefix >= 14) {		
level_suffix	3 | 4	u(v)

levelCode += level_suffix		
}		
si(level_prefix >= 15 && suffixLength == 0)		
levelCode += 15		
si(level_prefix >= 16)		
levelCode += (1 << (level_prefix – 3)) – 4096		
si(i == TrailingOnes(coeff_token) && TrailingOnes(coeff_token) < 3)		
levelCode += 2		
si(levelCode % 2 == 0)		
level[i] = (levelCode + 2) >> 1		
sinon		
level[i] = (–levelCode – 1) >> 1		
si(suffixLength == 0)		
suffixLength = 1		
si(Abs(level[i]) > (3 << (suffixLength – 1)) && suffixLength < 6)		
suffixLength++		
}		
si(TotalCoeff(coeff_token) < maxNumCoeff) {		
total_zeros	3 | 4	ce(v)
zerosLeft = total_zeros		
} sinon		
zerosLeft = 0		
pour(i = 0; i < TotalCoeff(coeff_token) – 1; i++) {		
si(zerosLeft > 0) {		
run_before	3 | 4	ce(v)
run[i] = run_before		
} sinon		
run[i] = 0		
zerosLeft = zerosLeft – run[i]		
}		
run[TotalCoeff(coeff_token) – 1] = zerosLeft		
coeffNum = – 1		
pour(i = TotalCoeff(coeff_token) – 1; i >= 0; i--) {		
coeffNum += run[i] + 1		
coeffLevel[coeffNum] = level[i]		
}		
}		
}		

7.3.5.3.2 Syntaxe du codage CABAC de bloc résiduel

residual_block_cabac(coeffLevel, maxNumCoeff) {	C	Descripteur
si(maxNumCoeff == 64)		
coded_block_flag = 1		
sinon		
coded_block_flag	3 | 4	ae(v)
si(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
faire {		
significant_coeff_flag[i **]**	3 | 4	ae(v)
si(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i **]**	3 | 4	ae(v)
si(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
pour(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		

i++		
} tant que(i < numCoeff – 1)		
coeff_abs_level_minus1[numCoeff – 1 **]**	3 | 4	ae(v)
coeff_sign_flag[numCoeff – 1 **]**	3 | 4	ae(v)
coeffLevel[numCoeff – 1] = (coeff_abs_level_minus1[numCoeff – 1] + 1) * (1 – 2 * coeff_sign_flag[numCoeff – 1])		
pour(i = numCoeff – 2; i >= 0; i--)		
si(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i **]**	3 | 4	ae(v)
coeff_sign_flag[i **]**	3 | 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) * (1 – 2 * coeff_sign_flag[i])		
} sinon		
coeffLevel[i] = 0		
} sinon		
pour(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

7.4 Sémantique

La sémantique associée aux structures syntaxiques et aux éléments syntaxiques contenus dans ces structures est spécifiée dans le présent paragraphe. Lorsque la sémantique d'un élément syntaxique est spécifiée au moyen d'un tableau ou d'un ensemble de tableaux, toutes les valeurs qui ne sont pas spécifiées dans ce ou ces tableaux ne doivent pas être présentes dans le flux binaire sauf spécification contraire dans la présente Recommandation | Norme internationale.

7.4.1 Sémantique d'unité NAL

NOTE 1 – La couche VCL est spécifiée afin de représenter efficacement le contenu des données vidéo. La couche NAL est spécifiée afin de formater ces données et fournir des informations d'en-tête d'une façon appropriée pour le transport sur des canaux de communication ou pour des moyens de stockage variés. Toutes les données sont contenues dans les unités NAL, chacune d'elles contenant un nombre entier d'octets. Une unité NAL spécifie un format générique à utiliser à la fois pour les systèmes en mode paquet et pour les systèmes à flux binaire. Le format des unités NAL est identique aussi bien pour le transport en mode paquet que pour le flux d'octets sauf que chaque unité NAL peut être précédée d'un préfixe de code de déclenchement et d'octets de bourrage dans le format de flux d'octets.

NumBytesInNALunit spécifie la taille de l'unité NAL en octets. Cette valeur est nécessaire pour le décodage de l'unité NAL. Une certaine forme de démarcation des frontières de l'unité NAL est nécessaire afin de permettre le calcul de NumBytesInNALunit. Une telle méthode de démarcation est spécifiée à l'Annexe B pour le format de flux d'octets. D'autres méthodes de démarcation peuvent être spécifiées en dehors de la présente Recommandation | Norme internationale.

forbidden_zero_bit doit être égal à 0.

nal_ref_idc non égal à 0 spécifie que le contenu de l'unité NAL comporte un ensemble de paramètres de séquence ou un ensemble de paramètres d'image ou une tranche d'une image de référence ou subdivision de données de tranche d'une image de référence.

nal_ref_idc égal à 0 pour une unité NAL contenant une tranche ou subdivision de données de tranche indique que la tranche ou subdivision de données de tranche fait partie d'une image non référentielle.

nal_ref_idc ne doit pas être égal à 0 pour les unités NAL d'ensemble de paramètres de séquence ou d'extension d'ensemble de paramètres de séquence ou d'ensemble de paramètres d'image. Lorsque nal_ref_idc est égal à 0 pour une unité NAL de tranche ou subdivision de données de tranche d'une image particulière, il doit être égal à 0 pour toute unité NAL de tranche et subdivision de données de tranche de l'image.

nal_ref_idc doit être non égal à 0 pour les unités NAL IDR, c'est-à-dire, les unités NAL avec nal_unit_type égal à 5.

nal_ref_idc doit être égal à 0 pour toutes les unités NAL ayant nal_unit_type égal à 6, 9, 10, 11 ou 12.

nal_unit_type spécifie le type de structure de données de charge utile RBSP contenues dans l'unité NAL comme spécifié au Tableau 7-1. Les unités NAL de couche VCL sont spécifiées comme étant celles des unités NAL qui ont nal_unit_type égal à 1 à 5, inclus. Toutes les unités NAL restantes sont appelées unités NAL non VCL.

La colonne marquée "C" dans le Tableau 7-1 donne la liste des catégories d'éléments syntaxiques qui peuvent être présents dans l'unité NAL. De plus, des éléments syntaxiques avec la catégorie de syntaxe "Toutes" peuvent être présents, selon la détermination de la syntaxe et la sémantique de la structure des données de la charge utile RBSP. La présence ou l'absence de tous éléments syntaxiques d'une catégorie particulière de la liste est déterminée à partir de la syntaxe et de la sémantique de la structure des données de la charge utile RBSP. nal_unit_type ne doit pas être égal à 3 ou 4 à moins qu'au moins un élément syntaxique ne soit présent dans la structure des données de la charge utile RBSP ayant une valeur de catégorie d'élément syntaxique égale à la valeur de nal_unit_type et n'appartenant pas à la catégorie "Toutes".

Tableau 7-1 – Codes de type d'unité NAL

nal_unit_type	Contenu de la structure syntaxique de l'unité NAL et de la charge RBSP	C
0	Non spécifié	
1	Tranche codée d'une image non IDR slice_layer_without_subdivision_rbsp( )	2, 3, 4
2	Subdivision A de données de tranche codée slice_data_partition_a_layer_rbsp( )	2
3	Subdivision B de données de tranche codée slice_data_partition_b_layer_rbsp( )	3
4	Subdivision C de données de tranche codée slice_data_partition_c_layer_rbsp( )	4
5	Tranche codée d'uneimage à rafraîchissement IDR slice_layer_without_subdivision_rbsp( )	2, 3
6	Informations d'amélioration supplémentaires (SEI, *supplemental enhancement information*) sei_rbsp( )	5
7	Ensemble de paramètres de séquence seq_parameter_set_rbsp( )	0
8	Ensemble de paramètres d'image pic_parameter_set_rbsp( )	1
9	Délimiteur d'unité d'accès access_unit_delimiter_rbsp( )	6
10	Fin de séquence end_of_seq_rbsp( )	7
11	Fin de flux end_of_stream_rbsp( )	8
12	Données de remplissage filler_data_rbsp( )	9
13	Extension d'ensemble de paramètres de séquence seq_parameter_set_extension_rbsp( )	10
14..18	Réservé	
19	Tranche codée d'une image codée auxiliaire sans subdivision slice_layer_without_partitioning_rbsp( )	2, 3, 4
20..23	Réservé	
24..31	Non spécifié	

Les unités de couche NAL qui utilisent nal_unit_type égal à 13 et 19 peuvent être rejetées par les décodeurs sans affecter le processus de décodage pour les unités de couche NAL qui utilisent nal_unit_type non égal à 13 ou 19 et sans affecter la conformité à la présente Recommandation | Norme internationale.

Les unités NAL qui utilisent nal_unit_type égal à 0 ou dans la gamme de 24..31, inclusivement, ne doivent pas affecter le processus de décodage spécifié dans la présente Recommandation | Norme internationale.

NOTE 2 – Les types d'unité NAL 0 et 24..31 peuvent être utilisés comme déterminé par l'application. Il n'est pas spécifié de décodage pour ces valeurs de nal_unit_type dans la présente Recommandation | Norme internationale.

Les décodeurs doivent ignorer (retirer du flux binaire et mettre à l'écart) le contenu de toutes les unités NAL qui utilisent les valeurs réservées de nal_unit_type.

NOTE 3 – Cette exigence permet une définition d'extensions futures compatibles avec la présente Recommandation | Norme internationale.

Dans le texte, l'expression *unité NAL de tranche codée* se rapporte collectivement à une tranche codée d'unité NAL d'une image non IDR ou à une tranche codée d'unité NAL d'image à rafraîchissement IDR.

Lorsque la valeur de nal_unit_type est égale à 5 pour une unité NAL contenant une tranche d'image codée, la valeur de nal_unit_type doit être 5 dans toutes les autres unités NAL de couche VCL de la même image codée. Une telle image est appelée uneimage à rafraîchissement IDR.

NOTE 4 – Une subdivision de données de tranche ne peut pas être utilisée pour des images à rafraîchissement IDR.

rbsp_byte[i **]** est le ième octet d'une charge utile RBSP. Une charge utile RBSP est spécifiée comme une séquence ordonnée d'octets comme suit.

La charge utile contient une chaîne SODB (*chaîne de bits de données*) comme suit.

– Si la chaîne SODB est vide (c'est-à-dire, zéro bit en longueur), la charge utile RBSP est vide elle aussi.

– Sinon, la charge utile RBSP contient la chaîne SODB comme suit.

1) Le premier octet de la charge utile RBSP contient les huit bits (celui de plus fort poids le plus à gauche) de la chaîne SODB; le prochain octet de la charge utile RBSP doit contenir les huit bits suivants de la chaîne SODB, etc., jusqu'à ce que restent moins de huit bits de la chaîne SODB.

2) rbsp_trailing_bits( ) sont présents après la chaîne SODB comme suit:

   i) les premiers bits (celui de plus fort poids le plus à gauche) du dernier octet de la charge utile RBSP contient les bits restants de la chaîne SODB, (s'il y en a);

   ii) le bit suivant consiste en un seul rbsp_stop_one_bit égal à 1;

   iii) lorsque le rbsp_stop_one_bit n'est pas le dernier bit d'un octet aligné sur les octets, un ou plusieurs rbsp_alignment_zero_bit sont présents afin d'assurer l'alignement d'octet.

3) Un ou plusieurs éléments syntaxiques à 16 bit cabac_zero_word égaux à 0x0000 peuvent être présents dans certaines charges utiles RBSP après les rbsp_trailing_bits( ) à la fin de la charge utile RBSP.

Les structures syntaxiques ayant ces propriétés de charge utile RBSP sont notées dans les tableaux syntaxiques au moyen d'un suffixe "_rbsp". Ces structures doivent être transportées au sein des unités NAL en tant que contenu des octets de données de rbsp_byte[i]. L'association des structures syntaxiques de la charge utile RBSP aux unités NAL doit être conforme à ce qui est spécifié au Tableau 7-1.

NOTE 5 – Lorsque les frontières de la charge utile RBSP sont connues, le décodeur peut extraire la chaîne SODB de la charge utile RBSP par concaténation des bits des octets de la RBSP et en mettant à l'écart le rbsp_stop_one_bit, qui est le dernier bit (celui de plus faible poids, le plus à droite) égal à 1, et en mettant à l'écart tous les bits suivants (de moindre poids, les plus à droite) qui le suivent, qui sont égaux à 0. Les données nécessaires pour le processus de décodage sont contenues dans la partie SODB de la charge utile RBSP.

emulation_prevention_three_byte est un octet égal à 0x03. Lorsqu'un octet emulation_prevention_three_byte est présent dans l'unité NAL, il doit être mis à l'écart par le processus de décodage.

Le dernier octet de l'unité NAL ne doit pas être égal à 0x00.

Au sein de l'unité NAL, les séquences de trois octets suivantes ne doivent pas survenir dans une position d'alignement d'octet:

– 0x000000

– 0x000001

– 0x000002

Au sein de l'unité NAL, toute séquence de quatre octets qui débute par 0x000003, autre que les séquences suivantes, ne doit survenir à aucune des positions d'alignement d'octet:

– 0x00000300

– 0x00000301

– 0x00000302

– 0x00000303

NOTE 6 – Lorsque nal_unit_type est égal à 0, un soin particulier doit toujours être pris dans la conception des codeurs afin d'éviter la présence des structures susmentionnées à trois octets et à quatre octets au début de la structure syntaxique d'unité de couche NAL, car les trois octets de prévention d'émulation d'élément syntaxique ne peuvent pas être le troisième octet d'une unité de couche NAL.

7.4.1.1 Encapsulation d'une chaîne SODB au sein d'une charge utile RBSP (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La forme de l'encapsulation d'une chaîne SODB au sein d'une charge utile RBSP et l'utilisation de emulation_prevention_three_byte pour l'encapsulation d'une charge utile RBSP au sein d'une unité NAL sont spécifiées pour les raisons suivantes:

– afin d'empêcher l'émulation de codes de démarrage au sein des unités NAL alors qu'on permet à toute chaîne SODB arbitraire d'être représentée au sein d'une unité NAL;

– afin de permettre l'identification de la fin de la chaîne SODB au sein de l'unité NAL en recherchant la charge utile RBSP pour le rbsp_stop_one_bit commençant à la fin de la charge utile RBSP;

– afin de permettre à une unité NAL d'avoir une taille plus grande que celle de la chaîne SODB dans certaines circonstances (en utilisant un ou plusieurs cabac_zero_word).

Le codeur peut produire une unité NAL à partir d'une charge utile RBSP par la procédure suivante:

les données de la charge utile RBSP sont recherchées dans les bits d'alignement d'octet des schémas binaires suivants:

'00000000 00000000 000000xx' (où xx représente tout schéma à 2 bit: 00, 01, 10 ou 11),

et un octet égal à 0x03 est inséré afin de remplacer ces schémas binaires par les schémas

'00000000 00000000 00000011 000000xx',

et finalement, lorsque le dernier octet des données de la charge utile RBSP est égal à 0x00 (ce qui ne peut survenir que lorsque la charge utile RBSP se termine par un cabac_zero_word), un octet final égal à 0x03 est ajouté à la fin des données.

La séquence d'octets résultante est alors mise en préfixe avec le premier octet de l'unité NAL contenant l'indication du type de structure de données de charge utile RBSP qu'il contient. Il en résulte la construction de l'unité NAL entière.

Ce processus peut permettre à toute chaîne SODB d'être représentée dans une unité NAL tout en s'assurant que:

– aucun préfixe de code de déclenchement aligné sur les octets n'est émulé au sein de l'unité NAL;

– aucune séquence de 8 bit de valeur zéro suivie d'un préfixe de code de déclenchement, sans considération de l'alignement d'octet, n'est émulée au sein de l'unité NAL.

7.4.1.2 Ordre des unités NAL et association aux images codées, unités d'accès et séquences vidéo

Le présent paragraphe spécifie les contraintes sur l'ordre des unités NAL dans le flux binaire. Tout ordre des unités NAL dans le flux binaire obéissant à ces contraintes est désigné dans le texte comme l'ordre de décodage des unités NAL. Au sein d'une unité NAL, la syntaxe des § 7.3, D.1 et E.1 spécifie l'ordre de décodage des éléments syntaxiques. Les décodeurs conformes à la présente Recommandation | Norme internationale doivent être capables de recevoir des unités NAL et leurs éléments syntaxiques dans l'ordre de décodage.

7.4.1.2.1 Ordre de séquence et charges utiles RBSP d'ensemble de paramètres d'image et leur activation

NOTE 1 – Le mécanisme de séquence et d'ensemble de paramètres d'image découple la transmission des informations peu changeantes de la transmission des données de macroblocs codés. Les ensembles de séquence et de paramètres d'image peuvent, dans certaines applications, être convoyés "hors bande" au moyen d'un mécanisme de transport fiable.

Une charge utile RBSP d'ensemble de paramètres d'image inclut des paramètres auxquels peuvent faire référence les unités NAL de tranche codée ou les unités NAL de subdivision A de données de tranche codée d'une ou plusieurs images codées. Chaque charge utile RBSP d'ensemble de paramètre d'image, au départ, est considérée comme n'étant pas active au début de l'opération du processus de décodage. Au plus, une charge utile RBSP d'ensemble de paramètres d'image est considérée comme étant active à un moment donné quelconque durant l'opération du processus de décodage, et l'activation de telle ou telle charge utile RBSP d'ensemble de paramètres d'image déclenche la désactivation de la charge utile RBSP d'ensemble de paramètres d'image précédemment active (le cas échéant).

Lorsqu'une charge utile RBSP d'ensemble de paramètres d'image (avec une valeur particulière de pic_parameter_set_id) n'est pas active et qu'elle est prise en charge en référence par une unité NAL de tranche codée ou une unité NAL de subdivision A de données de tranche codée (utilisant cette valeur de pic_parameter_set_id), elle est activée. Cette charge utile RBSP d'ensemble de paramètres d'image est appelée la charge utile active d'ensemble de paramètres d'image jusqu'à ce qu'elle soit désactivée par l'activation d'une autre charge utile RBSP d'ensemble de paramètres d'image. Une charge utile RBSP d'ensemble de paramètres d'image, avec cette valeur particulière de pic_parameter_set_id, doit être disponible pour le processus de décodage avant son activation.

Toute unité NAL d'ensemble de paramètres d'image contenant la valeur de pic_parameter_set_id pour la charge utile active d'ensemble de paramètres d'image doit avoir le même contenu que celui de la charge utile RBSP active de l'ensemble de paramètres d'image à moins qu'elle ne suive la dernière unité NAL de couche VCL d'une image codée et précède la première unité NAL de couche VCL d'une autre image codée.

Une charge utile RBSP d'ensemble de paramètres de séquence inclut des paramètres auxquels peuvent faire référence une ou plusieurs charges utiles RBSP d'ensemble de paramètres d'image ou une ou plusieurs unités NAL d'informations SEI contenant un message SEI de période de temporisation. Chaque charge utile RBSP d'ensemble de paramètre de séquence, au départ, est considérée comme n'étant pas active au début de l'opération du processus de décodage. Au plus, une charge utile RBSP d'ensemble de paramètres de séquence est considérée comme étant active à un moment donné quelconque durant l'opération du processus de décodage, et l'activation de telle ou telle charge utile RBSP d'ensemble de paramètres de séquence déclenche la désactivation de la charge utile RBSP d'ensemble de paramètres de séquence précédemment active (le cas échéant).

Lorsqu'une charge utile RBSP d'ensemble de paramètres de séquence (avec une valeur particulière de seq_parameter_set_id) n'est pas déjà active et qu'elle est prise en référence par l'activation d'une charge utile RBSP d'ensemble de paramètres d'image (utilisant cette valeur de seq_parameter_set_id) ou est prise en référence par une unité NAL d'informations SEI contenant un message SEI de période de temporisation (utilisant cette valeur de seq_parameter_set_id), elle est activée. Cette charge utile RBSP d'ensemble de paramètres de séquence est appelée la charge utile RBSP active d'ensemble de paramètres de séquence jusqu'à ce qu'elle soit désactivée par l'activation d'une autre charge utile RBSP d'ensemble de paramètres de séquence. Une charge utile RBSP d'ensemble de paramètres de séquence, avec cette valeur particulière de seq_parameter_set_id, doit être disponible pour le processus de décodage avant son activation. Une charge utile RBSP activée d'ensemble de paramètres de séquence doit rester active pour la séquence vidéo codée entière.

NOTE 2 – Etant donné qu'une unité d'accès IDR ouvre une nouvelle séquence vidéo codée et qu'une charge utile RBSP activée d'ensemble de paramètres de séquence doit rester active pour la séquence vidéo codée entière, une charge utile RBSP d'ensemble de paramètres de séquence ne peut être activée que par un message SEI de période de temporisation faisant partie d'une unité d'accès IDR.

Toute unité NAL d'ensemble de paramètres de séquence contenant la valeur de seq_parameter_set_id pour la charge utile RBSP active d'ensemble de paramètres de séquence doit avoir le même contenu que celui de la charge utile RBSP active d'ensemble de paramètres de séquence à moins qu'elle ne suive la dernière unité d'accès d'une séquence vidéo codée et ne précède la première unité NAL de couche VCL et la première unité NAL d'informations SEI contenant un message SEI de période de temporisation (le cas échéant) d'une autre séquence vidéo codée.

NOTE 3 – Si une charge utile RBSP d'ensemble de paramètres d'image ou une charge utile RBSP d'ensemble de paramètres de séquence est transportée au sein du flux binaire, ces contraintes imposent un ordre aux unités NAL qui contiennent la charge utile RBSP d'ensemble de paramètres d'image ou respectivement, la charge utile RBSP d'ensemble de paramètres de séquence. Sinon la charge utile RBSP d'ensemble de paramètres d'image ou la charge utile RBSP d'ensemble de paramètres de séquence est convoyée par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale), elle doit être disponible pour le processus de décodage à temps afin que ces contraintes puissent être respectées.

Lorsqu'elle est présente, une charge utile RBSP d'extension d'ensemble de paramètres de séquence comporte des paramètres qui utilisent une fonction similaire à celles d'une charge utile RBSP d'extension d'ensemble de paramètres de séquence. Aux fins de l'établissement de contraintes sur les éléments syntaxiques de la charge utile RBSP d'extension d'ensemble de paramètres de séquence et aux fins de la détermination de l'activation d'une charge utile RBSP d'extension d'ensemble de paramètres de séquence, la charge utile RBSP d'extension d'ensemble de paramètres de séquence doit être considérée comme faisant partie de la charge utile RBSP précédente d'extension d'ensemble de paramètres de séquence avec la même valeur de seq_parameter_set_id. Lorsqu'une charge utile RBSP d'extension d'ensemble de paramètres de séquence est présente mais n'est pas suivie par une charge utile RBSP d'extension d'ensemble de paramètres de séquence avec la même valeur de seq_parameter_set_id avant l'activation de la charge utile RBSP d'ensemble de paramètres de séquence, la charge utile RBSP d'extension d'ensemble de paramètres de séquence et ses éléments syntaxiques doivent être considérés comme absents pour la charge utile RBSP d'ensemble de paramètres de séquence active.

Toutes les contraintes qui sont organisées sur la relation entre les valeurs des éléments syntaxiques (et les valeurs des variables extraites de ces éléments syntaxiques) des ensembles de paramètres de séquence et des ensembles de paramètres d'image et d'autres éléments syntaxiques sont des expressions de contraintes qui s'appliquent uniquement à l'ensemble de paramètres de séquence actif et à l'ensemble de paramètres d'image actif. Si une charge utile RBSP d'ensemble de paramètres de séquence qui n'est pas activée dans le flux binaire est présente, ses éléments syntaxiques doivent avoir des valeurs qui seraient conformes aux contraintes spécifiées si elle était activée par référence dans un autre flux binaire conforme. Si une charge utile RBSP d'ensemble de paramètres d'image qui n'est pas toujours activée dans le flux binaire est présente, ses éléments syntaxiques doivent avoir des valeurs qui seraient conformes aux contraintes spécifiées si elle était activée par référence dans un autre flux binaire conforme.

Durant l'opération du processus de décodage (voir le paragraphe 8), les valeurs des paramètres de l'ensemble actif de paramètres d'image et de l'ensemble actif de paramètres de séquence doivent être considérées comme effectives. Pour l'interprétation des messages SEI, les valeurs des paramètres de l'ensemble de paramètres d'image et de l'ensemble de paramètres de séquence qui sont actifs pour l'opération du processus de décodage pour les unités NAL de couche VCL de l'image codée primaire dans la même unité d'accès, doivent être considérées comme effectives à moins qu'il n'en soit spécifié autrement dans la sémantique de message SEI.

7.4.1.2.2 Ordre des unités d'accès et association aux séquences vidéo codées

Un flux binaire conforme à la présente Recommandation | Norme internationale consiste en une ou plusieurs séquences vidéo codées.

Une séquence vidéo codée consiste en une ou plusieurs unités d'accès. L'ordre des unités NAL et des images codées et leur association aux unités d'accès sont décrits au § 7.4.1.2.3.

La première unité d'accès de chaque séquence vidéo codée est une unité d'accès IDR. Toutes les unités d'accès suivantes dans la séquence vidéo codée sont des unités d'accès non IDR.

Les valeurs de compte d'ordre d'image pour les images codées en unités d'accès consécutives dans l'ordre de décodage contenant des images qui ne sont pas de référence doivent être non décroissantes.

Lorsqu'elle est présente, une unité d'accès suivant une unité d'accès qui contient une fin de séquence d'unité NAL doit être une unité d'accès IDR.

Lorsqu'une unité NAL d'informations SEI contient des données qui appartiennent à plus d'une unité d'accès (par exemple, lorsque l'unité NAL d'informations SEI a une séquence vidéo codée comme domaine d'application), elle doit être contenue dans la première unité d'accès à laquelle elle s'applique.

Lorsqu'une fin de flux d'unité NAL est présente dans une unité d'accès, cette unité d'accès doit être la dernière unité d'accès dans le flux binaire et la fin de flux de l'unité NAL doit être la dernière unité NAL dans cette unité d'accès.

7.4.1.2.3 Ordre des unités NAL et images codées et association aux unités d'accès

Une unité d'accès consiste en une image codée primaire, zéro ou plusieurs images codées redondantes correspondantes, et zéro ou plusieurs unités NAL non VCL. L'association des unités NAL de couche VCL aux images codées redondantes ou primaires est décrite au § 7.4.1.2.5.

La première unité d'accès contenue dans le flux binaire commence à la première unité de couche NAL du flux binaire.

La première de toutes les unités NAL suivantes après la dernière unité NAL de couche VCL d'une image codée primaire spécifie le début d'une nouvelle unité d'accès.

– unité NAL de délimiteur d'unité d'accès (le cas échéant);

– unité NAL d'ensemble de paramètres de séquence (le cas échéant);

– unité NAL d'ensemble de paramètres d'image (le cas échéant);

– unité NAL d'informations SEI (le cas échéant);

– des unités NAL avec nal_unit_type dans la gamme de 14 à 18 inclus;

– première unité NAL de couche VCL d'une image codée primaire (toujours présente).

Les contraintes pour la détection de la première unité NAL de couche VCL d'une image codée primaire sont spécifiées au § 7.4.1.2.4.

Les contraintes suivantes doivent être respectées par l'ordre des images codées et des unités NAL non VCL au sein d'une unité d'accès.

– Lorsqu'une unité NAL de délimiteur d'unité d'accès est présente, elle doit être la première unité NAL. Il doit y avoir au plus une unité NAL de délimiteur d'unité d'accès dans chaque unité d'accès.

– Lorsqu'une unité NAL d'informations SEI est présente, elle doit précéder l'image codée primaire.

– Lorsqu'une unité NAL d'informations SEI contenant un message SEI de période de temporisation est présente, le message SEI de période de temporisation doit être la première charge utile de message SEI de la première unité NAL d'informations SEI dans l'unité d'accès.

– L'image codée primaire doit précéder les images codées redondantes correspondantes.

– Lorsque les images codées redondantes sont présentes, elles doivent être ordonnées en ordre ascendant de la valeur de redundant_pic_cnt.

– Lorsqu'une unité NAL d'extension d'ensemble de paramètres de séquence est présente, elle doit être la prochaine unité de couche NAL après une unité NAL d'ensemble de paramètres de séquence qui utilise la même valeur de seq_parameter_set_id que dans l'unité NAL d'extension d'ensemble de paramètres de séquence.

– Lorsqu'une ou plusieurs tranches codées d'une image codée auxiliaire sans unités NAL de subdivision sont présentes, elles doivent suivre l'image codée primaire et toutes les images codées redondantes (le cas échéant).

– Lorsqu'une unité NAL de fin de séquence est présente, elle doit suivre l'image codée primaire et toutes les images codées redondantes (le cas échéant) et toutes les tranches codées d'une image codée auxiliaire sans unités NAL de subdivision (le cas échéant).

– Lorsqu'une unité NAL de fin de flux est présente, elle doit être la dernière unité NAL.

– Des unités NAL ayant nal_unit_type égal à 0, 12 ou dans la gamme de 20 à 31 inclus, ne doivent pas précéder la première unité NAL de couche VCL de l'image codée primaire.

NOTE 1 – Les unités NAL d'ensemble de paramètres de séquence ou d'ensemble de paramètres d'image peuvent être présentes dans une unité d'accès, mais ne peuvent pas suivre la dernière unité NAL de couche VCL de l'image codée primaire au sein de l'unité d'accès, car cette condition spécifierait le début d'une nouvelle unité d'accès.

NOTE 2 – Lorsqu'une unité NAL ayant nal_unit_type égal à 7 ou 8 est présente dans une unité d'accès, elle peut ne pas être référencée dans les images codées de l'unité d'accès dans laquelle elle est présente, et peut être référencée dans des images codées d'unités d'accès ultérieures.

La structure des unités d'accès ne contenant aucune unité NAL avec nal_unit_type égal à 0, 7, 8 ou dans la gamme de 12 à 18 inclus ou dans la gamme de 20 à 31 inclus, est indiquée à la Figure 7-1.

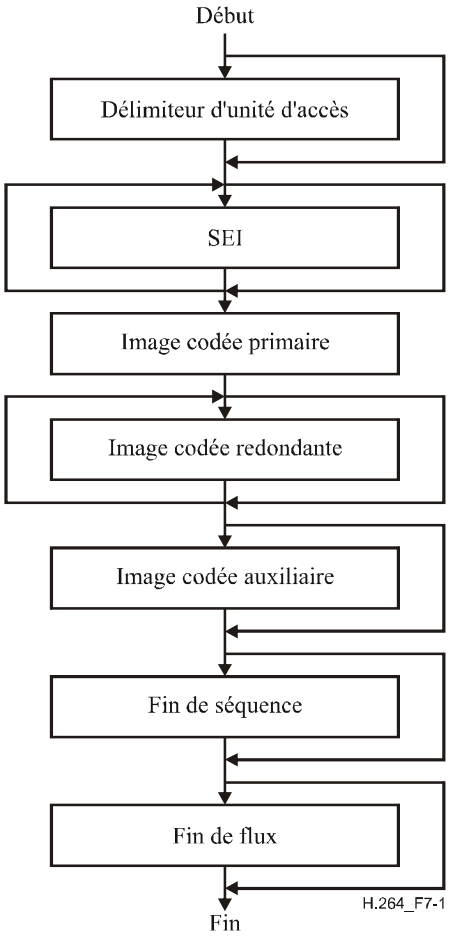


Figure 7-1 – Structure d'une unité d'accès ne contenant aucune unité NAL avec nal_unit_type égal à 0, 7, 8 ou dans la gamme de 12 à 18 inclus, ou dans la gamme de 20 à 31 inclus

7.4.1.2.4 Détection de la première unité NAL de couche VCL d'une image codée primaire

Le présent paragraphe spécifie les contraintes applicables à la syntaxe d'une unité NAL de couche VCL qui sont suffisantes afin de permettre la détection de la première unité NAL de couche VCL de chaque image codée primaire.

Toute unité NAL de tranche codée ou unité NAL de subdivision A de données de tranche codée de l'image codée primaire de l'unité d'accès en cours doit être différente de toute unité NAL de tranche codée ou unité NAL de subdivision A de données de tranche codée de l'image codée primaire de l'unité d'accès précédente d'une ou plusieurs des façons suivantes.

– frame_num diffère en valeur. La valeur de frame_num utilisée afin de tester cette condition est la valeur de frame_num qui apparaît dans la syntaxe de l'en-tête de tranche, indépendamment de savoir s'il ressort que cette

valeur a été égale à 0 pour utilisation ultérieure dans le processus de décodage en raison de la présence de l'opération de commande de gestion de mémoire égale à 5.

NOTE 1 – Une des conséquences de l'énoncé ci-dessus est qu'une image codée primaire dont frame_num est égal à 1 ne peut pas contenir une opération memory_management_control_operation égale à 5 à moins qu'une des autres conditions énumérées ci-dessous ne soit remplie pour la prochaine image codée primaire qui la suit (le cas échéant).

– pic_parameter_set_id diffère en valeur.

– field_pic_flag diffère en valeur.

– bottom_field_flag est présent dans les deux et diffère en valeur.

– nal_ref_idc diffère en valeur avec une des valeurs nal_ref_idc égale à 0.

– pic_order_cnt_type est égal à 0 pour les deux et soit pic_order_cnt_lsb diffère en valeur, soit delta_pic_order_cnt_bottom diffère en valeur.

– pic_order_cnt_type est égal à 1 pour les deux et soit delta_pic_order_cnt[0] diffère en valeur, soit delta_pic_order_cnt[1] diffère en valeur.

– nal_unit_type diffère en valeur avec une des valeur nal_unit_type égale à 5.

– nal_unit_type est égal à 5 pour les deux et idr_pic_id diffère en valeur.

NOTE 2 – Certaines des unités NAL de couche VCL dans les images codées redondantes ou certaines unités NAL non VCL (par exemple, une unité NAL de délimiteur d'unité d'accès) peuvent aussi être utilisées pour la détection des frontières entre les unités d'accès, et peuvent donc aider à la détection du début d'une nouvelle image codée primaire.

7.4.1.2.5 Ordre des unités NAL de couche VCL et association aux images codées

Chaque unité NAL de couche VCL fait partie d'une image codée.

L'ordre des unités NAL de couche VCL au sein d'uneimage à rafraîchissement IDR codée est obligatoirement le suivant.

– Si un ordre de tranche arbitraire est admis comme spécifié à l'Annexe A, la tranche codée des unités NAL d'uneimage à rafraîchissement IDR peut avoir n'importe quel ordre par rapport à chacune des autres.

– Sinon (si l'ordre arbitraire de tranche n'est pas admis), l'ordre de tranche codée des unités NAL d'uneimage à rafraîchissement IDR doit être dans l'ordre croissant des adresses de macrobloc pour le premier macrobloc de chaque tranche codée d'unité NAL d'uneimage à rafraîchissement IDR.

L'ordre des unités NAL de couche VCL au sein d'une image codée non IDR est obligatoirement le suivant.

– Si l'ordre arbitraire de tranche est admis comme spécifié à l'Annexe A, la tranche codée des unités NAL d'image non IDR ou des unités NAL de subdivision A de données de tranche codée peut avoir n'importe quel ordre par rapport à chacune des autres. Une unité NAL de subdivision A de données de tranche codée avec une valeur particulière de slice_id doit précéder chaque unité NAL de subdivision B de données de tranche codée présente avec la même valeur de slice_id. Une unité NAL de subdivision A de données de tranche codée avec une valeur particulière de slice_id doit précéder toute unité NAL de subdivision C de données de tranche codée présente avec la même valeur de slice_id. Lorsqu'une unité NAL de subdivision B de données de tranche codée avec une valeur particulière de slice_id est présente, elle doit précéder toute unité NAL de subdivision C de données de tranche codée avec la même valeur de slice_id.

– Sinon (si l'ordre arbitraire de tranche n'est pas admis), l'ordre de tranche codée des unités NAL d'image non IDR ou des unités NAL de subdivision A de données de tranche codée doit être l'ordre croissant des adresses de macrobloc pour le premier macrobloc de chaque tranche codée d'une unité NAL d'image non IDR ou d'une unité NAL de subdivision A de données de tranche codée. Une unité NAL avec une valeur particulière de slice_id doit immédiatement précéder toute unité NAL de subdivision B de données de tranche codée avec la même valeur de slice_id. Une unité NAL de subdivision A de données de tranche codée avec une valeur particulière de slice_id doit immédiatement précéder toute unité NAL de subdivision A de données de tranche codée présente avec la même valeur de slice_id, lorsque n'est pas présente une unité NAL de subdivision B de données de tranche codée avec la même valeur de slice_id. Lorsqu'est présente une unité NAL de subdivision B de données de tranche codée avec une valeur particulière de slice_id, elle doit immédiatement précéder toute unité NAL de subdivision C de données de tranche codée avec la même valeur de slice_id.

Les unités NAL ayant nal_unit_type égal à 12 peuvent être présentes dans l'unité d'accès mais elles ne doivent pas précéder la première unité NAL de couche VCL de l'image codée primaire au sein de l'unité d'accès.

Les unités NAL ayant nal_unit_type égal à 0 ou dans la gamme de 24 à 31 inclus, qui ne sont pas spécifiés, peuvent être présentes dans une unité d'accès mais ne doivent pas précéder la première unité NAL de couche VCL de l'image codée primaire au sein de l'unité d'accès.

Les unités NAL ayant nal_unit_type dans la gamme de 20 à 23 inclus, qui sont réservés, ne doivent pas précéder la première unité NAL de couche VCL de l'image codée primaire au sein de l'unité d'accès (lorsqu'elle sera spécifiée dans l'avenir par l'UIT-T | ISO/CEI).

7.4.2 Charges utiles de séquence d'octets bruts et sémantique des bits de traîne RBSP

7.4.2.1 Sémantique de la charge utile RBSP d'ensemble de paramètres de séquence

profile_idc et **level_idc** indiquent le profil et le niveau auquel se conforme le flux binaire, comme spécifié à l'Annexe A.

constraint_set0_flag égal à 1 indique que le flux binaire obéit à toutes les contraintes spécifiées au § A.2.1. constraint_set0_flag égal à 0 indique que le flux binaire peut obéir ou non à toutes les contraintes spécifiées au § A.2.1.

constraint_set1_flag égal à 1 indique que le flux binaire obéit à toutes les contraintes spécifiées au § A.2.2. constraint_set1_flag égal à 0 indique que le flux binaire peut ou non obéir à toutes les contraintes spécifiées au § A.2.2.

constraint_set2_flag égal à 1 indique que le flux binaire obéit à toutes les contraintes spécifiées au § A.2.3. constraint_set2_flag égal à 0 indique que le flux binaire peut ou non obéir à toutes les contraintes spécifiées au § A.2.3.

NOTE 1 – Lorsqu'un ou plusieurs des fanions constraint_set0_flag, constraint_set1_flag, ou constraint_set2_flag sont égaux à 1, le flux binaire doit impérativement respecter les contraintes de tous les paragraphes indiqués du § A.2. Lorsque profile_idc est égal à 100, 110, 122, ou 144, les valeurs de constraint_set0_flag, constraint_set1_flag, et constraint_set2_flag doivent impérativement être toutes égales à 0.

constraint_set3_flag indique ce qui suit.

– Si profile_idc est égal à 66, 77, ou 88 et si level_idc est égal à 11, constraint_set3_flag égal à 1 indique que le flux binaire respecte toutes les contraintes spécifiées dans l'Annexe A pour level 1b et constraint_set3_flag égal à 0 indique que le flux binaire peut, le cas échéant, respecter toutes les contraintes spécifiées dans l'Annexe A pour level 1b.

– Sinon (si profile_idc est égal à 100, 110, 122, ou 144 ou si level_idc n'est pas égal à 11), la valeur de 1 pour constraint_set3_flag est réservée pour utilisation future par l'UIT-T | ISO/CEI. constraint_set3_flag doit être égal à 0 dans les flux binaires conformes à la présente Recommandation | Norme internationale lorsque profile_idc est égal à 100, 110, 122, ou 144 ou lorsque level_idc n'est pas égal à 11. Les décodeurs conformes à la présente Recommandation | Norme internationale doivent ignorer la valeur de constraint_set3_flag lorsque profile_idc est égal à 100, 110, 122, ou 144 ou level_idc n'est pas égal à 11.

reserved_zero_4bits doit être égal à 0. D'autres valeurs de reserved_zero_4bits pourront être spécifiées ultérieurement par l'UIT-T | ISO/CEI. Les décodeurs doivent ignorer la valeur de reserved_zero_4bits.

seq_parameter_set_id identifie les ensembles de paramètres de séquence auxquels se rapporte l'ensemble de paramètres d'image. La valeur de seq_parameter_set_id doit être dans la gamme de 0 à 31 inclus.

NOTE 2 – Si cela est faisable, les codeurs devraient utiliser des valeurs distinctes de seq_parameter_set_id lorsque les valeurs des autres éléments syntaxiques d'ensembles de paramètres de séquence diffèrent, plutôt que de changer les valeurs des éléments syntaxiques associés à une valeur spécifique de seq_parameter_set_id.

chroma_format_idc spécifie l'échantillonnage chroma relatif à l'échantillonnage luma comme spécifié dans le § 6.2. La valeur de chroma_format_idc doit être dans l'étendue de 0 à 3, inclus. Lorsque chroma_format_idc n'est pas présent, sa valeur doit être supposée égale à 1 (format chromatique 4:2:0).

residual_colour_transform_flag égal à 1 spécifie que la transformée chromatique résiduelle est appliquée comme spécifié dans le § 8.5. residual_colour_transform_flag égal à 0 spécifie que la transformée chromatique résiduelle n'est pas appliquée. Lorsque le fanion residual_colour_transform_flag n'est pas présent, sa valeur doit être supposée égale à 0.

bit_depth_luma_minus8 spécifie la profondeur de bits des échantillons de la série matricielle d'échantillons luma et la valeur du décalage d'étendue paramétrique de quantification luma $QpBdOffset_Y$, comme spécifié par:

$$BitDepth_Y = 8 + bit\_depth\_luma\_minus8 \quad (7\text{-}1)$$

$$QpBdOffset_Y = 6 * bit\_depth\_luma\_minus8 \quad (7\text{-}2)$$

Lorsque bit_depth_luma_minus8 n'est pas présent, sa valeur doit être supposée égale à 0. bit_depth_luma_minus8 doit être dans l'étendue de 0 à 4, inclus.

bit_depth_chroma_minus8 spécifie la profondeur de bits des échantillons de la série matricielle d'échantillons chroma et la valeur du décalage d'étendue paramétrique de quantification chroma $QpBdOffset_C$, comme spécifié par:

$$BitDepthC = 8 + bit\_depth\_chroma\_minus8 \quad (7\text{-}3)$$

$$QpBdOffsetC = 6 * (bit\_depth\_chroma\_minus8 + residual\_colour\_transform\_flag) \quad (7\text{-}4)$$

Lorsque bit_depth_chroma_minus8 n'est pas présent, sa valeur doit être supposée égale à 0. bit_depth_chroma_minus8 doit être dans l'étendue de 0 à 4, inclus.

La variable RawMbBits est calculée comme suit:

$$RawMbBits = 256 * BitDepth_Y + 2 * MbWidthC * MbHeightC * BitDepth_C \quad (7\text{-}5)$$

qpprime_y_zero_transform_bypass_flag égal à 1 spécifie que, lorsque QP'_Y est égal à 0, une opération de contournement de transformation doit être appliquée comme spécifié dans le § 8.5 pour le processus de décodage des coefficients de transformée et pour le processus de construction d'image avant processus de filtre de déblocage. qpprime_y_zero_transform_bypass_flag égal à 0 spécifie que le processus de décodage des coefficients de transformée et le processus de construction d'image avant processus de filtre de déblocage ne doivent pas utiliser l'opération de contournement de transformation. Lorsque qpprime_y_zero_transform_bypass_flag n'est pas présent, sa valeur doit être supposée égale à 0.

seq_scaling_matrix_present_flag égal à 1 spécifie que les fanions seq_scaling_list_present_flag[i] pour i = 0..7 sont présents. seq_scaling_matrix_present_flag égal à 0 spécifie que ces fanions ne sont pas présents et que la liste de normalisation au niveau de la séquence spécifiée par Flat_4x4_16 doit être présumée pour i = 0..5 et que la liste de normalisation au niveau de la séquence spécifiée par Flat_8x8_16 doit être présumée pour i = 6..7. Lorsque le fanion seq_scaling_matrix_present_flag n'est pas présent, sa valeur doit être supposée égale à 0.

Les listes de normalisation Flat_4x4_16 et Flat_8x8_16 sont spécifiées comme suit:

$$Flat\_4x4\_16[i] = 16, \quad \text{avec } i = 0..15, \quad (7\text{-}6)$$

$$Flat\_8x8\_16[i] = 16, \quad \text{avec } i = 0..63. \quad (7\text{-}7)$$

seq_scaling_list_present_flag[i **]** égal à 1 spécifie que la structure syntaxique pour la liste de normalisation i est présente dans l'ensemble de paramètres de séquence. seq_scaling_list_present_flag[i] égal à 0 spécifie que la structure syntaxique pour la liste de normalisation i n'est pas présente dans l'ensemble de paramètres de séquence et que l'ensemble A de règles de repli vers une liste de normalisation, spécifié dans le Tableau 7-2, doit être utilisé afin de présumer la liste de normalisation au niveau de la séquence pour l'indice i.

Tableau 7-2 – Allocation de noms mnémoniques aux indices de liste de normalisation et spécification d'une règle de repli

Indice de valeur de liste de normali-sation	Nom mnémonique	Taille de bloc	Type de prédiction de MB	Composante	Ensemble A de règles de repli vers une liste de normalisation	Ensemble B de règles de repli vers une liste de normalisation	Liste de normalisation par défaut
0	Sl_4x4_Intra_Y	4x4	Intra	Y	liste de normalisation par défaut	liste de normalisation au niveau de la séquence	Default_4x4_Intra
1	Sl_4x4_Intra_Cb	4x4	Intra	Cb	liste de normalisation pour i = 0	liste de normalisation pour i = 0	Default_4x4_Intra
2	Sl_4x4_Intra_Cr	4x4	Intra	Cr	liste de normalisation pour i = 1	liste de normalisation pour i = 1	Default_4x4_Intra
3	Sl_4x4_Inter_Y	4x4	Inter	Y	liste de normalisation par défaut	liste de normalisation au niveau de la séquence	Default_4x4_Inter
4	Sl_4x4_Inter_Cb	4x4	Inter	Cb	liste de normalisation pour i = 3	liste de normalisation pour i = 3	Default_4x4_Inter
5	Sl_4x4_Inter_Cr	4x4	Inter	Cr	liste de normalisation pour i = 4	liste de normalisation pour i = 4	Default_4x4_Inter
6	Sl_8x8_Intra_Y	8x8	Intra	Y	liste de normalisation par défaut	liste de normalisation au niveau de la séquence	Default_8x8_Intra
7	Sl_8x8_Inter_Y	8x8	Inter	Y	liste de normalisation par défaut	liste de normalisation au niveau de la séquence	Default_8x8_Inter

Le Tableau 7-3 spécifie les listes de normalisation par défaut Default_4x4_Intra et Default_4x4_Inter. Le Tableau 7-4 spécifie les listes de normalisation par défaut Default_8x8_Intra et Default_8x8_Inter.

Tableau 7-3 – Spécification des listes de normalisation par défaut Default_4x4_Intra et Default_4x4_Inter

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Default_4x4_Intra[idx]	6	13	13	20	20	20	28	28	28	28	32	32	32	37	37	42
Default_4x4_Inter[idx]	10	14	14	20	20	20	24	24	24	24	27	27	27	30	30	34

Tableau 7-4 – Spécification des listes de normalisation par défaut Default_8x8_Intra et Default_8x8_Inter

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Default_8x8_Intra[idx]	6	10	10	13	11	13	16	16	16	16	18	18	18	18	18	23
Default_8x8_Inter[idx]	9	13	13	15	13	15	17	17	17	17	19	19	19	19	19	21

Tableau 7-4 (suite) – Spécification des listes de normalisation par défaut Default_8x8_Intra et Default_8x8_Inter

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Default_8x8_Intra[idx]	23	23	23	23	23	25	25	25	25	25	25	25	27	27	27	27
Default_8x8_Inter[idx]	21	21	21	21	21	22	22	22	22	22	22	22	24	24	24	24

Tableau 7-4 (suite) – Spécification des listes de normalisation par défaut Default_8x8_Intra et Default_8x8_Inter

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Default_8x8_Intra[idx]	27	27	27	27	29	29	29	29	29	29	29	31	31	31	31	31
Default_8x8_Inter[idx]	24	24	24	24	25	25	25	25	25	25	25	27	27	27	27	27

Tableau 7-4 (fin) – Spécification des listes de normalisation par défaut Default_8x8_Intra et Default_8x8_Inter

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Default_8x8_Intra[idx]	31	33	33	33	33	33	36	36	36	36	38	38	38	40	40	42
Default_8x8_Inter[idx]	27	28	28	28	28	28	30	30	30	30	32	32	32	33	33	35

log2_max_frame_num_minus4 spécifie la valeur de la variable MaxFrameNum qui est utilisée comme suit dans les calculs de frame_num qui s'y rapportent:

$$\text{MaxFrameNum} = 2^{(\text{log2\_max\_frame\_num\_minus4} + 4)} \tag{7-8}$$

La valeur de log2_max_frame_num_minus4 doit être dans la gamme de 0 à 12 inclus.

pic_order_cnt_type spécifie la méthode de décodage du compte d'ordre d'image (comme spécifié au § 8.2.1). La valeur de pic_order_cnt_type doit être dans la gamme de 0 à 2 inclus.

pic_order_cnt_type ne doit pas être égal à 2 dans une séquence vidéo codée qui contient un des éléments suivants:

- une unité d'accès contenant une trame non référentielle suivie immédiatement par une unité d'accès contenant une image non référentielle;
- deux unités d'accès contenant chacune une sous-trame avec les deux sous-trames formant ensemble une paire complémentaire de sous-trames non référentielles suivie immédiatement par une unité d'accès contenant une image non référentielle;
- une unité d'accès contenant une sous-trame non référentielle suivi immédiatement par une unité d'accès contenant une autre image non référentielle qui ne forme pas une paire complémentaire de sous-trames non référentielles avec la première des deux unités d'accès.

log2_max_pic_order_cnt_lsb_minus4 spécifie la valeur de la variable MaxPicOrderCntLsb qui est utilisée comme suit dans le processus de décodage pour le compte d'ordre d'image comme spécifié au § 8.2.1:

$$\text{MaxPicOrderCntLsb} = 2^{(\text{log2\_max\_pic\_order\_cnt\_lsb\_minus4} + 4)} \tag{7-9}$$

La valeur de log2_max_pic_order_cnt_lsb_minus4 doit être dans la gamme de 0 à 12 inclus.

delta_pic_order_always_zero_flag égal à 1 spécifie que delta_pic_order_cnt[0] et delta_pic_order_cnt[1] ne sont pas présents dans les en-têtes de tranche de la séquence et doivent être supposés égaux à 0. delta_pic_order_always_zero_flag égal à 0 spécifie que delta_pic_order_cnt[0] est présent dans les en-têtes de tranche de la séquence et delta_pic_order_cnt[1] peut être présent dans les en-têtes de tranche de la séquence.

offset_for_non_ref_pic est utilisé afin de calculer le compte d'ordre d'image d'une image non référentielle comme spécifié au § 8.2.1. La valeur de offset_for_non_ref_pic doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

offset_for_top_to_bottom_field est utilisé afin de calculer le compte d'ordre d'image d'une sous-trame inférieure comme spécifié au § 8.2.1. La valeur de offset_for_top_to_bottom_field doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

num_ref_frames_in_pic_order_cnt_cycle est utilisé dans le processus de décodage pour le compte d'ordre d'image comme spécifié au § 8.2.1. La valeur de num_ref_frames_in_pic_order_cnt_cycle doit être dans la gamme de 0 à 255 inclus.

offset_for_ref_frame[i **]** est un élément d'une liste de valeurs de num_ref_frames_in_pic_order_cnt_cycle utilisées dans le processus de décodage pour le compte d'ordre d'image comme spécifié au § 8.2.1. La valeur de offset_for_ref_frame[i] doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

num_ref_frames spécifie le nombre maximal de trames de référence à court terme et à long terme, de paires de sous-trames de référence complémentaires et de sous-trames de référence non appariées qui peuvent être utilisées par le processus de décodage pour l'interprédiction de toute image dans la séquence. num_ref_frames détermine aussi la taille de l'opération de fenêtre glissante, comme spécifié au § 8.2.5.3. La valeur de num_ref_frames doit être dans la gamme de 0 à MaxDpbSize (comme spécifié au § A.3.1 ou A.3.2) inclus.

gaps_in_frame_num_value_allowed_flag spécifie les valeurs admises de frame_num comme spécifié au § 7.4.3 et le processus de décodage dans le cas de la déduction d'une lacune dans les valeurs de frame_num comme spécifié au § 8.2.5.2.

pic_width_in_mbs_minus1 plus 1 spécifie la largeur de chaque image décodée en unités de macroblocs.

La variable pour la largeur d'image en unités de macroblocs est déduite comme suit:

$$PicWidthInMbs = pic\_width\_in\_mbs\_minus1 + 1 \quad (7\text{-}10)$$

La variable pour la largeur d'image pour le composante luma est déduite comme suit:

$$PicWidthInSamples_L = PicWidthInMbs * 16 \quad (7\text{-}11)$$

La variable pour la largeur d'image pour les composantes chroma est déduite comme suit:

$$PicWidthInSamples_C = PicWidthInMbs * 8MbWidthC \quad (7\text{-}12)$$

pic_height_in_map_units_minus1 plus 1 spécifie la hauteur en unités de mapagge de groupe de tranches d'une trame ou sous-trame décodé.

Les variables PicHeightInMapUnits et PicSizeInMapUnits sont déduites comme suit:

$$PicHeightInMapUnits = pic\_height\_in\_map\_units\_minus1 + 1 \quad (7\text{-}13)$$

$$PicSizeInMapUnits = PicWidthInMbs * PicHeightInMapUnits \quad (7\text{-}14)$$

frame_mbs_only_flag égal à 0 spécifie que les images codées de la séquence vidéo codée peuvent être soit des sous-trames codées soit des trames codées. frame_mbs_only_flag égal à 1 spécifie que chaque image codée de la séquence vidéo codée est une trame codée ne contenant que des macroblocs de trame.

La gamme admise de valeurs pour pic_width_in_mbs_minus1, pic_height_in_map_units_minus1, et frame_mbs_only_flag est spécifiée par des contraintes à l'Annexe A.

En fonction de frame_mbs_only_flag, la sémantique est attribuée comme suit à pic_height_in_map_units_minus1.

– Si frame_mbs_only_flag est égal à 0, pic_height_in_map_units_minus1 plus 1 est la hauteur d'une sous-trame en unités de macroblocs.

– Sinon (si frame_mbs_only_flag est égal à 1), pic_height_in_map_units_minus1 plus 1 est la hauteur d'une trame en unités de macroblocs.

La variable FrameHeightInMbs est déduite comme suit:

$$FrameHeightInMbs = (2 - frame\_mbs\_only\_flag) * PicHeightInMapUnits \quad (7\text{-}15)$$

mb_adaptive_frame_field_flag égal à 0 spécifie qu'il n'y a pas de commutation entre macroblocs de trame et de sous-trame au sein d'une image. mb_adaptive_frame_field_flag égal à 1 spécifie l'utilisation possible d'une commutation entre macroblocs de trame et de sous-trame au sein des trames. Lorsque mb_adaptive_frame_field_flag n'est pas présent, on doit déduire qu'il est égal à 0.

direct_8x8_inference_flag spécifie la méthode utilisée dans le processus de déduction pour les vecteurs cinétiques luma pour B_Skip, B_Direct_16x16 et B_Direct_8x8 comme spécifié au § 8.4.1.2. Lorsque frame_mbs_only_flag est égal à 0, direct_8x8_inference_flag doit être égal à 1.

frame_cropping_flag égal à 1 spécifie que les paramètres de décalage de cadrage de trame viennent à la suite dans l'ensemble de paramètres de séquence. frame_cropping_flag égal à 0 spécifie que les paramètres de décalage de cadrage de trame ne sont pas présents.

frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, frame_crop_bottom_offset spécifient les échantillons des images contenues dans la séquence vidéo codée qui sont issus du processus de décodage, sous la forme d'une région rectangulaire spécifiée en coordonnées de trame avant sortie.

Les variables CropUnitX et CropUnitY sont calculées comme suit:

– Si chroma_format_idc est égal à 0, CropUnitX et CropUnitY sont des variables calculées comme suit:

$$CropUnitX = 1 \quad (7\text{-}16)$$
$$CropUnitY = 2 - frame\_mbs\_only\_flag \quad (7\text{-}17)$$

– Sinon (si chroma_format_idc est égal à 1, 2, ou 3), CropUnitX et CropUnitY sont des variables calculées comme suit:

$$CropUnitX = SubWidthC \quad (7\text{-}18)$$
$$CropUnitY = SubHeightC * (2 - frame\_mbs\_only\_flag) \quad (7\text{-}19)$$

Le rectangle de recadrage de trame contient des échantillons luma avec des coordonnées horizontales de trame allant de CropUnitX * frame_crop_left_offset à $PicWidthInSamples_L$ – (CropUnitX * frame_crop_right_offset + 1) et les coordonnées verticales de trame allant de CropUnitY * frame_crop_top_offset à (16 * FrameHeightInMbs) – (CropUnitY * frame_crop_bottom_offset + 1), inclus. La valeur de frame_crop_left_offset doit être dans l'étendue de 0 à ($PicWidthInSamples_L$ / CropUnitX) – (frame_crop_right_offset + 1), inclus; et la valeur de frame_crop_top_offset doit être dans l'étendue de 0 à (16 * FrameHeightInMbs / CropUnitY) – (frame_crop_bottom_offset + 1), inclus.

Lorsque frame_cropping_flag est égal à 0, les valeurs de frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, et frame_crop_bottom_offset doivent être présumées égales à 0.

Lorsque chroma_format_idc n'est pas égal à 0, les échantillons correspondants qui ont été spécifiés des deux séries matricielles d'échantillons chroma sont les échantillons qui utilisent les coordonnées de trame (x / SubWidthC, y / SubHeightC), où (x, y) sont les coordonnées de trame des échantillons luma spécifiés.

Pour les sous-trames décodées, les échantillons spécifiés des sous-trames décodées sont les échantillons qui tombent dans les rectangles spécifiés dans les coordonnées de trame.

vui_parameters_present_flag égal à 1 spécifie que la structure syntaxique vui_parameters( ) telle que spécifiée à l'Annexe E est présente. vui_parameters_present_flag égal à 0 spécifie que la structure syntaxique vui_parameters( ) telle que spécifiée à l'Annexe E n'est pas présente.

7.4.2.1.1 Sémantique de liste de normalisation

delta_scale sert à calculer le jième élément de la liste de normalisation pour j dans l'étendue de 0 à sizeOfScalingList – 1, inclus. La valeur de delta_scale doit être dans l'étendue de –128 à +127, inclus.

Lorsque useDefaultScalingMatrixFlag est calculé comme étant égal à 1, la liste de normalisation doit être présumée égale à la liste de normalisation par défaut comme spécifié dans le Tableau 7-2.

7.4.2.1.2 Sémantique de charge utile RBSP d'extension d'ensemble de paramètres de séquence

seq_parameter_set_id identifie l'ensemble de paramètres de séquence associé à l'extension d'ensemble de paramètres de séquence. La valeur de seq_parameter_set_id doit être dans l'étendue de 0 à 31, inclus.

aux_format_idc égal à 0 indique qu'il n'y a pas d'image codée auxiliaire dans la séquence vidéo codée. Aux_format_idc égal à 1 indique qu'exactement une seule image codée auxiliaire est présente dans chaque unité d'accès de la séquence vidéo codée, et qu'aux fins de la fusion alpha les échantillons décodés de l'image codée primaire associée dans chaque unité d'accès devraient être multipliés par les valeurs d'échantillon d'interprétation de l'image codée auxiliaire contenue dans l'unité d'accès au cours du processus d'affichage à la sortie du processus de décodage. Aux_format_idc égal à 2 indique qu'exactement une seule image codée auxiliaire existe dans chaque unité d'accès de la séquence vidéo codée, et qu'aux fins de la fusion alpha les échantillons décodés de l'image codée primaire associée dans chaque unité d'accès ne devraient pas être multipliés par les valeurs d'échantillon d'interprétation de l'image codée auxiliaire contenue dans l'unité d'accès au cours du processus d'affichage à la sortie du processus de décodage. Aux_format_idc égal à 3 indique qu'exactement une seule image codée auxiliaire existe dans chaque unité d'accès de la séquence vidéo codée, et que l'usage des images codées auxiliaires n'est pas spécifié. La valeur de aux_format_idc doit être dans l'étendue de 0 à 3, inclus. Les valeurs supérieures à 3 de aux_format_idc sont réservées à l'indication de la présence d'exactement une seule image codée auxiliaire dans chaque unité d'accès de la séquence vidéo codée à des fins qui seront spécifiées ultérieurement par l'UIT-T | ISO/CEI. Lorsque aux_format_idc n'est pas présent, sa valeur doit être supposée égale à 0.

NOTE 1 – Les décodeurs conformes à la présente Recommandation | Norme internationale ne sont pas tenus de décoder les images codées auxiliaires.

bit_depth_aux_minus8 spécifie la profondeur de bits des échantillons de la série matricielle d'échantillons de l'image codée auxiliaire. bit_depth_aux_minus8 doit être dans l'étendue de 0 à 4, inclus.

alpha_incr_flag égal à 0 indique que la valeur d'échantillon d'interprétation pour chaque valeur d'échantillon décodé d'image codée auxiliaire est égale à la valeur d'échantillon décodé d'image codée auxiliaire aux fins de fusion alpha. alpha_incr_flag égal à 1 indique que, aux fins de fusion alpha, après décodage des échantillons de l'image codée auxiliaire, toute valeur d'échantillon d'image codée auxiliaire qui est plus grande que Min(alpha_opaque_value, alpha_transparent_value) devrait être augmentée d'une unité afin d'obtenir la valeur d'échantillon d'interprétation pour l'échantillon d'image codée auxiliaire, et toute valeur d'échantillon d'image codée auxiliaire qui est inférieure ou égale à Min(alpha_opaque_value, alpha_transparent_value) devrait être utilisée sans altération en tant que valeur d'échantillon d'interprétation pour la valeur d'échantillon décodé d'image codée auxiliaire.

alpha_opaque_value spécifie la valeur d'échantillon d'interprétation d'un échantillon d'image codée auxiliaire pour lequel les échantillons luma et chroma associés de la même unité d'accès sont considérés comme opaques aux fins de la fusion alpha. Le nombre de bits utilisés pour la représentation de l'élément syntaxique alpha_opaque_value est bit_depth_aux_minus8 + 9 bits.

alpha_transparent_value spécifie la valeur d'échantillon d'interprétation d'un échantillon d'image codée auxiliaire pour lequel les échantillons luma et chroma associés de la même unité d'accès sont considérés comme transparents aux fins de la fusion alpha. Le nombre de bits utilisés pour la représentation de l'élément syntaxique alpha_transparent_value est bit_depth_aux_minus8 + 9 bits.

Lorsque alpha_incr_flag est égal à 1, alpha_transparent_value ne doit pas être égal à alpha_opaque_value et Log2(Abs(alpha_opaque_value – alpha_transparent_value)) doit avoir une valeur d'entier. Une valeur de alpha_transparent_value qui est égale à alpha_opaque_value indique que l'image codée auxiliaire n'est pas destinée aux fins de la fusion alpha.

NOTE 2 – Aux fins de la fusion alpha, la valeur d'opacité alpha peut être plus grande que celle de transparence alpha, ou peut être inférieure à cette dernière valeur. Les valeurs d'échantillon d'interprétation devraient être écrêtées de façon à être dans l'étendue des valeurs d'opacité à transparence alpha, bornes incluses.

Le décodage de l'extension d'ensemble de paramètres de séquence et le décodage d'images codées auxiliaires ne sont pas requis pour la conformité avec la présente Recommandation | Norme internationale.

La syntaxe de chaque tranche codée d'une image codée auxiliaire doit respecter les mêmes contraintes qu'une tranche codée d'image redondante, avec les différences suivantes concernant les contraintes.

– Ce qui suit est applicable afin de déterminer si l'image codée primaire est une image à rafraîchissement IDR.
  – Si l'image codée primaire est une image à rafraîchissement IDR, la syntaxe de tranche codée auxiliaire doit correspondre à celle d'une tranche qui utilise nal_unit_type égal à 5 (une tranche d'image à rafraîchissement IDR);
  – Sinon (si l'image codée primaire n'est pas une image à rafraîchissement IDR), la syntaxe de tranche codée auxiliaire doit correspondre à celle d'une tranche qui utilise nal_unit_type égal à 1 (une tranche d'image sans rafraîchissement IDR).
– Les tranches d'une image codée auxiliaire (lorsqu'elle est présente) doivent contenir tous les macroblocs correspondant à ceux de l'image codée primaire.
– redundant_pic_cnt doit être égal à 0 dans toutes les tranches codées d'image auxiliaire.

Le processus de décodage (facultatif) pour le décodage d'images codées auxiliaires est le même que si les images codées auxiliaires étaient des images codées primaires dans un flux vidéo codé distinct, qui diffère des images codées primaires dans l'actuel flux vidéo codé des différentes façons suivantes.

– L'état IDR ou non-IDR de chaque image codée auxiliaire doit être présumé le même que l'état IDR ou non-IDR de l'image primaire située dans la même unité d'accès, plutôt que présumé égal à la valeur de nal_ref_idc.

– La valeur de chroma_format_idc doit être présumée égale à 0 pour le décodage des images codées auxiliaires.

– La valeur de bit_depth_luma_minus8 doit être présumée égale à bit_depth_aux_minus8 pour le décodage des images codées auxiliaires.

NOTE 3 – La composition d'une fusion alpha est normalement effectuée avec une image d'arrière-plan B, une image d'avant-plan F, et une image auxiliaire décodée A, toutes de la même taille. Aux fins de l'illustration de cet exemple, on part du principe que la résolution chromatique de B et de F a été suréchantillonnée à la même résolution que les échantillons luma. Repérer les échantillons correspondants de B, F et A par, respectivement b, f et a. Repérer les échantillons luma et chroma par les indices inférieurs Y, Cb et Cr.

Définir les variables alphaRange, alphaFwt et alphaBwt comme suit:

alphaRange = Abs(alpha_opaque_value – alpha_transparent_value)

alphaFwt = Abs(a – alpha_transparent_value)

alphaBwt = Abs(a – alpha_opaque_value)

Ensuite, dans la composition d'une fusion alpha, les échantillons d de l'image affichée D peuvent être calculées comme suit:

$d_Y = (alphaFwt*f_Y + alphaBwt*b_Y + alphaRange/2) / alphaRange$

$d_{CB} = (alphaFwt*f_{CB} + alphaBwt*b_{CB} + alphaRange/2) / alphaRange$

$d_{CR} = (alphaFwt*f_{CR} + alphaBwt*b_{CR} + alphaRange/2) / alphaRange$

Les échantillons des images D, F et B pourraient également représenter des valeurs des composantes rouge, verte, et bleue (voir § E.2.1). Ici, nous avons pris comme hypothèse Y, Cb et Cr comme valeurs de composante. Chaque composante, p. ex. Y, est supposée avoir aux fins de l'illustration de l'exemple ci-dessus, la même profondeur de bits dans chacune des images D, F et B. Cependant, différentes composantes, p. ex. Y et Cb, n'ont pas besoin d'avoir la même profondeur de bits dans cet exemple.

Lorsque aux_format_idc est égal à 1, F sera l'image décodée obtenue à partir des échantillons luma et chroma décodés, et A sera l'image décodée obtenue de l'image auxiliaire décodée. Dans ce cas, la composition d'une fusion alpha indiquée dans cet exemple implique la multiplication des échantillons de F par des facteurs obtenus à partir des échantillons de A.

Un format d'image qui est utile pour l'édition vidéo ou le visionnement direct, et qui est couramment utilisé, est appelé *vidéo prémultipliée à fond noir*. si l'image d'avant-plan était F, alors la vidéo prémultipliée à fond noir S est donnée par:

$s_Y = (alphaFwt*f_Y) / alphaRange$

$s_{CB} = (alphaFwt*f_{CB}) / alphaRange$

$s_{CR} = (alphaFwt*f_{CR}) / alphaRange$

La vidéo prémultipliée à fond noir offre la caractéristique que l'image S paraîtra correcte si elle affichée sur un fond noir. Pour une image autre qu'à fond noir B, la composition de l'image affichée D peut être calculée comme suit:

$d_Y = s_Y + (alphaBwt*b_Y + alphaRange/2) / alphaRange$

$d_{CB} = s_{CB} + (alphaBwt*b_{CB} + alphaRange/2) / alphaRange$

$d_{CR} = s_{CR} + (alphaBwt*b_{CR} + alphaRange/2) / alphaRange$

Lorsque la variable aux_format_idc est égale à 2, S sera l'image décodée obtenue des échantillons luma et chroma décodés, et A sera de nouveau l'image décodée obtenue de l'image auxiliaire décodée. Dans ce cas, la composition d'une fusion alpha n'implique pas la multiplication des échantillons de S par des facteurs obtenus à partir des échantillons de A.

additional_extension_flag égal à 0 indique qu'aucune donnée additionnelle ne fait suite dans la structure syntaxique de l'extension d'ensemble de paramètres de séquence avant les bits de traîne de charge RBSP. La valeur du fanion additional_extension_flag doit être égale à 0. La valeur de 1 pour le fanion additional_extension_flag est réservée pour utilisation future par l'UIT-T | ISO/CEI. Les décodeurs qui sont conformes à la présente Recommandation | Norme internationale doivent ignorer toutes les données qui font suite à la valeur de 1 pour le fanion additional_extension_flag dans une unité NAL d'extension d'ensemble de paramètres de séquence.

7.4.2.2 Sémantique de la charge utile RBSP d'ensemble de paramètres d'image

pic_parameter_set_id identifie l'ensemble de paramètres d'image qui est référencé dans l'en-tête de tranche. La valeur de pic_parameter_set_id doit être dans la gamme de 0 à 255 inclus.

seq_parameter_set_id se rapporte à l'ensemble actif de paramètres de séquence. La valeur de seq_parameter_set_id doit être dans la gamme de 0 à 31 inclus.

entropy_coding_mode_flag choisit la méthode de décodage entropique à appliquer aux éléments syntaxiques pour lesquels deux descripteurs apparaissent comme suit dans les tableaux syntaxiques:

– Si entropy_coding_mode_flag est égal à 0, on applique la méthode spécifiée par le descripteur de gauche dans le tableau syntaxique (Exp-Golomb codé, voir le § 9.1 ou CAVLC, voir le § 9.2).

– Sinon (si entropy_coding_mode_flag est égal à 1), on applique la méthode spécifiée par le descripteur de droite dans le tableau syntaxique (CABAC, voir le § 9.3).

pic_order_present_flag égal à 1 spécifie que les éléments syntaxiques se rapportant au compte d'ordre d'image sont présents dans les en-têtes de tranche comme spécifié au § 7.3.3. pic_order_present_flag égal à 0 spécifie que les éléments syntaxiques se rapportant au compte d'ordre d'image ne sont pas présents dans les en-têtes de tranche.

num_slice_groups_minus1 plus 1 spécifie le nombre des groupes de tranches pour une image. Lorsque num_slice_groups_minus1 est égal à 0, toutes les tranches de l'image appartiennent au même groupe de tranches. La gamme admise de num_slice_groups_minus1 est spécifiée à l'Annexe A.

slice_group_map_type spécifie comment est codée le mappage d'unités de mappage de groupe de tranches avec les groupes de tranches. La valeur de slice_group_map_type doit être dans la gamme de 0 à 6 inclus.

slice_group_map_type égal à 0 spécifie des groupes de tranches entrelacés.

slice_group_map_type égal à 1 spécifie un mappage dispersé de groupe de tranches.

slice_group_map_type égal à 2 spécifie un ou plusieurs groupes de tranches "de premier plan" et un groupe de tranches "de reste".

Les valeurs de slice_group_map_type égales à 3, 4 et 5 spécifient un changement de groupes de tranches. Lorsque num_slice_groups_minus1 n'est pas égal à 1, slice_group_map_type ne doit pas être égal à 3, 4 ou 5.

slice_group_map_type égal à 6 spécifie une allocation explicite d'un groupe de tranches à chaque unité de mappage de groupe de tranches.

Les unités de mappage de groupe de tranches sont spécifiées comme suit.

– Si frame_mbs_only_flag est égal à 0 et mb_adaptive_frame_field_flag est égal à 1 et si l'image codée est une trame, les unités de mappage de groupe de tranches sont des unités de paires de macroblocs.

– Sinon, si frame_mbs_only_flag est égal à 1 ou si une image codée est une sous-trame, les unités de mappage de groupe de tranches sont des unités de macroblocs.

– Sinon (si frame_mbs_only_flag est égal à 0 et mb_adaptive_frame_field_flag est égal à 0 et l'image codée est une trame), les unités de mappage de groupe de tranches sont des unités de deux macroblocs qui sont verticalement contigus comme dans une paire de macroblocs de trame d'une trame MBAFF.

run_length_minus1[i **]** est utilisé afin de spécifier le nombre d'unités de mappage de groupe de tranches consécutives à allouer au ième groupe de tranches dans l'ordre de balayage matriciel d'unités de mappage de groupe de tranches. La valeur de run_length_minus1[i] doit être dans la gamme de 0 à PicSizeInMapUnits – 1 inclus.

top_left[i **]** et **bottom_right[** i **]** spécifient respectivement les coins supérieur gauche et inférieur droit d'un rectangle. top_left[i] et bottom_right[i] sont des positions d'unité de mappage de groupe de tranches dans un balayage matriciel de l'image pour les unités de mappage de groupe de tranches. Pour chaque rectangle i, toutes les contraintes suivantes doivent être respectées par les valeurs des éléments syntaxiques top_left[i] et bottom_right[i]:

– top_left[i] doit être inférieur ou égal à bottom_right[i] et bottom_right[i] doit être inférieur à PicSizeInMapUnits;

– (top_left[i] % PicWidthInMbs) doit être inférieur ou égal à la valeur de (bottom_right[i] % PicWidthInMbs).

slice_group_change_direction_flag est utilisé avec slice_group_map_type afin de spécifier le type de mappage précis lorsque slice_group_map_type est 3, 4 ou 5.

slice_group_change_rate_minus1 est utilisé afin de spécifier la variable SliceGroupChangeRate. SliceGroupChangeRate spécifie le multiple en nombre d'unités de mappage de groupe de tranches par lequel peut changer la taille d'un groupe de tranches d'une image à la suivante. La valeur de slice_group_change_rate_minus1 doit être dans la gamme de 0 à PicSizeInMapUnits – 1 inclus. La variable SliceGroupChangeRate est spécifiée comme suit:

$$SliceGroupChangeRate = slice\_group\_change\_rate\_minus1 + 1 \quad (7\text{-}20)$$

pic_size_in_map_units_minus1 est utilisé afin de spécifier le nombre d'unités de mappage de groupe de tranches dans l'image. pic_size_in_map_units_minus1 doit être égal à PicSizeInMapUnits – 1.

slice_group_id[i **]** identifie un groupe de tranches de la ième unité de mappage de groupe de tranches dans l'ordre de balayage matriciel. La taille de l'élément syntaxique slice_group_id[i] est de Ceil(Log2(num_slice_groups_minus1 + 1)) bits. La valeur de slice_group_id[i] doit être dans la gamme de 0 à num_slice_groups_minus1 inclus.

num_ref_idx_l0_active_minus1 spécifie l'indice de référence maximal pour la liste 0 d'image de référence qui doit être utilisée afin de décoder chaque tranche de l'image dans laquelle la prédiction de liste 0 est utilisée lorsque

num_ref_idx_active_override_flag est égal à 0 pour la tranche. Lorsque MbaffFrameFlag est égal à 1, num_ref_idx_l0_active_minus1 est la valeur d'indice maximale pour le décodage de macroblocs de trame et 2 * num_ref_idx_l0_active_minus1 + 1 est la valeur d'indice maximale pour le décodage de macroblocs de sous-trame. La valeur de num_ref_idx_l0_active_minus1 doit être dans la gamme de 0 à 31 inclus.

num_ref_idx_l1_active_minus1 a la même sémantique que num_ref_idx_l0_active_minus1 avec l0 et liste 0 remplacés respectivement par l1 et liste 1.

weighted_pred_flag égal à 0 spécifie que la prédiction pondérée ne doit pas être appliquée aux tranches P et SP. weighted_pred_flag égal à 1 spécifie que la prédiction pondérée doit être appliquée aux tranches P et SP.

weighted_bipred_idc égal à 0 spécifie que la prédiction pondérée par défaut doit s'appliquer aux tranches B. weighted_bipred_idc égal à 1 spécifie que la prédiction pondérée explicite doit s'appliquer aux tranches B. weighted_bipred_idc égal à 2 spécifie que la prédiction pondérée implicite doit s'appliquer aux tranches B. La valeur de weighted_bipred_idc doit être dans la gamme de 0 à 2 inclus.

pic_init_qp_minus26 spécifie la valeur initiale moins 26 de $SliceQP_Y$ pour chaque tranche. La valeur initiale est modifiée à la couche de tranche lorsqu'une valeur différente de zéro de slice_qp_delta est décodée, et est modifiée ultérieurement lorsqu'une valeur différente de zéro de mb_qp_delta est décodée à la couche de macrobloc. La valeur de pic_init_qp_minus26 doit être dans la gamme de $-(26 + QpBdOffset_Y)$ à + 25 inclus.

pic_init_qs_minus26 spécifie la valeur initiale moins 26 de $SliceQS_Y$ pour tous les macroblocs dans les tranches SP ou SI. La valeur initiale est modifiée à la couche de tranche lorsqu'une valeur différente de zéro de slice_qs_delta est décodée. La valeur de pic_init_qs_minus26 doit être dans la gamme de – 26 à + 25 inclus.

chroma_qp_index_offset spécifie le décalage qui doit être ajouté à QP_Y et QS_Y afin de s'appliquer au tableau des valeurs de QP_C pour la composante chromatique Cb. La valeur de chroma_qp_index_offset doit être dans la gamme de –12 à +12 inclus.

deblocking_filter_control_present_flag égal à 1 spécifie qu'un ensemble d'éléments syntaxiques contrôlant les caractéristiques du filtre de déblocage est présent dans l'en-tête de tranche. deblocking_filter_control_present_flag égal à 0 spécifie que l'ensemble d'éléments syntaxiques contrôlant les caractéristiques du filtre de déblocage n'est pas présent dans l'en-tête de tranches et que ce sont leurs valeurs déduites qui sont actives.

constrained_intra_pred_flag égal à 0 spécifie que l'intraprédiction permet l'utilisation des données résiduelles et des échantillons décodés des macroblocs du voisinage codés au moyen des modes de prédiction intermacrobloc pour la prédiction des macroblocs codés au moyen des modes de prédiction intramacrobloc. constrained_intra_pred_flag égal à 1 spécifie une intraprédiction soumise à des contraintes, auquel cas la prédiction des macroblocs codés au moyen des modes de prédiction intramacrobloc n'utilise que les données résiduelles et les échantillons décodés provenant des types de macrobloc I ou SI.

redundant_pic_cnt_present_flag égal à 0 spécifie que l'élément syntaxique redundant_pic_cnt n'est pas présent dans l'en-tête de tranches, les subdivisions B de données et les subdivisions C de données qui se rapportent (directement ou par association avec une subdivision A de données correspondante) à l'ensemble de paramètres d'image. redundant_pic_cnt_present_flag égal à 1 spécifie que l'élément syntaxique redundant_pic_cnt est présent dans tout en-tête de tranches, subdivision B de données et subdivision C de données qui se rapporte (directement ou par association avec une subdivision A de données correspondante) à l'ensemble de paramètres d'image.

transform_8x8_mode_flag égal à 1 spécifie que le processus de décodage de transformée 8x8 peut être en cours d'utilisation (voir § 8.5). Un fanion transform_8x8_mode_flag égal à 0 spécifie que le processus de décodage de transformée 8x8 n'est pas en cours d'utilisation. Lorsque transform_8x8_mode_flag n'est pas présent, sa valeur doit être présumée égale à 0.

pic_scaling_matrix_present_flag égal à 1 spécifie que des paramètres sont présents afin de modifier les listes de normalisation spécifiées dans l'ensemble de paramètres de séquence. Un fanion pic_scaling_matrix_present_flag égal à 0 spécifie que les listes de normalisation utilisées pour l'image doivent être présumées égales à celles qui sont spécifiées par l'ensemble de paramètres de séquence. Lorsque pic_scaling_matrix_present_flag n'est pas présent, sa valeur doit être supposée égale à 0.

pic_scaling_list_present_flag[i **]** égal à 1 spécifie que la structure syntaxique de liste de normalisation est présente afin de spécifier la liste de normalisation pour l'indice i. Un fanion pic_scaling_list_present_flag[i] égal à 0 spécifie que la structure syntaxique pour la liste de normalisation i n'est pas présente dans l'ensemble de paramètres d'image et que, selon la valeur de seq_scaling_matrix_present_flag, ce qui suit est applicable:

– Si seq_scaling_matrix_present_flag est égal à 0, l'ensemble de règles de repli vers une liste de normalisation A comme spécifié dans le Tableau 7-2 doit être utilisé afin de calculer la liste de normalisation au niveau des images pour l'indice i.

– Sinon (si seq_scaling_matrix_present_flag est égal à 1), l'ensemble de règles de repli vers une liste de normalisation B comme spécifié dans le Tableau 7-2 doit être utilisé afin de calculer la liste de normalisation au niveau des images pour l'indice i.

second_chroma_qp_indice_offset spécifie le décalage qui doit être ajouté à QP_Y et QS_Y afin de consulter la table de valeurs QP_C pour la composante chroma Cr. La valeur de la variable second_chroma_qp_indice_offset doit être dans l'étendue de –12 à +12, inclus.

Lorsque la variable second_chroma_qp_indice_offset n'est pas présente, sa valeur doit être supposée égale à chroma_qp_indice_offset.

7.4.2.3 Sémantique de charge utile RBSP d'informations d'amélioration supplémentaires

Les informations d'amélioration supplémentaires (SEI, *supplemental enhancement information*) contiennent des informations qui ne sont pas nécessaires afin de décoder les échantillons d'images codées à partir des unités NAL de couche VCL.

7.4.2.3.1 Sémantique de message d'informations d'amélioration supplémentaires

Une unité NAL d'informations SEI contient un ou plusieurs messages SEI. Chaque message SEI se compose des variables spécifiant le type payloadType et la longueur payloadSize de la charge utile SEI. Les charges utiles SEI sont spécifiées à l'Annexe D. La longueur de charge utile de SEI déduite payloadSize est spécifiée en octets et doit être égale au nombre d'octets de la charge utile SEI.

ff_byte est un octet égal à 0xFF identifiant le besoin d'une plus longue représentation de la structure syntaxique qui est utilisée en son sein.

last_payload_type_byte est le dernier octet du type de charge utile d'un message SEI.

last_payload_size_byte est le dernier octet de la longueur d'un message SEI.

7.4.2.4 Sémantique de la charge utile RBSP de délimiteur d'unité d'accès

Le délimiteur d'unité d'accès peut être utilisé afin d'indiquer le type de tranche présente dans une image codée primaire et afin de simplifier la détection de la frontière entre les unités d'accès. Il n'y a pas de processus normatif de décodage associé au délimiteur d'unité d'accès.

primary_pic_type indique que les valeurs de slice_type pour toutes les tranches de l'image codée primaire font partie de l'ensemble dont la liste figure au Tableau 7-5 pour les valeurs données de primary_pic_type.

Tableau 7-5 – Signification de primary_pic_type

primary_pic_type	Valeurs de slice_type qui peuvent être présentes dans l'image codée primaire
0	I
1	I, P
2	I, P, B
3	SI
4	SI, SP
5	I, SI
6	I, SI, P, SP
7	I, SI, P, SP, B

7.4.2.5 Sémantique de charge utile RBSP de fin de séquence

La charge utile RBSP de fin de séquence spécifie que l'unité d'accès suivante dans le flux binaire dans l'ordre de décodage (le cas échéant) doit être une unité d'accès IDR. Le contenu syntaxique de la chaîne SODB et de la charge utile RBSP pour la fin de la séquence RBSP est vide. Aucun processus de décodage normalisé n'est spécifié pour une charge RBSP de fin de séquence.

7.4.2.6 Sémantique de charge utile RBSP de fin de flux

La charge utile RBSP de fin de flux indique qu'aucune unité NAL supplémentaire ne doit être présente dans le flux binaire qui suit la charge utile RBSP de fin de flux dans l'ordre de décodage. Le contenu syntaxique de la chaîne SODB et de la charge utile RBSP pour la RBSP de fin de flux est vide. Aucun processus de décodage normalisé n'est spécifié pour une charge RBSP de fin de flux.

7.4.2.7 Sémantique de charge utile RBSP de données de remplissage

La charge utile RBSP de données de remplissage contient des octets dont la valeur doit être égale à 0xFF. Aucun processus de décodage normalisé n'est spécifié pour une charge RBSP de données de remplissage.

ff_byte est un octet égal à 0xFF.

7.4.2.8 Sémantique de charge utile RBSP de couche de tranche sans subdivision

La charge utile RBSP de couche de tranche sans subdivision consiste en un en-tête de tranche et des données de tranche.

7.4.2.9 Sémantique de charge utile RBSP de subdivision de données de tranche

7.4.2.9.1 Sémantique de charge utile RBSP de subdivision A de données de tranche

Lorsqu'on utilise la subdivision de données de tranche, les données codées pour une tranche seule sont partagées en trois subdivisions séparées. La subdivision A contient tous les éléments syntaxiques de catégorie 2.

Les éléments syntaxiques de catégorie 2 incluent tous les éléments syntaxiques des structures syntaxiques d'en-tête de tranche et de données de tranche autres que les éléments syntaxiques dans les structures syntaxiques residual( ).

slice_id identifie la tranche associée à la subdivision de données. Chaque tranche doit avoir une valeur de slice_id unique au sein de l'image codée qui contient la tranche. Lorsqu'un ordre de tranche arbitraire n'est pas admis, comme spécifié à l'Annexe A, la première tranche d'une image codée, dans l'ordre de décodage, doit avoir slice_id égal à 0 et la valeur de slice_id doit être incrémentée d'une unité pour chaque tranche suivante de l'image codée dans l'ordre de décodage.

La gamme de slice_id est spécifiée comme suit.

– Si MbaffFrameFlag est égal à 0, slice_id doit être dans la gamme de 0 à PicSizeInMbs – 1 inclus.

– Sinon (si MbaffFrameFlag est égal à 1), slice_id doit être dans la gamme de 0 à PicSizeInMbs / 2 – 1 inclus.

7.4.2.9.2 Sémantique de charge utile RBSP de subdivision B de données de tranche

Lorsqu'on utilise la subdivision de données de tranche, les données codées pour une tranche seule sont partagées en trois subdivisions séparées. La subdivision B contient tous les éléments syntaxiques de catégorie 3.

Les éléments syntaxiques de catégorie 3 incluent tous les éléments syntaxiques des structures syntaxiques residual( ) et des structures syntaxiques utilisées au sein de cette structure syntaxique pour les types collectifs de macrobloc I et SI comme spécifié au Tableau 7-10.

slice_id a la même sémantique que spécifié au § 7.4.2.9.1.

redundant_pic_cnt doit être égal à 0 pour les tranches et les subdivisions de données de tranche appartenant à l'image codée primaire. Le compte redundant_pic_cnt doit être supérieur à 0 pour les tranches codées et les subdivisions de données de tranche codée dans les images codées redondantes. Lorsque redundant_pic_cnt n'est pas présent, sa valeur doit être supposée égale à 0. La valeur de redundant_pic_cnt doit être dans la gamme de 0 à 127 inclus.

La présence d'une charge utile RBSP de subdivision B de données de tranche est spécifiée comme suit.

– Si les éléments syntaxiques d'une charge utile RBSP de subdivision B de données de tranche indiquent la présence de tout élément syntaxique de catégorie 3 dans les données de tranche pour une tranche, une charge utile RBSP de subdivision B de données de tranche ayant la même valeur de slice_id et de redundant_pic_cnt que dans la RBSP de subdivision A de données de tranche doit être présente.

– Sinon (si les éléments syntaxiques d'une charge utile RBSP de subdivision A de données de tranche n'indiquent pas la présence d'éléments syntaxiques de catégorie 3 dans les données de tranche pour une tranche), aucune charge utile RBSP de subdivision B de données de tranche ayant la même valeur de slice_id et de redundant_pic_cnt que dans la RBSP de subdivision A de données de tranche ne doit être présente.

7.4.2.9.3 Sémantique de charge utile RBSP de subdivision C de données de tranche

Lorsqu'on utilise la subdivision de données de tranche, les données codées pour une tranche seule sont partagées en trois subdivisions séparées. La subdivision C contient tous les éléments syntaxiques de catégorie 4.

Les éléments syntaxiques de catégorie 4 incluent tous les éléments syntaxiques des structures syntaxiques residual( ) et des structures syntaxiques utilisées au sein de cette structure syntaxique pour les types collectifs de macrobloc P et B comme spécifié au Tableau 7-10.

slice_id a la même sémantique que spécifié au § 7.4.2.9.1.

redundant_pic_cnt a la même sémantique que spécifié au § 7.4.2.9.2.

La présence d'une charge utile RBSP de subdivision C de données de tranche est spécifiée comme suit.

– Si les éléments syntaxiques d'une charge utile RBSP de subdivision A de données de tranche indiquent la présence de tout élément syntaxique de catégorie 4 dans les données de tranche pour une tranche, une charge utile RBSP de subdivision C de données de tranche ayant la même valeur de slice_id et de redundant_pic_cnt que dans la RBSP de subdivision A de données de tranche doit être présente.

– Sinon (si les éléments syntaxiques d'une charge utile RBSP de subdivision A de données de tranche n'indiquent pas la présence d'éléments syntaxiques de catégorie 4 dans les données de tranche pour une tranche), aucune charge utile RBSP de subdivision C de données de tranche ayant la même valeur de slice_id et de redundant_pic_cnt que dans la RBSP de subdivision A de données de tranche ne doit être présente.

7.4.2.10 Sémantique des bits de traîne de tranche de charge utile RBSP

cabac_zero_word est une séquence alignée sur les octets de deux octets égale à 0x0000.

Soit NumBytesInVclNALunits la somme des valeurs de NumBytesInNALunit pour toutes les unités NAL de couche VCL d'une image codée.

Soit BinCountsInNALunits le nombre de fois que la fonction du processus d'analyse syntaxique DecodeBin( ), spécifiée au § 9.3.3.2, est invoquée afin de décoder le contenu de toutes les unités NAL de couche VCL d'une image codée. Lorsque le fanion entropy_coding_mode_flag est égal à 1, BinCountsInNALunits ne doit pas excéder (32 ÷ 3) * NumBytesInVclNALunits + (RawMbBits * PicSizeInMbs) ÷ 32.

NOTE – La contrainte sur le nombre maximal de segments résultant du décodage du contenu de la couche de tranche des unités NAL peut être satisfaite en insérant un nombre d'éléments syntaxiques cabac_zero_word afin d'augmenter la valeur de NumBytesInVclNALunits. Chaque cabac_zero_word est représenté dans une unité NAL par la séquence de trois octets 0x000003 (résultant de la contrainte sur le contenu de l'unité NAL qui requiert l'inclusion d'un emulation_prevention_three_byte pour chaque cabac_zero_word).

7.4.2.11 Sémantique des bits de traîne de charge utile RBSP

rbsp_stop_one_bit doit être égal à 1.

rbsp_alignment_zero_bit doit être égal à 0.

7.4.3 Sémantique d'en-tête de tranche

Lorsqu'elle est présente, la valeur des éléments syntaxiques d'en-tête de tranche pic_parameter_set_id, frame_num, field_pic_flag, bottom_field_flag, idr_pic_id, pic_order_cnt_lsb, delta_pic_order_cnt_bottom, delta_pic_order_cnt[0], delta_pic_order_cnt[1], sp_for_switch_flag et slice_group_change_cycle doit être la même dans tous les en-têtes de tranche d'une image codée.

first_mb_in_slice spécifie l'adresse du premier macrobloc dans la tranche. Lorsque l'ordre arbitraire de tranche n'est pas admis comme spécifié à l'Annexe A, la valeur de first_mb_in_slice ne doit pas être inférieure à la valeur de first_mb_in_slice pour toute autre tranche de l'image en cours qui précède la tranche en cours dans l'ordre de décodage.

La première adresse de macrobloc de la tranche est déduite comme suit.

– Si MbaffFrameFlag est égal à 0, first_mb_in_slice est l'adresse de macrobloc du premier macrobloc dans la tranche et first_mb_in_slice doit être dans la gamme de 0 à PicSizeInMbs – 1 inclus.

– Sinon (si MbaffFrameFlag est égal à 1), first_mb_in_slice * 2 est l'adresse de macrobloc du premier macrobloc dans la tranche, qui est le macrobloc supérieur de la première paire de macroblocs dans la tranche, et first_mb_in_slice doit être dans la gamme de 0 à PicSizeInMbs / 2 – 1 inclus.

slice_type spécifie le type de codage de la tranche conformément au Tableau 7-6**.**

Tableau 7-6 – Association du nom au type de tranche slice_type

slice_type	Nom de type slice_type
0	P (tranche P)
1	B (tranche B)
2	I (tranche I)
3	SP (tranche SP)
4	SI (tranche SI)
5	P (tranche P)
6	B (tranche B)
7	I (tranche I)
8	SP (tranche SP)
9	SI (tranche SI)

Les valeurs de slice_type dans la gamme de 5..9 spécifient, en plus du type de codage de la tranche en cours, que toutes les autres tranches de l'image codée en cours doivent avoir une valeur de slice_type égale à la valeur courante de slice_type ou égale à la valeur courante de slice_type – 5.

Lorsque nal_unit_type est égal à 5 (image à rafraîchissement IDR), slice_type doit être égal à 2, 4, 7 ou 9.

Lorsque num_ref_frames est égal à 0, slice_type doit être égal à 2, 4, 7 ou 9.

pic_parameter_set_id spécifie l'ensemble de paramètres d'image utilisé. La valeur de pic_parameter_set_id doit être dans la gamme de 0 à 255 inclus.

frame_num est utilisé comme un identificateur pour les images et doit être représenté par log2_max_frame_num_minus4 + 4 bit dans le flux binaire. frame_num est soumis aux contraintes suivantes.

La variable PrevRefFrameNum est déduite comme suit.

– Si l'image en cours est à rafraîchissement IDR, PrevRefFrameNum est mis égal à 0.

– Sinon (si l'image en cours n'est pas à rafraîchissement IDR), PrevRefFrameNum est réglé comme suit.

  – Si le processus de décodage de lacunes dans frame_num, spécifié dans le § 8.2.5.2, a été invoqué par le processus de décodage pour une unité d'accès qui contenait une image non référentielle qui suivait la précédente unité d'accès dans l'ordre du décodage qui contenait une image de référence, PrevRefFrameNum est mis à la valeur de frame_num pour la dernière des trames de référence "non existantes" présumées par le processus de décodage de lacunes dans frame_num, spécifié dans le § 8.2.5.2.

  – Sinon, PrevRefFrameNum est mis à la valeur de frame_num pour la précédente unité d'accès dans l'ordre du décodage qui contenait une image de référence.

La valeur de frame_num est soumise aux contraintes suivantes.

– Si l'image en cours est à rafraîchissement IDR, frame_num doit être égal à 0.

– Sinon (si l'image en cours n'est pas à rafraîchissement IDR), par rapport à l'image codée primaire dans la précédente unité d'accès dans l'ordre de décodage qui contient une image de référence en tant que précédente image de référence, la valeur de frame_num pour l'image en cours ne doit pas être égale à PrevRefFrameNum à moins que toutes les trois conditions suivantes ne soient vraies:

  – l'image en cours et la précédente image de référence appartiennent à des unités d'accès consécutives dans l'ordre de décodage;

  – l'image en cours et la précédente image de référence sont des sous-trames de référence ayant une parité opposée;

  – une ou plusieurs des conditions suivantes sont vraies:

    – l'image de référence précédente est à rafraîchissement IDR;

    – l'image de référence précédente inclut un élément syntaxique memory_management_control_operation égal à 5.

      NOTE 1 – Lorsque l'image de référence précédente inclut un élément syntaxique memory_management_control_operation égal à 5, PrevRefFrameNum est égal à 0;

    – il y a une image codée primaire qui précède la précédente image de référence et l'image codée primaire qui précède la précédente image de référence n'a pas de frame_num égal à PrevRefFrameNum;

– il y a une image codée primaire qui précède la précédente image de référence et l'image codée primaire qui précède la précédente image de référence n'est pas une image de référence.

Lorsque la valeur de frame_num n'est pas égale à PrevRefFrameNum, on applique ce qui suit.

– Il ne doit pas y avoir de sous-trame ou trame précédant dans l'ordre de décodage qui soit actuellement marquée "utilisée comme référence à court terme" qui ait une valeur de frame_num égale à toute valeur prise par la variable UnusedShortTermFrameNum dans ce qui suit:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum  !=  frame_num )                    (7-21)
UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

– La valeur de frame_num est soumise aux contraintes suivantes.
  – Si gaps_in_frame_num_value_allowed_flag est égal à 0, la valeur de frame_num pour l'image en cours doit être égale à (PrevRefFrameNum + 1) % MaxFrameNum.
  – Sinon (si gaps_in_frame_num_value_allowed_flag est égal à 1), on applique ce qui suit:
    – Si frame_num est supérieur à PrevRefFrameNum, il ne doit y avoir aucune image non référentielle dans le flux binaire qui suit l'image de référence précédente et précède l'image en cours dans l'ordre de décodage pour lequel l'une des deux conditions suivantes est vraie.
      – La valeur de frame_num pour l'image non référentielle est inférieure à PrevRefFrameNum.
      – La valeur de frame_num pour l'image non référentielle est supérieure à la valeur de frame_num pour l'image en cours.
    – Sinon (si frame_num est inférieur à PrevRefFrameNum), il ne doit y avoir aucune image non référentielle dans le flux binaire qui suit l'image de référence précédente et précède l'image en cours dans l'ordre de décodage pour lequel les deux conditions suivantes sont vraies.
      – La valeur de frame_num pour l'image non référentielle est inférieure à PrevRefFrameNum.
      – La valeur de frame_num pour l'image non référentielle est supérieure à la valeur de frame_num pour l'image en cours.

Une image incluant un élément memory_management_control_operation égal à 5 doit avoir un frame_num soumis aux contraintes décrites ci-dessus et après le décodage de l'image en cours et le traitement des opérations de commande de gestion de mémoire, l'image doit être supposée avoir eu frame_num égal à 0 pour toutes les utilisations suivantes dans le processus de décodage, sauf comme indiqué au § 7.4.1.2.4.

NOTE 2 – Lorsque l'image codée primaire n'est pas à rafraîchissement IDR et ne contient pas d'élément syntaxique memory_management_control_operation égal à 5, la valeur de frame_num d'une image codée redondante est la même que la valeur de frame_num dans l'image codée primaire. Sinon, l'image codée redondante inclut un élément syntaxique memory_management_control_operation égal à 5 et l'image codée primaire correspondante est à rafraîchissement IDR.

field_pic_flag égal à 1 spécifie que la tranche est une tranche d'une sous-trame codée. field_pic_flag égal à 0 spécifie que la tranche est une tranche d'une trame codée. Lorsque field_pic_flag n'est pas présent il doit être supposé égal à 0.

La variable MbaffFrameFlag est déduite comme suit:

```
MbaffFrameFlag = ( mb_adaptive_frame_field_flag  &&  !field_pic_flag )    (7-22)
```

La variable pour la hauteur de l'image en unités de macroblocs est déduite comme suit:

```
PicHeightInMbs = FrameHeightInMbs / ( 1 + field_pic_flag )                (7-23)
```

La variable pour la hauteur de l'image pour la composante luma est déduite comme suit:

$$PicHeightInSamples_L = PicHeightInMbs * 16 \quad (7\text{-}24)$$

La variable pour la hauteur de l'image pour la composante chroma est déduite comme suit:

$$PicHeightInSamples_C = PicHeightInMbs * MbHeightC \quad (7\text{-}25)$$

La variable PicSizeInMbs pour l'image en cours est déduite conformément à:

$$PicSizeInMbs = PicWidthInMbs * PicHeightInMbs \quad (7\text{-}26)$$

La variable MaxPicNum est déduite comme suit:

– Si field_pic_flag est égal à 0, MaxPicNum est mis égal à MaxFrameNum.

– Sinon (si field_pic_flag est égal à 1), MaxPicNum est mis égal à 2*MaxFrameNum.

La variable CurrPicNum est déduite comme suit:

– Si field_pic_flag est égal à 0, CurrPicNum est mis égal à frame_num.

– Sinon (si field_pic_flag est égal à 1), CurrPicNum est mis égal à 2 * frame_num + 1.

bottom_field_flag égal à 1 spécifie que la tranche fait partie d'une sous-trame inférieure codée. Un fanion bottom_field_flag égal à 0 spécifie que l'image est une sous-trame supérieure codée. Lorsque cet élément syntaxique n'est pas présent pour la tranche en cours, il doit être supposé égal à 0.

idr_pic_id identifie uneimage à rafraîchissement IDR. Les valeurs de idr_pic_id dans toutes les tranches d'uneimage à rafraîchissement IDR doivent rester inchangées. Lorsque deux unités d'accès consécutives dans l'ordre de décodage sont toutes deux des unités d'accès IDR, la valeur de idr_pic_id dans les tranches de la première d'une telle unité d'accès IDR doit différer de celle de idr_pic_id dans la seconde unité d'accès IDR. La valeur de idr_pic_id doit être dans la gamme de 0 à 65535 inclus.

pic_order_cnt_lsb spécifie le compte d'ordre d'image modulo MaxPicOrderCntLsb pour la sous-trame supérieure d'une trame codée ou pour une sous-trame codée. La taille de l'élément syntaxique pic_order_cnt_lsb est log2_max_pic_order_cnt_lsb_minus4 + 4 bit. La valeur du pic_order_cnt_lsb doit être dans la gamme de 0 à MaxPicOrderCntLsb – 1, inclus.

delta_pic_order_cnt_bottom spécifie comme suit la différence de compte d'ordre d'image entre la sous-trame inférieure et la sous-trame supérieure d'une trame codée.

– Si l'image en cours inclut une memory_management_control_operation égale à 5, la valeur de delta_pic_order_cnt_bottom doit être dans la gamme de (1 – MaxPicOrderCntLsb) à $2^{31} – 1$ inclus.

– Sinon (si l'image en cours n'inclut pas une memory_management_control_operation égale à 5), la valeur de delta_pic_order_cnt_bottom doit être dans la gamme de $–2^{31}$ à $2^{31} – 1$ inclus.

Lorsque cet élément syntaxique n'est pas présent dans le flux binaire pour la tranche en cours, il doit être supposé égal à 0.

delta_pic_order_cnt[0 **]** spécifie la différence de compte d'ordre d'image par rapport au compte d'ordre d'image attendu pour la sous-trame supérieure d'une trame codée ou pour une sous-trame codée comme spécifié au § 8.2.1. La valeur de delta_pic_order_cnt[0] doit être dans la gamme de $–2^{31}$ à $2^{31} – 1$ inclus. Lorsque cet élément syntaxique n'est pas présent dans le flux binaire pour la tranche en cours, il doit être supposé égal à 0.

delta_pic_order_cnt[1 **]** spécifie la différence de compte d'ordre d'image par rapport au compte d'ordre d'image attendu pour la sous-trame inférieure d'une trame codée comme spécifié au § 8.2.1. La valeur de delta_pic_order_cnt[1] doit être dans la gamme de $–2^{31}$ à $2^{31} – 1$ inclus. Lorsque cet élément syntaxique n'est pas présent dans le flux binaire pour la tranche en cours, il doit être supposé égal à 0.

redundant_pic_cnt doit être égal à 0 pour les tranches et subdivisions de données de tranche appartenant à l'image codée primaire. La valeur de redundant_pic_cnt doit être supérieure à 0 pour les tranches codées ou subdivisions de données de tranche codée d'une image codée redondante. Lorsque redundant_pic_cnt n'est pas présent dans le flux binaire, sa valeur doit être supposée égale à 0. La valeur de redundant_pic_cnt doit être dans la gamme de 0 à 127 inclus.

NOTE 3 – Toute zone de l'image primaire décodée et la zone correspondante résultant de l'application du processus de décodage spécifié à l'article 8 pour toute image redondante dans la même unité d'accès devraient avoir une apparence visuelle similaire.

La valeur de pic_parameter_set_id dans une tranche codée ou subdivision de données de tranche codée d'une image codée redondante doit être telle que la valeur de pic_order_present_flag dans l'ensemble de paramètres d'image utilisés dans une image codée redondante soit égale à la valeur de pic_order_present_flag dans l'ensemble de paramètres d'image utilisé dans l'image codée primaire correspondante.

Lorsqu'ils sont présents dans l'image codée primaire et toute image codée redondante, les éléments syntaxiques suivants doivent avoir la même valeur: field_pic_flag, bottom_field_flag et idr_pic_id.

Lorsque la valeur de nal_ref_idc dans une unité NAL de couche VCL d'une unité d'accès est égale à 0, la valeur de nal_ref_idc dans toutes les autres unités NAL de couche VCL de la même unité d'accès doit être égale à 0.

NOTE 4 – La contrainte ci-dessus a les implications suivantes. Si la valeur de nal_ref_idc pour les unités NAL de couche VCL de l'image codée primaire est égale à 0, la valeur de nal_ref_idc pour les unités NAL de couche VCL de toute image codée redondante correspondante est égale à 0; Sinon (si la valeur de nal_ref_idc pour les unités NAL de couche VCL de l'image codée primaire est supérieure à 0), la valeur de nal_ref_idc pour les unités NAL de couche VCL de toute image codée redondante correspondante est aussi supérieure à 0.

L'état de marquage des images de référence et la valeur de frame_num après que le processus de marquage d'image de référence décodée comme spécifié au § 8.2.5 est invoqué pour l'image codée primaire ou pour toute image codée redondante de la même unité d'accès, doivent être identiques, sans considérer si l'image codée primaire ou toute image codée redondante (à la place de l'image codée primaire) de l'unité d'accès serait décodée.

NOTE 5 – La contrainte ci-dessus a aussi les implications suivantes.

Si une image codée primaire n'est pas à rafraîchissement IDR, le contenu de la structure syntaxique dec_ref_pic_marking() doit être identique dans tous les en-têtes de tranche de l'image codée primaire et toutes les images codées redondantes correspondantes à l'image codée primaire.

Sinon (si une image codée primaire est à rafraîchissement IDR), on applique ce qui suit.

Si une image codée redondante correspondant à l'image codée primaire est à rafraîchissement IDR, le contenu de la structure syntaxique dec_ref_pic_marking() doit être identique dans tous les en-têtes de tranche de l'image codée primaire et les images codées redondantes correspondantes à l'image codée primaire.

Sinon (si une image redondante correspondant à l'image codée primaire n'est pas à rafraîchissement IDR), tous les en-têtes de tranche de l'image redondante doivent contenir une structure syntaxique dec_ref_pic_marking() incluant un élément syntaxique memory_management_control_operation égal à 5 et on applique ce qui suit.

Si la valeur de long_term_reference_flag dans l'image codée primaire est égale à 0, la structure syntaxique dec_ref_pic_marking de l'image codée redondante ne doit pas inclure d'élément syntaxique memory_management_control_operation égal à 6.

Sinon (si la valeur de long_term_reference_flag dans l'image codée primaire est égale à 1), la structure syntaxique dec_ref_pic_marking de l'image codée redondante doit inclure des éléments syntaxiques memory_management_control_operation égaux à 5, 4 et 6 en ordre de décodage et la valeur de max_long_term_trame_idx_plus1 doit être égale à 1 et la valeur de long_term_trame_idx doit être égale à 0.

Les valeurs de TopFieldOrderCnt et BottomFieldOrderCnt (s'ils sont applicables) qui résultent après achèvement du processus de décodage pour toute image codée redondante ou l'image codée primaire de la même unité d'accès doivent être identiques sans considérer si l'image codée primaire ou toute image codée redondante (à la place de l'image codée primaire) de l'unité d'accès serait décodée.

Aucun processus de décodage n'est requis pour une tranche codée ou subdivision de données de tranche codée d'une image codée redondante. Lorsque le redundant_pic_cnt dans l'en-tête de tranche d'une tranche codée est supérieur à 0, le décodeur peut mettre à l'écart la tranche codée. Cependant, une tranche codée ou subdivision de données de tranche codée de toute image codée redondante doit respecter les mêmes contraintes qu'une tranche codée ou subdivision de données de tranche codée d'une image primaire.

NOTE 6 – Lorsque quelques-uns des échantillons dans l'image primaire décodée ne peuvent pas être correctement décodés à cause d'erreurs ou pertes dans la transmission de la séquence et qu'une tranche redondante codée peut être correctement décodée, le décodeur devrait remplacer les échantillons de l'image primaire décodée par les échantillons correspondants de la tranche redondante décodée. Lorsque plus d'une tranche redondante couvre la région considérée de l'image primaire, la tranche redondante ayant la plus faible valeur de redundant_pic_cnt devrait être utilisée.

Les tranches redondantes et les subdivisions de données de tranche ayant la même valeur de redundant_pic_cnt appartiennent à la même image redondante. Les tranches décodées au sein de la même image redondante ne doivent pas nécessairement couvrir l'image entière et ne doivent pas se chevaucher.

direct_spatial_mv_pred_flag spécifie comme suit la méthode utilisée dans le processus de décodage afin de déduire les vecteurs cinétiques et les indices de référence pour l'interprédiction.

– Si direct_spatial_mv_pred_flag est égal à 1, le processus de déduction pour les vecteurs cinétiques luma pour B_Skip, B_Direct_16x16 et B_Direct_8x8 décrit au § 8.4.1.2 doit utiliser la prédiction spatiale de mode direct comme spécifié au § 8.4.1.2.2.

– Sinon (si direct_spatial_mv_pred_flag est égal à 0), le processus de déduction pour les vecteurs cinétiques luma pour B_Skip, B_Direct_16x16 et B_Direct_8x8 décrit au § 8.4.1.2 doit utiliser la prédiction temporelle de mode direct comme spécifié au § 8.4.1.2.3.

num_ref_idx_active_override_flag égal à 0 spécifie que les valeurs des éléments syntaxiques num_ref_idx_l0_active_minus1 et num_ref_idx_l1_active_minus1 spécifiés dans l'ensemble de paramètres d'image référencé sont effectifs. num_ref_idx_active_override_flag égal à 1 spécifie que num_ref_idx_l0_active_minus1 et num_ref_idx_l1_active_minus1 spécifiés dans l'ensemble de paramètres d'image pris en référence sont supplantés pour la tranche en cours (et seulement pour la tranche en cours) par les valeurs suivantes dans l'en-tête de tranche.

Lorsque la tranche en cours est une tranche P, SP ou B et que field_pic_flag est égal à 0 et que la valeur de num_ref_idx_l0_active_minus1 dans l'ensemble de paramètres d'image excède 15, num_ref_idx_active_override_flag doit être égal à 1.

Lorsque la tranche en cours est une tranche B et que field_pic_flag est égal à 0 et que la valeur de num_ref_idx_l1_active_minus1 dans l'ensemble de paramètres d'image excède 15, num_ref_idx_active_override_flag doit être égal à 1.

num_ref_idx_l0_active_minus1 spécifie l'indice de référence maximal pour la liste 0 d'images de référence qui doit être utilisée afin de décoder la tranche.

La gamme de num_ref_idx_l0_active_minus1 est spécifiée comme suit.

– Si field_pic_flag est égal à 0, num_ref_idx_l0_active_minus1 doit être dans la gamme de 0 à 15 inclus. Lorsque MbaffFrameFlag est égal à 1, num_ref_idx_l0_active_minus1 est la valeur maximale d'indice pour le décodage de macroblocs de trame et 2 * num_ref_idx_l0_active_minus1 + 1 est la valeur maximale d'indice pour le décodage de macroblocs de sous-trame.

– Sinon (si field_pic_flag est égal à 1), num_ref_idx_l0_active_minus1 doit être dans la gamme de 0 à 31 inclus.

num_ref_idx_l1_active_minus1 a la même sémantique que num_ref_idx_l0_active_minus1 avec l0 et liste 0 remplacés respectivement par l1 et liste 1.

cabac_init_idc spécifie l'indice afin de déterminer le tableau d'initialisation utilisé dans le processus d'initialisation pour les variables de contexte. La valeur de cabac_init_idc doit être dans la gamme de 0 à 2 inclus.

slice_qp_delta spécifie la valeur initiale de QP_Y à utiliser pour tous les macroblocs dans la tranche jusqu'à sa modification par la valeur de mb_qp_delta dans la couche de macrobloc. Le paramètre initial de quantification QP_Y pour la tranche est calculé comme suit:

$$SliceQP_Y = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta \quad (7\text{-}27)$$

La valeur de slice_qp_delta doit être limitée de telle sorte que QP_Y soit dans la gamme de $-QpBdOffset_Y$ à +51 inclus.

sp_for_switch_flag spécifie le processus de décodage à utiliser afin de décoder comme suit des macroblocs P dans une tranche SP.

– Si sp_for_switch_flag est égal à 0, les macroblocs P dans la tranche SP doivent être décodés au moyen du processus de décodage SP pour les images non commutatives, comme spécifié au § 8.6.1.

– Sinon (si sp_for_switch_flag est égal à 1), les macroblocs P dans la tranche SP doivent être décodés au moyen des processus de décodage SP et SI pour les images commutatives, comme spécifié au § 8.6.2.

slice_qs_delta spécifie la valeur de QS_Y pour tous les macroblocs dans les tranches SP et SI. Le paramètre de quantification QS_Y pour la tranche est calculé comme suit:

$$QS_Y = 26 + pic\_init\_qs\_minus26 + slice\_qs\_delta \quad (7\text{-}28)$$

La valeur de slice_qs_delta doit être limitée de telle sorte que QS_Y soit dans la gamme de 0 à 51 inclus. Cette valeur de QS_Y est utilisée pour le décodage de tous les macroblocs dans les tranches SI avec mb_type égal à SI et tous les macroblocs dans les tranches SP avec le mode de prédiction égal à inter.

disable_deblocking_filter_idc spécifie si le fonctionnement du filtre de démontage de bloc peut être désactivé sur certaines bordures de blocs de la tranche et spécifie pour quelles bordures le filtrage est désactivé. Lorsque disable_deblocking_filter_idc n'est pas présent dans l'en-tête de tranche, la valeur de disable_deblocking_filter_idc doit être supposée égale à 0.

La valeur de disable_deblocking_filter_idc doit être dans la gamme de 0 à 2 inclus.

slice_alpha_c0_offset_div2 spécifie le décalage utilisé en accédant aux tableaux α et t_{C0} de filtre de démontage de bloc pour les opérations de filtrage contrôlées par les macroblocs au sein de la tranche. A partir de cette valeur, le décalage qui doit être appliqué lors de l'utilisation de ces tableaux doit être calculé comme suit:

$$FilterOffsetA = slice\_alpha\_c0\_offset\_div2 << 1 \quad (7\text{-}29)$$

La valeur de slice_alpha_c0_offset_div2 doit être dans la gamme de –6 à +6 inclus. Lorsque slice_alpha_c0_offset_div2 n'est pas présent dans l'en-tête de tranche, la valeur de slice_alpha_c0_offset_div2 doit être supposée égale à 0.

slice_beta_offset_div2 spécifie le décalage utilisé en accédant au tableau β de filtre de démontage de bloc pour les opérations de filtrage contrôlées par les macroblocs au sein de la tranche. A partir de cette valeur, le décalage qui est appliqué lors de l'utilisation du tableau β du filtre de démontage de bloc doit être calculé comme suit:

$$FilterOffsetB = slice\_beta\_offset\_div2 << 1 \quad (7\text{-}30)$$

La valeur de slice_beta_offset_div2 doit être dans la gamme de –6 à +6 inclus. Lorsque slice_beta_offset_div2 n'est pas présent dans l'en-tête de tranche, la valeur de slice_beta_offset_div2 doit être supposée égale à 0.

slice_group_change_cycle est utilisé afin de déduire le nombre d'unités de mappage de groupes de tranche dans le groupe de tranches 0 lorsque slice_group_map_type est égal à 3, 4 ou 5, comme spécifié par:

$$MapUnitsInSliceGroup0 = Min(slice\_group\_change\_cycle * SliceGroupChangeRate, PicSizeInMapUnits) \quad (7\text{-}31)$$

La valeur de slice_group_change_cycle est représentée dans le flux binaire par le nombre de bits suivant:

$$Ceil(Log2(PicSizeInMapUnits \div SliceGroupChangeRate + 1)) \quad (7\text{-}32)$$

La valeur de slice_group_change_cycle doit être dans la gamme de 0 à Ceil(PicSizeInMapUnits÷SliceGroupChangeRate), inclus.

7.4.3.1 Sémantique de remise en ordre de liste d'images de référence

Les éléments syntaxiques reordering_of_pic_nums_idc, abs_diff_pic_num_minus1 et long_term_pic_num spécifient le changement à utiliser afin de décoder la tranche en passant des listes d'images de référence initiales aux listes d'images de référence.

ref_pic_list_reordering_flag_l0 égal à 1 spécifie que la remise en ordre des éléments syntaxiques reordering_of_pic_nums_idc est présente afin de spécifier la liste 0 d'images de référence. ref_pic_list_reordering_flag_l0 égal à 0 spécifie que cet élément syntaxique n'est pas présent.

Lorsque ref_pic_list_reordering_flag_l0 est égal à 1, le nombre de fois où reordering_of_pic_nums_idc n'est pas égal à 3 à la suite de ref_pic_list_reordering_flag_l0 ne doit pas excéder num_ref_idx_l0_active_minus1 + 1.

Lorsque dans la liste initiale d'images de référence produite comme spécifié au § 8.2.4.2 RefPicList0[num_ref_idx_l0_active_minus1] est égal à "pas d'image de référence", ref_pic_list_reordering_flag_l0 doit être égal à 1 et reordering_of_pic_nums_idc ne doit pas être égal à 3 jusqu'à ce que dans la liste remise en ordre produite comme spécifié au § 8.2.4.3, RefPicList0[num_ref_idx_l0_active_minus1] ne soit pas égal à "pas d'image de référence".

ref_pic_list_reordering_flag_l1 égal à 1 spécifie que l'élément syntaxique reordering_of_pic_nums_idc est présent afin de spécifier la liste 1 d'images de référence. ref_pic_list_reordering_flag_l1 égal à 0 spécifie que cet élément syntaxique n'est pas présent.

Lorsque ref_pic_list_reordering_flag_l1 est égal à 1, le nombre de fois où reordering_of_pic_nums_idc n'est pas égal à 3 à la suite de ref_pic_list_reordering_flag_l1 ne doit pas excéder num_ref_idx_l1_active_minus1 + 1.

Lorsqu'on décode une tranche B et que RefPicList1[num_ref_idx_l1_active_minus1] dans la liste initiale d'images de référence produite comme spécifié au § 8.2.4.2 est égal à "pas d'image de référence", ref_pic_list_reordering_flag_l1 doit être égal à 1 et reordering_of_pic_nums_idc ne doit pas être égal à 3 jusqu'à ce que dans la liste remise en ordre produite comme spécifié au § 8.2.4.3, RefPicList1[num_ref_idx_l1_active_minus1] ne soit pas égal à "pas d'image de référence".

reordering_of_pic_nums_idc, conjointement avec abs_diff_pic_num_minus1 ou long_term_pic_num, spécifie laquelle des images de référence est remise en mappage. Les valeurs de reordering_of_pic_nums_idc sont spécifiées au Tableau 7-7. La valeur du premier reordering_of_pic_nums_idc qui suit immédiatement après ref_pic_list_reordering_flag_l0 ou ref_pic_list_reordering_flag_l1 ne doit pas être égale à 3.

Tableau 7-7 – Opérations de reordering_of_pic_nums_idc pour la remise en ordre des listes d'images de référence

reordering_of_pic_nums_idc	Remise en ordre spécifiée
0	abs_diff_pic_num_minus1 est présent et correspond à une différence à soustraire d'une valeur de prédiction du numéro d'image
1	abs_diff_pic_num_minus1 est présent et correspond à une différence à ajouter à une valeur de prédiction du numéro d'image
2	long_term_pic_num est présent et spécifie le numéro d'image à long terme pour une image de référence
3	Fin de boucle pour la remise en ordre de la liste initiale d'images de référence

abs_diff_pic_num_minus1 plus 1 spécifie la différence absolue entre le numéro d'image de l'image en cours de déplacement jusqu'à l'indice actuel dans la liste et la valeur de prédiction du numéro d'image. abs_diff_pic_num_minus1 doit être compris entre 0 et MaxPicNum –1. Les valeurs admises de abs_diff_pic_num_minus1 sont de plus restreintes comme spécifié au § 8.2.4.3.1.

long_term_pic_num spécifie le numéro d'image à long terme de l'image en cours déplacée vers l'indice actuel dans la liste. Lors du décodage d'une trame codée, long_term_pic_num doit être égal à un LongTermPicNum alloué à une des trames de référence ou paire de sous-trames de référence complémentaires marquées "utilisée comme référence à long terme". Lors du décodage d'une sous-trame codée, long_term_pic_num doit être égal à un LongTermPicNum alloué à une des sous-trames de référence marquées "utilisée comme référence à long terme".

7.4.3.2 Sémantique de tableau de prédiction de pondération

luma_log2_weight_denom est le logarithme de base 2 du dénominateur pour tous les facteurs de pondération luma. La valeur de luma_log2_weight_denom doit être dans la gamme de 0 à 7 inclus.

chroma_log2_weight_denom est le logarithme de base 2 du dénominateur pour tous les facteurs de pondération chroma. La valeur de chroma_log2_weight_denom doit être dans la gamme de 0 à 7 inclus.

luma_weight_l0_flag égal à 1 spécifie que les facteurs de pondération pour la prédiction de la liste 0 de composantes luma sont présents. luma_weight_l0_flag égal à 0 spécifie que ces facteurs de pondération ne sont pas présents.

luma_weight_l0[i **]** est le facteur de pondération appliqué à la valeur de prédiction luma pour la prédiction de liste 0 au moyen de RefPicList0[i]. Lorsque luma_weight_l0_flag est égal à 1, la valeur de luma_weight_l0[i] doit être dans la gamme de –128 à 127 inclus. Lorsque luma_weight_l0_flag est égal à 0, luma_weight_l0[i] doit être supposé égal à $2^{\text{luma\_log2\_weight\_denom}}$ pour RefPicList0[i].

luma_offset_l0[i **]** est le décalage supplémentaire appliqué à la valeur de prédiction luma pour la prédiction de liste 0 au moyen de RefPicList0[i]. La valeur de luma_offset_l0[i] doit être dans la gamme de –128 à 127 inclus. Lorsque luma_weight_l0_flag est égal à 0, luma_offset_l0[i] doit être supposé égal à 0 pour RefPicList0[i].

chroma_weight_l0_flag égal à 1 spécifie que les facteurs de pondération pour les valeurs de prédiction chroma de prédiction de liste 0 sont présents. chroma_weight_l0_flag égal à 0 spécifie que ces facteurs de pondération ne sont pas présents.

chroma_weight_l0[i **][** j **]** est le facteur de pondération appliqué aux valeurs de prédiction chroma pour la prédiction de liste 0 au moyen de RefPicList0[i] avec j égal à 0 pour Cb et j égal à 1 pour Cr. Lorsque chroma_weight_l0_flag est égal à 1, la valeur de chroma_weight_l0[i][j] doit être dans la gamme de –128 à 127 inclus. Lorsque chroma_weight_l0_flag est égal à 0, chroma_weight_l0[i][j] doit être supposé égal à $2^{\text{chroma\_log2\_weight\_denom}}$ pour RefPicList0[i].

chroma_offset_l0[i **][** j **]** est le décalage supplémentaire appliqué aux valeurs de prédiction chroma pour la prédiction de liste 0 au moyen de RefPicList0[i] avec j égal à 0 pour Cb et j égal à 1 pour Cr. La valeur de chroma_offset_l0[i][j] doit être dans la gamme de –128 à 127 inclus. Lorsque chroma_weight_l0_flag est égal à 0, chroma_offset_l0[i][j] doit être supposé égal à 0 pour RefPicList0[i].

luma_weight_l1_flag, **luma_weight_l1**, **luma_offset_l1**, **chroma_weight_l1_flag**, **chroma_weight_l1**, **chroma_offset_l1** ont la même sémantique que respectivement luma_weight_l0_flag, luma_weight_l0, luma_offset_l0, chroma_weight_l0_flag, chroma_weight_l0, chroma_offset_l0, avec l0, list 0 et List0 remplacés respectivement par l1, list 1 et List1.

7.4.3.3 Sémantique de marquage d'image de référence décodée

Les éléments syntaxiques no_output_of_prior_pics_flag, long_term_reference_flag, adaptive_ref_pic_marking_mode_flag, memory_management_control_operation, difference_of_pic_nums_minus1,

long_term_frame_idx, long_term_pic_num et max_long_term_frame_idx_plus1 spécifient le marquage des images de référence.

Le marquage d'une image de référence peut être "inutilisée comme référence", "utilisée comme référence à court terme" ou "utilisée comme référence à long terme", mais seulement un parmi ces trois. Lorsqu'une image de référence est référencée comme étant marquée "utilisée comme référence", cela se rapporte collectivement à une image marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme" (mais pas les deux à la fois). Une image de référence qui est marquée "utilisée comme référence à court terme" est référencée comme étant une image de référence à court terme. Une image de référence qui est marquée "utilisée comme référence à long terme" est référencée comme étant une image de référence à long terme.

L'élément syntaxique adaptive_ref_pic_marking_mode_flag et le contenu de la structure syntaxique de marquage d'image de référence décodée doivent être identiques pour toutes les tranches codées d'une image codée.

La catégorie syntaxique de la structure syntaxique de marquage d'image de référence décodée doit être déduite comme suit:

– si la structure syntaxique de marquage d'image de référence décodée est dans un en-tête de tranche, la catégorie syntaxique de la structure syntaxique de marquage d'image de référence décodée doit être supposée égale à 2;

– Sinon (si la structure syntaxique de marquage d'image de référence décodée est dans un message SEI de répétition de marquage d'image de référence décodée comme spécifié à l'Annexe D), la catégorie syntaxique de la structure syntaxique de marquage d'image de référence décodée doit être supposée égale à 5.

no_output_of_prior_pics_flag spécifie comment sont traitées les images précédemment décodées dans la mémoire tampon d'image décodée après le décodage d'uneimage à rafraîchissement IDR. Voir l'Annexe C. Lorsque l'image à rafraîchissement IDR est la premièreimage à rafraîchissement IDR dans le flux binaire, la valeur de no_output_of_prior_pics_flag n'a pas d'effet sur le processus de décodage. Lorsque l'image à rafraîchissement IDR n'est pas la premièreimage à rafraîchissement IDR dans le flux binaire et que la valeur de PicWidthInMbs, FrameHeightInMbs ou max_dec_frame_buffering, déduite de l'ensemble actif de paramètres de séquence, est différente de la valeur de PicWidthInMbs, FrameHeightInMbs ou max_dec_frame_buffering, déduite de l'ensemble actif de paramètres de séquence pour la séquence précédente, no_output_of_prior_pics_flag peut être supposé égal à 1 par le décodeur, sans considération de la valeur réelle de no_output_of_prior_pics_flag.

long_term_reference_flag égal à 0 spécifie que la variable MaxLongTermFrameIdx est mise égale à "pas d'indices de trame à long terme" et que l'image à rafraîchissement IDR est marquée "utilisée comme référence à court terme". long_term_reference_flag égal à 1 spécifie que la variable MaxLongTermFrameIdx est mise égale à 0 et que l'image à rafraîchissement IDR en cours est marquée "utilisée comme référence à long terme" et que LongTermFrameIdx est alloué égal à 0. Lorsque num_ref_frames est égal à 0, long_term_reference_flag doit être égal à 0.

adaptive_ref_pic_marking_mode_flag choisit le mode de marquage d'image de référence de l'image décodée en cours comme spécifié au Tableau 7-8. adaptive_ref_pic_marking_mode_flag doit être égal à 1 lorsque le nombre des trames, des paires de sous-trames complémentaires et des sous-trames non appariées qui sont actuellement marquées "utilisée comme référence à long terme" est égal à Max (num_ref_frames, 1).

Tableau 7-8 – Interprétation de adaptive_ref_pic_marking_mode_flag

adaptive_ref_pic_marking_mode_flag	Mode de marquage d'image de référence spécifié
0	Mode de marquage d'image de référence par fenêtre glissante: mode de marquage donnant un mécanisme de premier entré, premier sorti pour les images de référence à court terme
1	Mode de marquage d'image de référence adaptatif: mode de marquage d'image de référence fournissant des éléments syntaxiques afin de spécifier le marquage des images de référence "inutilisée comme référence" et afin d'allouer des indices de trame à long terme

memory_management_control_operation spécifie une opération de contrôle qui doit être appliquée afin d'affecter le marquage d'image de référence. L'élément syntaxique memory_management_control_operation est suivi des données nécessaires pour l'opération spécifiée par la valeur de memory_management_control_operation. Les valeurs et opérations de contrôle associées à memory_management_control_operation sont spécifiées au Tableau 7-9. Les éléments syntaxiques memory_management_control_operation sont traités par le processus de décodage dans l'ordre dans lequel ils figurent dans l'en-tête de tranche, et les contraintes sémantiques exprimées pour chaque memory_management_control_operation s'appliquent à l'emplacement précis, dans cet ordre, où ce memory_management_control_operation est traité.

Pour l'interprétation de l'élément memory_management_control_operation, le terme *image de référence* est interprété comme suit.

– Si l'image actuelle est une trame, le terme *image de référence* se rapporte soit à une trame de référence ou à une paire de sous-trames de référence complémentaires.

– Sinon (si l'image actuelle est une sous-trame), le terme *image de référence* se rapporte soit à une sous-trame de référence ou à une sous-trame d'une trame de référence.

L'élément memory_management_control_operation ne doit pas être égal à 1 dans un en-tête de tranche à moins que l'image de référence spécifiée ne soit marquée comme étant "utilisée comme référence à court terme" lorsque l'élément memory_management_control_operation est traité par le processus de décodage.

L'élément memory_management_control_operation ne doit pas être égal à 2 dans un en-tête de tranche à moins que le numéro spécifié d'image à long terme ne se rapporte à une image de référence qui est marquée comme étant "utilisée comme référence à long terme" lorsque l'élément memory_management_control_operation est traité par le processus de décodage.

L'élément memory_management_control_operation ne doit pas être égal à 3 dans un en-tête de tranche à moins que l'image de référence spécifiée ne soit marquée comme étant "utilisée comme référence à court terme" lorsque l'élément memory_management_control_operation est traité par le processus de décodage.

memory_management_control_operation ne doit pas être égal à 3 ou 6 si la valeur de la variable MaxLongTermFrameIdx est égale à "pas d'indices de trames à long terme" lorsque l'élément memory_management_control_operation est traité par le processus de décodage.

Il ne doit pas y avoir plus d'un seul élément memory_management_control_operation égal à 4 dans un en-tête de tranche.

Il ne doit pas y avoir plus d'un seul élément memory_management_control_operation égal à 5 dans un en-tête de tranche.

Il ne doit pas y avoir plus d'un seul élément memory_management_control_operation égal à 6 dans un en-tête de tranche.

memory_management_control_operation ne doit pas être égal à 5 dans un en-tête de tranche à moins qu'aucun élément memory_management_control_operation dans la gamme de 1 à 3 ne soit présent dans la même structure syntaxique de marquage d'image de référence décodée.

Un élément memory_management_control_operation égal à 5 ne doit pas suivre un élément memory_management_control_operation égal à 6 dans le même en-tête de tranche.

Lorsqu'un élément memory_management_control_operation égal à 6 est présent, tout memory_management_control_operation égal à 2, 3 ou 4 qui suit l'élément memory_management_control_operation égal à 6 dans le même en-tête de tranche ne doit pas spécifier pour l'image en cours la valeur de marquage "inutilisée comme référence".

NOTE 1 – Ces contraintes interdisent toute combinaison de plusieurs éléments syntaxiques memory_management_control_operation qui spécifieraient pour l'image en cours la valeur de marquage "inutilisée comme référence". Toutefois, d'autres combinaisons d'éléments syntaxiques memory_management_control_operation sont autorisées, qui peuvent affecter l'état de marquage d'autres images de référence plusieurs fois dans le même en-tête de tranche. En particulier, un élément memory_management_control_operation égal à 3 qui spécifie qu'un indice de trame à long terme soit alloué à une image de référence à court terme donnée peut être suivi dans le même en-tête de tranche d'un élément memory_management_control_operation égal à 2, 3, 4 ou 6 qui spécifie que la même image de référence soit ultérieurement marquée "inutilisée comme référence".

Tableau 7-9 – Valeurs d'opération de commande de gestion de mémoire (memory_management_control_operation)

memory_management_control_operation	Opération de commande de gestion de mémoire
0	Met fin à la boucle de l'élément syntaxique memory_management_control_operation
1	Marque une image de référence à court terme "inutilisée comme référence"
2	Marque une image de référence à long terme "inutilisée comme référence"
3	Marque une image de référence à court terme "utilisée comme référence à long terme" et lui attribue un indice de trame à long terme
4	Spécifie l'indice maximal de trame à long terme et marque comme étant "inutilisée comme référence" toutes les images de référence à long terme ayant des indices de trame à long terme supérieurs à la valeur maximale
5	Marque toutes les images de référence "inutilisée comme référence" et met la variable MaxLongTermFrameIdx à la valeur "pas d'indices de trame à long terme"
6	Marque l'image actuelle comme étant "utilisée comme référence à long terme" et lui attribue un indice de trame à long terme

Lorsqu'on décode une sous-trame et qu'une commande memory_management_control_operation égale à 3 alloue un indice de trame à long terme à une sous-trame qui fait partie d'une trame de référence à court terme ou d'une paire de sous-trames de référence à court terme complémentaires, une autre commande memory_management_control_operation doit être présente dans la même structure syntaxique de marquage d'image de référence décodée afin d'allouer le même indice de trame à long terme à l'autre sous-trame de la même trame ou paire de sous-trames de référence complémentaires.

NOTE 2 – L'exigence ci-dessus doit toujours être satisfaite, même lorsque la sous-trame citée en référence par l'élément memory_management_control_operation égal à 3 est ultérieurement marquée "inutilisée comme référence" (par exemple lorsqu'un élément memory_management_control_operation égal à 2 est présent dans le même en-tête de tranche que celui qui provoque le marquage de la sous-trame comme étant "inutilisée comme référence").

Lorsque la première sous-trame (dans l'ordre de décodage) d'une paire de sous-trames de référence complémentaires inclut un long_term_reference_flag égal à 1 ou une commande memory_management_control_operation égale à 6, la structure syntaxique de marquage d'image de référence décodée pour l'autre sous-trame de la paire de sous-trames de référence complémentaires doit contenir une commande memory_management_control_operation égale à 6 qui alloue le même indice de trame à long terme à l'autre sous-trame.

NOTE 3 – L'exigence ci-dessus doit toujours être satisfaite, même lorsque la première sous-trame de la paire de sous-trames de référence complémentaires est ultérieurement marquée "inutilisée comme référence" (par exemple, lorsqu'un élément memory_management_control_operation égal à 2 est présent dans l'en-tête de tranche du second sous-trame que celui qui provoque le marquage de la première sous-trame comme étant "inutilisée comme référence").

difference_of_pic_nums_minus1 est utilisé (avec memory_management_control_operation égal à 3 ou 1) afin d'allouer un indice de trame à long terme à une image de référence à court terme ou afin de marquer une image de référence à court terme "inutilisée comme référence". Lorsque l'élément memory_management_control_operation associé est traité par le processus de décodage, le numéro d'image résultant, déduit de difference_of_pic_nums_minus1, doit être un numéro d'image alloué à l'une des images de référence marquées "utilisée comme référence" et non alloué précédemment à un indice de trame à long terme.

Le numéro d'image résultant est soumis aux contraintes suivantes.

– Si field_pic_flag est égal à 0, le numéro d'image résultant doit être inclus dans l'ensemble des numéros d'image alloués aux trames de référence ou paires de sous-trames de référence complémentaires.

  NOTE 4 – Lorsque field_pic_flag est égal à 0, le numéro d'image résultant doit être un numéro d'image alloué à une paire de sous-trames de référence complémentaires dont les deux sous-trames sont marquées "utilisée comme référence" ou alloué à une trame dont les deux sous-trames sont marquées "utilisée comme référence". En particulier, lorsque field_pic_flag est égal à 0, le marquage d'une sous-trame ou d'une trame non appariée dont une seule sous-trame est marquée "utilisée comme référence" ne peut être affecté par un élément memory_management_control_operation égal à 1.

– Sinon (si field_pic_flag est égal à 1), le numéro d'image résultant doit être contenu dans l'ensemble des numéros d'image alloués aux sous-trames de référence.

long_term_pic_num est utilisé (avec memory_management_control_operation égal à 2) afin de marquer une image de référence à long terme "inutilisée comme référence". Lorsque l'élément memory_management_control_operation est traité par le processus de décodage, long_term_pic_num doit être égal à un numéro d'image à long terme alloué à une des images de référence qui est actuellement marquée "utilisée comme référence à long terme".

Le numéro d'image à long terme résultant est soumis aux contraintes suivantes.

– Si field_pic_flag est égal à 0, le numéro d'image à long terme résultant doit être inclus dans l'ensemble de numéros d'image à long terme alloués aux trames de référence ou aux paires de sous-trames de référence complémentaires.

  NOTE 5 – Lorsque field_pic_flag est égal à 0, le numéro d'image à long terme résultant doit être un numéro d'image à long terme alloué à une paire de sous-trames de référence complémentaires dont les deux sous-trames sont marquées "utilisée comme référence" ou à une trame dont les deux sous-trames sont marquées "utilisée comme référence". En particulier, lorsque field_pic_flag est égal à 0, le marquage d'une sous-trame ou d'une trame non appariée dont une seule sous-trame est marquée "utilisée comme référence" ne peut être affecté par un élément memory_management_control_operation égal à 2.

– Sinon (si field_pic_flag est égal à 1), le numéro d'image à long terme résultant doit être inclus dans l'ensemble de numéros d'image à long terme alloués aux sous-trames de référence.

long_term_frame_idx est utilisé (avec memory_management_control_operation égal à 3 ou 6) afin d'allouer un indice de trame à long terme à une image. Lorsque l'élément memory_management_control_operation est traité par le processus de décodage, la valeur de long_term_frame_idx doit être dans la gamme de 0 à MaxLongTermFrameIdx, inclus.

max_long_term_frame_idx_plus1 moins 1 spécifie la valeur maximale de l'indice de trame à long terme admise pour les images de référenceà long terme (jusqu'à réception d'une autre valeur de max_long_term_frame_idx_plus1). La valeur de max_long_term_frame_idx_plus1 doit être dans la gamme de 0 à num_ref_frames inclus.

7.4.4 Sémantique de données de tranche

cabac_alignment_one_bit est un bit égal à 1.

mb_skip_run spécifie le nombre de macroblocs consécutifs sautés pour lesquels, lorsqu'on décode une tranche P ou SP, on doit supposer que le type mb_type est P_Skip et le type de macrobloc est désigné collectivement comme type de macrobloc P ou pour lesquels, lorsqu'on décode une tranche B, on doit supposer que le mb_type est B_Skip et le type de macrobloc est désigné collectivement comme type de macrobloc B. La valeur de mb_skip_run doit être dans la gamme de 0 à PicSizeInMbs – CurrMbAddr inclus.

mb_skip_flag égal à 1 spécifie que pour le macrobloc en cours, lorsqu'on décode une tranche P ou SP, on doit supposer que le mb_type est P_Skip et que le type de macrobloc est désigné collectivement comme type de macrobloc P ou que pour lequel, lorsqu'on décode une tranche B, on doit supposer que le mb_type est B_Skip et que le type de macrobloc est désigné collectivement comme type de macrobloc B. mb_skip_flag égal à 0 spécifie que le macrobloc en cours n'est pas sauté.

mb_field_decoding_flag égal à 0 spécifie que la paire de macroblocs en cours est une paire de macroblocs de trame. mb_field_decoding_flag égal à 1 spécifie que la paire de macroblocs est une paire de macroblocs de sous-trame. Les deux macroblocs d'une paire de macroblocs de trame sont désignés dans le texte comme *macroblocs de trame*, tandis que les deux macroblocs d'une paire de macroblocs de sous-trame sont désignés dans le texte comme *macroblocs de sous-trame*.

Lorsque mb_field_decoding_flag n'est pas présent pour l'un ou l'autre macrobloc d'une paire de macroblocs, la valeur de mb_field_decoding_flag est déduite comme suit.

– S'il y a une paire de macroblocs dans le voisinage immédiatement à gauche de la paire de macroblocs en cours dans la même tranche, la valeur de mb_field_decoding_flag doit être supposée égale à la valeur de mb_field_decoding_flag pour la paire de macroblocs du voisinage immédiatement à gauche de la paire de macroblocs en cours;

– Sinon, s'il n'y a pas de paire de macroblocs dans le voisinage immédiatement à gauche de la paire de macroblocs en cours dans la même tranche et s'il y a une paire de macroblocs dans le voisinage immédiatement au-dessus de la paire de macroblocs en cours dans la même tranche, la valeur de mb_field_decoding_flag doit être supposée égale à la valeur de mb_field_decoding_flag pour la paire de macroblocs dans le voisinage immédiatement au-dessus de la paire de macroblocs en cours;

– Sinon (s'il n'y a pas de paire de macroblocs dans le voisinage ni immédiatement à gauche ni immédiatement au-dessus de la paire de macroblocs en cours dans la même tranche), la valeur de mb_field_decoding_flag doit être supposée égale à 0.

end_of_slice_flag égal à 0 spécifie qu'un autre macrobloc suit dans la tranche. end_of_slice_flag égal à 1 spécifie la fin de la tranche et qu'aucun autre macrobloc ne suit.

La fonction NextMbAddress( ) utilisée dans le tableau de syntaxe des données de tranche est spécifiée au § 8.2.2.

7.4.5 Sémantique de couche de macrobloc

mb_type spécifie le type de macrobloc. La sémantique de mb_type dépend du type de tranche.

Les tableaux et la sémantique sont spécifiés pour les divers types de macrobloc pour les tranches I, SI, P, SP et B. Chaque tableau présente la valeur de mb_type, le nom de mb_type, le nombre de subdivisions de macrobloc utilisées (donné par la fonction NumMbPart(mb_type)), le mode de prédiction du macrobloc (lorsqu'il n'est pas subdivisé) ou la première subdivision (donnée par la fonction MbPartPredMode(mb_type, 0)) et le mode de prédiction de la seconde subdivision (donnée par la fonction MbPartPredMode(mb_type, 1)). Lorsqu'une valeur n'est pas applicable, elle est mentionnée par "na". Dans le texte, la valeur de mb_type peut être mentionnée comme le type de macrobloc et une valeur X de MbPartPredMode( ) peut être mentionnée dans le texte par "mode de prévision de macrobloc (subdivision) X" ou comme "macroblocs de prédiction X".

Le Tableau 7-10 montre les types collectifs de macrobloc admis pour slice_type.

NOTE 1 – Quelques types de macrobloc avec le mode de prédiction Pred_L0 sont classés dans le type de macrobloc B.

Tableau 7-10 – Types collectifs de macrobloc admis pour slice_type

slice_type	Types collectifs de macrobloc admis
(tranche) I	(types de macrobloc) I (voir Tableau 7-11)
(tranche) P	(types de macrobloc) P (voir Tableau 7-13) et I (voir Tableau 7-11)
(tranche) B	(types de macrobloc) B (voir Tableau 7-14) et I (voir Tableau 7-11)
(tranche) SI	(types de macrobloc) SI (voir Tableau 7-12) et I (voir Tableau 7-11)
(tranche) SP	(types de macrobloc) P (voir Tableau 7-13) et I (voir Tableau 7-11)

transform_size_8x8_flag égal à 1 spécifie que, pour le macrobloc actuel, le processus de décodage des coefficients de transformée et le processus de construction d'image avant processus de filtre de déblocage pour blocs de 8x8 échantillons résiduels doivent être invoqués pour les échantillons luma. Un élément transform_size_8x8_flag égal à 0 spécifie que, pour le macrobloc actuel, le processus de décodage des coefficients de transformée et le processus de construction d'image avant processus de filtre de déblocage pour blocs de 4x4 échantillons résiduels doit être invoqué pour les échantillons luma. Lorsque le fanion transform_size_8x8_flag n'est pas présent dans le flux binaire, sa valeur doit être supposée égale à 0.

NOTE 2 – Lorsque le mode de prédiction du macrobloc actuel MbPartPredMode(mb_type, 0) est égal à Intra_16x16, le fanion transform_size_8x8_flag n'est pas présent dans le flux binaire et est alors présumé égal à 0.

Lorsque l'élément sub_mb_type[mbPartIdx] (voir § 7.4.5.2) est présent dans le flux binaire pour tous les blocs 8x8 indexés par mbPartIdx = 0..3, la variable noSubMbPartSizeLessThan8x8Flag indique si, pour chacun des quatre blocs 8x8, les variables correspondantes SubMbPartWidth(sub_mb_type[mbPartIdx]) et SubMbPartHeight(sub_mb_type[mbPartIdx]) sont toutes deux égales à 8.

NOTE 3 – Lorsque le fanion noSubMbPartSizeLessThan8x8Flag est égal à 0 et lorsque le type de macrobloc actuel n'est pas égal à I_NxN, le fanion transform_size_8x8_flag n'est pas présent dans le flux binaire et est alors présumé égal à 0.

Les types de macrobloc auxquels on peut faire référence collectivement sous le nom de *types de macrobloc I* sont spécifiés au Tableau 7-11.

Les types de macrobloc pour les tranches I sont tous des types de macrobloc I.

Tableau 7-11 – Types de macrobloc pour tranches I

mb_type (Type de macro-bloc)	Nom du type de macrobloc mb_type	transform_size_8x8_flag	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_NxN	0	Intra_4x4	na	Equation 7-33	Equation 7-33
0	I_NxN	1	Intra_8x8	na	Equation 7-33	Equation 7-33
1	I_16x16_0_0_0	na	Intra_16x16	0	0	0
2	I_16x16_1_0_0	na	Intra_16x16	1	0	0
3	I_16x16_2_0_0	na	Intra_16x16	2	0	0
4	I_16x16_3_0_0	na	Intra_16x16	3	0	0
5	I_16x16_0_1_0	na	Intra_16x16	0	1	0
6	I_16x16_1_1_0	na	Intra_16x16	1	1	0
7	I_16x16_2_1_0	na	Intra_16x16	2	1	0
8	I_16x16_3_1_0	na	Intra_16x16	3	1	0
9	I_16x16_0_2_0	na	Intra_16x16	0	2	0
10	I_16x16_1_2_0	na	Intra_16x16	1	2	0
11	I_16x16_2_2_0	na	Intra_16x16	2	2	0
12	I_16x16_3_2_0	na	Intra_16x16	3	2	0
13	I_16x16_0_0_1	na	Intra_16x16	0	0	15
14	I_16x16_1_0_1	na	Intra_16x16	1	0	15
15	I_16x16_2_0_1	na	Intra_16x16	2	0	15
16	I_16x16_3_0_1	na	Intra_16x16	3	0	15
17	I_16x16_0_1_1	na	Intra_16x16	0	1	15
18	I_16x16_1_1_1	na	Intra_16x16	1	1	15
19	I_16x16_2_1_1	na	Intra_16x16	2	1	15
20	I_16x16_3_1_1	na	Intra_16x16	3	1	15
21	I_16x16_0_2_1	na	Intra_16x16	0	2	15
22	I_16x16_1_2_1	na	Intra_16x16	1	2	15
23	I_16x16_2_2_1	na	Intra_16x16	2	2	15
24	I_16x16_3_2_1	na	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na	na

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-11:

I_NxN: nom mnémonique pour mb_type égal à 0 avec MbPartPredMode(mb_type, 0) égal à Intra_4x4 ou Intra_8x8.

I_16x16_0_0_0, I_16x16_1_0_0, I_16x16_2_0_0, I_16x16_3_0_0, I_16x16_0_1_0, I_16x16_1_1_0, I_16x16_2_1_0, I_16x16_3_1_0, I_16x16_0_2_0, I_16x16_1_2_0, I_16x16_2_2_0, I_16x16_3_2_0, I_16x16_0_0_1, I_16x16_1_0_1, I_16x16_2_0_1, I_16x16_3_0_1, I_16x16_0_1_1, I_16x16_1_1_1, I_16x16_2_1_1, I_16x16_3_1_1, I_16x16_0_2_1, I_16x16_1_2_1, I_16x16_2_2_1, I_16x16_3_2_1: le macrobloc est codé comme un macrobloc de mode de prédiction Intra_16x16.

A chaque macrobloc de prédiction Intra_16x16 est alloué un Intra16x16PredMode, qui spécifie le mode de prédiction Intra_16x16. CodedBlockPatternChroma contient la valeur du schéma de bloc codé pour chroma comme spécifié au Tableau 7-15. Lorsque chroma_format_idc est égal à 0, la variable CodedBlockPatternChroma doit être égale à 0. CodedBlockPatternLuma spécifie si des niveaux de coefficient de transformée AC différents de zéro sont présents pour la composante luma. CodedBlockPatternLuma égal à 0 spécifie que tous les niveaux de coefficient de transformée AC

dans la composante luma du macrobloc sont égaux à 0. CodedBlockPatternLuma égal à 15 spécifie qu'au moins un des niveaux de coefficient de transformée AC figurant dans la composante luma du macrobloc est différent de 0, requérant un balayage des niveaux de coefficient de transformée AC pour tous les 16 des blocs 4x4 dans le bloc 16x16.

Intra_4x4 spécifie le mode de prédiction de macrobloc et spécifie que le processus de prédiction Intra_4x4 est invoqué comme spécifié au § 8.3.1. Intra_4x4 est un mode de prédiction intramacrobloc.

Intra_8x8 spécifie le mode de prédiction de macrobloc et spécifie que le processus de prédiction Intra_8x8 est invoqué comme spécifié au § 8.3.2. Intra_8x8 est un mode de prédiction intramacrobloc.

Intra_16x16 spécifie le mode de prédiction de macrobloc et spécifie que le processus de prédiction Intra_16x16 est invoqué comme spécifié au § 8.3.3. Intra_16x16 est un mode de prédiction intramacrobloc.

Pour un macrobloc codé avec mb_type égal à I_PCM, on doit supposer le mode de prédiction intramacrobloc.

Un type de macrobloc auquel on peut se référer sous le nom de *type de macrobloc SI* est spécifié au Tableau 7-12.

Les types de macrobloc pour les tranches SI sont spécifiés au Tableau 7-12 et au Tableau 7-11. La valeur 0 de mb_type est spécifiée au Tableau 7-12 et les valeurs 1 à 26 de mb_type sont spécifiées au Tableau 7-11, indexées en soustrayant 1 de la valeur de mb_type.

Tableau 7-12 – Type de macrobloc avec la valeur 0 pour les tranches SI

mb_type	Nom de mb_type	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	SI	Intra_4x4	na	Equation 7-33	Equation 7-33

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-12. Le macrobloc SI est codé comme macrobloc de prédiction Intra_4x4.

Les types de macrobloc qu'on peut mentionner collectivement comme types de macrobloc P sont spécifiés au Tableau 7-13.

Les types de macrobloc pour les tranches P et SP sont spécifiés au Tableau 7-13 et au Tableau 7-11. Les valeurs de mb_type 0 à 4 sont spécifiées au Tableau 7-13 et les valeurs de mb_type de 5 à 30 sont spécifiées au Tableau 7-11, indexées en soustrayant 5 de la valeur de mb_type.

Tableau 7-13 – Valeurs de 0 à 4 de type de macrobloc pour les tranches P et SP

mb_type	Nom de mb_type	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	P_L0_16x16	1	Pred_L0	na	16	16
1	P_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
2	P_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
3	P_8x8	4	na	na	8	8
4	P_8x8ref0	4	na	na	8	8
Supposée	P_Skip	1	Pred_L0	na	16	16

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-13.

– P_L0_16x16: les échantillons du macrobloc sont prédits avec une subdivision de macrobloc luma d'une taille de 16x16 échantillons luma et les échantillons chroma associés.

– P_L0_L0_MxN, avec MxN remplacé par 16x8 ou 8x16: les échantillons du macrobloc sont prédits au moyen de deux subdivisions luma d'une taille de MxN égale à 16x8 ou deux subdivisions luma d'une taille de MxN égale à 8x16 et les échantillons chroma associés, respectivement.

– P_8x8: pour chaque sous-macrobloc est présent un élément syntaxique (sub_mb_type) additionnel dans le flux binaire qui spécifie le type du sous-macrobloc correspondant (voir le § 7.4.5.2).

– P_8x8ref0: a la même sémantique que P_8x8 mais il n'y a pas d'élément syntaxique présent pour l'indice de référence (ref_idx_l0) dans le flux binaire et ref_idx_l0[mbPartIdx] doit être supposé égal à 0 pour tous les sous-macroblocs du macrobloc (avec des indices mbPartIdx égaux à 0..3).

– P_Skip: pas d'autres données présentes pour le macrobloc dans le flux binaire.

La sémantique suivante est allouée au mode de prédiction de macroblocs (MbPartPredMode()) dans le Tableau 7-13.

– Pred_L0: spécifie que le processus d'interprédiction est invoqué au moyen de la prédiction de liste 0. Pred_L0 est un mode de prédiction intermacrobloc.

Les types de macrobloc qu'on peut mentionner collectivement comme types de macrobloc B sont spécifiés au Tableau 7-14.

Les types de macrobloc pour les tranches B sont spécifiés au Tableau 7-14 et au Tableau 7-11. Les valeurs de mb_type de 0 à 22 sont spécifiées au Tableau 7-14 et les valeurs de mb_type de 23 à 48 sont spécifiées au Tableau 7-11, indexées en soustrayant 23 de la valeur de mb_type.

Tableau 7-14 – Valeurs de type de macrobloc de 0 à 22 pour tranches B

mb_type	Nom de mb_type	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	B_Direct_16x16	na	Direct	na	8	8
1	B_L0_16x16	1	Pred_L0	na	16	16
2	B_L1_16x16	1	Pred_L1	na	16	16
3	B_Bi_16x16	1	BiPred	na	16	16
4	B_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
5	B_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
6	B_L1_L1_16x8	2	Pred_L1	Pred_L1	16	8
7	B_L1_L1_8x16	2	Pred_L1	Pred_L1	8	16
8	B_L0_L1_16x8	2	Pred_L0	Pred_L1	16	8
9	B_L0_L1_8x16	2	Pred_L0	Pred_L1	8	16
10	B_L1_L0_16x8	2	Pred_L1	Pred_L0	16	8
11	B_L1_L0_8x16	2	Pred_L1	Pred_L0	8	16
12	B_L0_Bi_16x8	2	Pred_L0	BiPred	16	8
13	B_L0_Bi_8x16	2	Pred_L0	BiPred	8	16
14	B_L1_Bi_16x8	2	Pred_L1	BiPred	16	8
15	B_L1_Bi_8x16	2	Pred_L1	BiPred	8	16
16	B_Bi_L0_16x8	2	BiPred	Pred_L0	16	8
17	B_Bi_L0_8x16	2	BiPred	Pred_L0	8	16
18	B_Bi_L1_16x8	2	BiPred	Pred_L1	16	8
19	B_Bi_L1_8x16	2	BiPred	Pred_L1	8	16
20	B_Bi_Bi_16x8	2	BiPred	BiPred	16	8
21	B_Bi_Bi_8x16	2	BiPred	BiPred	8	16
22	B_8x8	4	na	na	8	8
Supposée	B_Skip	na	Direct	na	8	8

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-14:

– B_Direct_16x16: il n'y a pas de différence de vecteur cinétique ou d'indice de référence présent pour le macrobloc dans le flux binaire. Les fonctions MbPartWidth(B_Direct_16x16) et MbPartHeight(B_Direct_16x16) sont utilisées dans le processus de déduction pour les indices des vecteurs cinétiques et de trame de référence au § 8.4.1 pour la prédiction de mode direct.

– B_X_16x16 avec X remplacé par L0, L1 ou Bi: les échantillons du macrobloc sont prédits avec une subdivision de macrobloc luma d'une taille d'échantillons luma 16x16 et les échantillons chroma associés. Pour un macrobloc du type B_X_16x16 avec X remplacé par L0 ou L1, une différence de vecteur cinétique et un indice de référence sont présents dans le flux binaire pour le macrobloc. Pour un macrobloc de type B_X_16x16 avec X remplacé par Bi, deux différences de vecteur cinétique et deux indices de référence sont présents dans le flux binaire pour le macrobloc.

– B_X0_X1_MxN, avec X0, X1 se rapportant à la première et la seconde subdivision de macrobloc et remplacés par L0, L1 ou Bi et MxN remplacé par 16x8 ou 8x16: les échantillons du macrobloc sont prédits au moyen de respectivement deux subdivisions luma d'une taille MxN égale à 16x8 ou deux subdivisions luma d'une taille de MxN égale à 8x16 et les échantillons chroma associés. Pour une subdivision de macrobloc X0 ou X1 avec X0 ou X1 remplacés par L0 ou L1, une différence de vecteur cinétique et un indice de référence sont présents dans le flux

binaire. Pour une subdivision de macrobloc X0 ou X1 avec X0 ou X1 remplacé par Bi, deux différences de vecteur cinétique et deux indices de référence sont présents dans le flux binaire pour la subdivision de macrobloc.

– B_8x8: pour chaque sous-macrobloc est présent un élément syntaxique (sub_mb_type) additionnel envoyé dans le flux binaire, qui spécifie le type du sous-macrobloc correspondant (voir le § 7.4.5.2).

– B_Skip: il n'y a pas d'autres données présentes pour le macrobloc dans le flux binaire. Les fonctions MbPartWidth(B_Skip) et MbPartHeight(B_Skip) sont utilisées dans le processus de déduction pour les vecteurs cinétiques et les indices de trame de référence au § 8.4.1 pour la prédiction de mode direct.

La sémantique suivante est allouée aux modes de prédiction de macrobloc (MbPartPredMode()) au Tableau 7-14.

– Direct: il n'y a pas de différence de vecteur cinétique ou d'indice de référence présent pour le macrobloc (dans le cas de B_Skip ou B_Direct_16x16) dans le flux binaire. Direct est un mode d'interprédiction de macrobloc.

– Pred_L0: voir la sémantique pour le Tableau 7-13.

– Pred_L1: spécifie que le processus d'interprédiction est invoqué au moyen de la prédiction de liste 1. Pred_L1 est un mode d'interprédiction de macrobloc.

– BiPred: spécifie que le processus d'interprédiction est invoqué au moyen de la prédiction de liste 0 et de liste 1. BiPred est un mode d'interprédiction de macrobloc.

pcm_alignment_zero_bit est un bit égal à 0.

pcm_sample_luma[i] est une valeur d'échantillon. Les pcm_sample_luma[i] premières valeurs représentent des valeurs d'échantillons luma dans le balayage matriciel du macrobloc. Le nombre de bits utilisés afin de représenter chacun de ces échantillons est $BitDepth_Y$. Lorsque profile_idc n'est pas égal à 100, 110, 122, ou 144, pcm_sample_luma[i] ne doit pas être égal à 0.

pcm_sample_chroma[i] est une valeur d'échantillon. Les MbWidthC * MbHeightC pcm_sample_chroma[i] premières valeurs représentent des valeurs d'échantillon Cb dans le balayage matriciel du macrobloc et les MbWidthC * MbHeightC pcm_sample_chroma[i] valeurs restantes représentent des valeurs d'échantillon Cr dans le balayage matriciel du macrobloc. Le nombre de bits utilisés afin de représenter chacun de ces échantillons est $BitDepth_C$. Lorsque profile_idc n'est pas égal à 100, 110, 122, ou 144, pcm_sample_chroma[i] ne doit pas être égal à 0.

coded_block_pattern spécifie lequel des quatre blocs 8x8 – luma et lequel des blocs chroma associés d'un macrobloc– peuvent contenir des niveaux de coefficient de transformée différents de zéro. Pour les macroblocs dont le mode de prédiction n'est pas égal à Intra_16x16, coded_block_pattern est présent dans le flux binaire et les variables CodedBlockPatternLuma et CodedBlockPatternChroma sont déduites comme suit.

```
CodedBlockPatternLuma = coded_block_pattern % 16
CodedBlockPatternChroma = coded_block_pattern / 16                    (7-33)
```

Lorsque coded_block_pattern est présent, CodedBlockPatternLuma spécifie, pour chacun des quatre blocs luma 8x8 du macrobloc, un des cas suivants:

– Tous les niveaux du coefficient de transformée des quatre blocs luma 4x4 dans le bloc luma 8x8 sont égaux à zéro;

– Un ou plusieurs niveaux de coefficient de transformée d'un ou de plusieurs des blocs luma 4x4 dans le bloc luma 8x8 doivent avoir une valeur différente de zéro.

La signification de CodedBlockPatternChroma est spécifiée au Tableau 7-15.

Tableau 7-15 – Spécification des valeurs de CodedBlockPatternChroma

CodedBlockPatternChroma	Description
0	Tous les niveaux de coefficient de transformée chroma sont égaux à 0
1	Un ou plusieurs niveaux de coefficient de transformée chroma DC doivent avoir une valeur différente de zéro. Tous les niveaux de coefficient de transformée chroma AC sont égaux à 0.
2	Zéro ou plusieurs niveaux de coefficient de transformée chroma DC ont une valeur différente de zéro. Un ou plusieurs niveaux de coefficient de transformée chroma AC doivent avoir une valeur différente de zéro.

mb_qp_delta peut changer la valeur de QP_Y dans la couche de macrobloc. La valeur décodée de mb_qp_delta doit être dans la gamme de $-(26 + QpBdOffset_Y / 2)$ à $+(25 + QpBdOffset_Y / 2)$ inclus. mb_qp_delta doit être supposé égal à 0 lorsqu'il n'est pas présent pour tout macrobloc (y compris les types de macroblocs P_Skip et B_Skip).

La valeur de QP_Y est calculée comme suit:

$$QP_Y = ((QP_{Y,PREV} + mb\_qp\_delta + 52 + 2 * QpBdOffset_Y) \% (52 + QpBdOffset_Y)) - QpBdOffset_Y \quad (7\text{-}34)$$

où $QP_{Y,PREV}$ est le paramètre de quantification luma, QP_Y, du précédant macrobloc dans l'ordre de décodage dans la tranche en cours. Pour le premier macrobloc dans la tranche, $QP_{Y,PREV}$ est mis initialement égal à $SliceQP_Y$ calculé dans l'équation 7-27 au début de chaque tranche.

La valeur de QP'_Y est calculée comme suit:

$$QP'_Y = QP_Y + QpBdOffset_Y \quad (7\text{-}35)$$

7.4.5.1 Sémantique de prédiction de macrobloc

Tous les échantillons du macrobloc sont prédits. Les modes de prédiction sont déduits au moyen des éléments syntaxiques suivants.

prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]** et **rem_intra4x4_pred_mode[** luma4x4BlkIdx] spécifient la prédiction Intra_4x4 des blocs luma 4x4 avec l'indice luma4x4BlkIdx = 0..15.

prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] et **rem_intra8x8_pred_mode**[luma8x8BlkIdx] spécifient la prédiction Intra_8x8 du bloc luma 8x8 ayant l'indice luma8x8BlkIdx = 0..3.

intra_chroma_pred_mode spécifie le type de prédiction spatiale utilisé pour chroma dans les macroblocs utilisant la prédiction Intra_4x4 ou Intra_16x16, comme indiqué au Tableau 7-16. La valeur de intra_chroma_pred_mode doit être dans l'étendue de 0 à 3, inclus.

Tableau 7-16 – Relations entre intra_chroma_pred_mode et modes de prédiction spatiale

intra_chroma_pred_mode	Mode d'intraprédiction chroma
0	DC
1	Horizontal
2	Vertical
3	Plan

ref_idx_l0[mbPartIdx **],** lorsqu'il est présent, spécifie l'indice dans la liste 0 d'images de référence, de l'image de référence à utiliser pour la prédiction.

La gamme de ref_idx_l0[mbPartIdx], l'indice dans la liste 0 d'images de référence et, si applicable, la parité de la sous-trame au sein de l'image de référence utilisée pour la prédiction sont spécifiés comme suit.

– Si MbaffFrameFlag est égal à 0 ou si mb_field_decoding_flag est égal à 0, la valeur de ref_idx_l0[mbPartIdx] doit être dans la gamme de 0 à num_ref_idx_l0_active_minus1 inclus.

– Sinon (si MbaffFrameFlag est égal à 1 et si mb_field_decoding_flag est égal à 1), la valeur de ref_idx_l0[mbPartIdx] doit être dans la gamme de 0 à 2 * num_ref_idx_l0_active_minus1 + 1, inclus.

Lorsque seule une image de référence est utilisée pour l'interprédiction, les valeurs de ref_idx_l0[mbPartIdx] doivent être supposées égales à 0.

ref_idx_l1[mbPartIdx **]** a la même sémantique que ref_idx_l0, avec l0 et liste 0 remplacés respectivement par l1 et liste 1.

mvd_l0[mbPartIdx **][** 0 **][** compIdx **]** spécifie la différence entre une composante vectorielle à utiliser et sa prédiction. L'indice mbPartIdx spécifie à quelle subdivision de macrobloc mvd_l0 est alloué. La subdivision du macrobloc est spécifiée par mb_type. La différence de composante horizontale de vecteur cinétique est décodée la première dans l'ordre de décodage et est allouée à CompIdx = 0. La composante verticale de vecteur cinétique est décodée en deuxième dans l'ordre de décodage et est allouée à CompIdx = 1. La gamme des composantes de mvd_l0[mbPartIdx][0][compIdx] est spécifiée par des contraintes sur les valeurs des variables de vecteur cinétique qui en sont déduites comme spécifié à l'Annexe A.

mvd_l1[mbPartIdx **][** 0 **][** compIdx **]** a la même sémantique que mvd_l0, avec l0 et L0 remplacés par l1 et L1, respectivement.

7.4.5.2 Sémantique de prédiction de sous-macrobloc

sub_mb_type[mbPartIdx **]** spécifie les types de sous-macrobloc.

Les tableaux et la sémantique sont spécifiés pour les divers types de sous-macrobloc contenus dans les macroblocs de type P et B. Chaque tableau présente la valeur de sub_mb_type, le nom de sub_mb_type, le nombre de subdivisions de sous-macrobloc utilisées (donné par la fonction NumSubMbPart(sub_mb_type)) et le mode de prédiction du sous-macrobloc (donné par la fonction SubMbPredMode(sub_mb_type)). Dans le texte, la valeur de sub_mb_type peut être référencée "type de sous-macrobloc". Dans le texte, la valeur de SubMbPredMode() peut être référencée "mode de prédiction de sous-macrobloc".

L'interprétation de sub_mb_type[mbPartIdx] pour les macroblocs de type P est spécifiée dans le Tableau 7-17, dans lequel la rangée des valeurs "présumées" spécifie des valeurs déduites en cas d'absence de l'élément sub_mb_type[mbPartIdx].

Tableau 7-17 – Types de sous-macrobloc dans les macroblocs P

sub_mb_type[mbPartIdx]	Nom de sub_mb_type[mbPartIdx]	NumSubMbPart (sub_mb_type[mbPartIdx])	SubMbPredMode (sub_mb_type[mbPartIdx])	SubMbPartWidth (sub_mb_type[mbPartIdx])	SubMbPartHeight (sub_mb_type[mbPartIdx])
Présumé	na	na	na	na	na
0	P_L0_8x8	1	Pred_L0	8	8
1	P_L0_8x4	2	Pred_L0	8	4
2	P_L0_4x8	2	Pred_L0	4	8
3	P_L0_4x4	4	Pred_L0	4	4

La sémantique suivante est allouée aux types de sous-macrobloc dans le Tableau 7-17.

– P_L0_MxN, avec MxN remplacé par 8x8, 8x4, 4x8 ou 4x4: les échantillons du sous-macrobloc sont prédits au moyen d'une subdivision luma d'une taille MxN égale à 8x8, deux subdivisions luma d'une taille MxN égale à 8x4 ou deux subdivisions luma d'une taille MxN égale à 4x8 ou quatre subdivisions luma d'une taille MxN égale à 4x4 et les échantillons chroma associés, respectivement.

La sémantique suivante est allouée aux modes de prédiction de sous-macrobloc (SubMbPredMode()) au Tableau 7-17.

– Pred_L0: voir la sémantique au Tableau 7-13.

L'interprétation de sub_mb_type[mbPartIdx] pour les macroblocs de type B est spécifiée dans le Tableau 7-18, où la rangée "présumé" spécifie les valeurs déduites lorsque sub_mb_type[mbPartIdx] n'est pas présent, et la valeur présumée "mb_type" spécifie que le nom de sub_mb_type[mbPartIdx] est le même que le nom de mb_type dans ce cas.

Tableau 7-18 – Types de sous-macrobloc dans les macroblocs B

sub_mb_type[mbPartIdx]	Nom de sub_mb_type[mbPartIdx]	NumSubMbPart (sub_mb_type[mbPartIdx])	SubMbPredMode (sub_mb_type[mbPartIdx])	SubMbPartWidth (sub_mb_type[mbPartIdx])	SubMbPartHeight (sub_mb_type[mbPartIdx])
Présumé	mb_type	4	Direct	4	4
0	B_Direct_8x8	4	Direct	4	4
1	B_L0_8x8	1	Pred_L0	8	8
2	B_L1_8x8	1	Pred_L1	8	8
3	B_Bi_8x8	1	BiPred	8	8
4	B_L0_8x4	2	Pred_L0	8	4
5	B_L0_4x8	2	Pred_L0	4	8
6	B_L1_8x4	2	Pred_L1	8	4
7	B_L1_4x8	2	Pred_L1	4	8
8	B_Bi_8x4	2	BiPred	8	4
9	B_Bi_4x8	2	BiPred	4	8
10	B_L0_4x4	4	Pred_L0	4	4
11	B_L1_4x4	4	Pred_L1	4	4
12	B_Bi_4x4	4	BiPred	4	4

La sémantique suivante est allouée aux types de sous-macrobloc dans le Tableau 7-18:

– B_Skip et B_Direct_16x16: aucune différence de vecteur cinétique ou d'indice de référence n'est présente pour le sous-macrobloc dans le flux binaire. Les fonctions SubMbPartWidth( ) et SubMbPartHeight( ) sont utilisées dans le processus de déduction pour les vecteurs cinétiques et les indices de trame de référence au § 8.4.1 pour la prédiction en mode direct.

– B_Direct_8x8: aucune différence de vecteur cinétique ou d'indice de référence n'est présente pour le sous-macrobloc dans le flux binaire. Les fonctions SubMbPartWidth(B_Direct_8x8) et SubMbPartHeight(B_Direct_8x8) sont utilisées dans le processus de déduction pour les vecteurs cinétiques et les indices de trame de référence au § 8.4.1 pour la prédiction en mode direct.

– B_X_MxN, avec X remplacé par L0, L1 ou Bi et MxN remplacé par 8x8, 8x4, 4x8 ou 4x4: les échantillons du sous-macrobloc sont prédits au moyen d'une subdivision luma de taille MxN égale à 8x8 ou les échantillons du sous-macrobloc sont prédits au moyen de deux subdivisions luma d'une taille MxN égale à 8x4 ou les échantillons du sous-macrobloc sont prédits au moyen de deux subdivisions luma d'une taille MxN égale à 4x8 ou les échantillons du sous-macrobloc sont prédits au moyen de quatre subdivisions luma de taille MxN égale à 4x4 et les échantillons chroma associés, respectivement. Toutes les subdivisions de sous-macrobloc partagent le même indice de référence. Pour une subdivision MxN de sous-macrobloc dans un sous-macrobloc dont le sub_mb_type est B_X_MxN avec X remplacé par L0 ou L1, une différence de vecteur cinétique est présente dans le flux binaire. Pour une subdivision MxN de sous-macrobloc dans un sous-macrobloc dont le sub_mb_type est B_Bi_MxN, deux différences de vecteur cinétique sont présentes dans le flux binaire.

La sémantique suivante est allouée aux modes de prédiction de sous-macrobloc (SubMbPredMode( )) au Tableau 7-18.

– Direct: voir la sémantique pour le Tableau 7-14.

– Pred_L0: voir la sémantique pour le Tableau 7-13.

– Pred_L1: voir la sémantique pour le Tableau 7-14.

– BiPred: voir la sémantique pour le Tableau 7-14.

ref_idx_l0[mbPartIdx **]** a la même sémantique que ref_idx_l0 au § 7.4.5.1.

ref_idx_l1[mbPartIdx **]** a la même sémantique que ref_idx_l1 au § 7.4.5.1.

mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]** a la même sémantique que mvd_l0 au § 7.4.5.1, excepté qu'il s'applique à l'indice de subdivision de sous-macrobloc avec subMbPartIdx. Les indices mbPartIdx et subMbPartIdx spécifient à quelle subdivision de macrobloc et subdivision de sous-macrobloc mvd_l0 est alloué.

mvd_l1[mbPartIdx **][** subMbPartIdx **][** compIdx **]** a la même sémantique que mvd_l1 au § 7.4.5.1.

7.4.5.3 Sémantique de données résiduelles

La structure syntaxique residual_block( ), qui est utilisée pour l'analyse syntaxique des niveaux des coefficients de transformée, est allouée comme suit.

– Si entropy_coding_mode_flag est égal à 0, residual_block est mis égal à residual_block_cavlc, qui est utilisé pour l'analyse syntaxique des éléments syntaxiques pour les niveaux de coefficients de transformée.

– Sinon (si entropy_coding_mode_flag est égal à 1), residual_block est mis égal à residual_block_cabac, qui est utilisé pour l'analyse syntaxique des éléments syntaxiques pour les niveaux de coefficients de transformée.

En fonction de mb_type, luma ou chroma, et selon le format chromatique, la structure syntaxique residual_block(coeffLevel, maxNumCoeff) est utilisée avec les arguments coeffLevel, qui est une liste contenant les niveaux de coefficient de transformée maxNumCoeff dont l'analyse syntaxique est effectuée dans residual_block( ) et maxNumCoeff comme suit.

– En fonction de MbPartPredMode(mb_type, 0), on applique ce qui suit.

  – Si MbPartPredMode(mb_type, 0) est égal à Intra_16x16, les niveaux de coefficient de transformée sont analysés dans la liste Intra16x16DCLevel et dans les 16 listes Intra16x16ACLevel[i]. Intra16x16DCLevel contient les 16 niveaux de coefficient de transformée des niveaux de coefficient de transformée DC pour chaque bloc luma 4x4. Pour chacun des 16 blocs luma 4x4 indexés par i = 0..15, les 15 niveaux de coefficients de transformée AC du $i^{ème}$ bloc sont analysés dans la $i^{ème}$ liste Intra16x16ACLevel[i].

  – Sinon (si MbPartPredMode(mb_type, 0) n'est pas égal à Intra_16x16), ce qui suit s'applique.

    – Si le fanion transform_size_8x8 est égal à 0, pour chacun des 16 blocs luma 4x4 indexés par i = 0..15, les 16 niveaux de coefficient de transformée AC du $i^{ème}$ bloc sont analysés dans la $i^{ème}$ liste LumaLevel[i].

    – Sinon (si transform_size_8x8_flag est égal à 1), pour chacun des 4 blocs luma 8x8 indexés par i8x8 = 0..3, ce qui suit est applicable:

      – Si entropy_coding_mode_flag est égal à 0, d'abord pour chacun des 4 blocs luma 4x4 indexés par i4x4 = 0..3, les 16 niveaux des coefficients de transformée du i4x4-ième bloc sont injectés par analyse syntaxique/sémantique dans la (i8x8 * 4 + i4x4)-ième liste LumaLevel[i8x8 * 4 + i4x4]. Ensuite, les 64 niveaux des coefficients de transformée du i8x8-ième bloc luma 8x8 qui sont indexés par 4 * i + i4x4, où i = 0..15 et i4x4 = 0..3, sont calculés par: LumaLevel8x8[i8x8][4 * i + i4x4] = LumaLevel[i8x8 * 4 + i4x4][i].

        NOTE – Les blocs luma 4x4, avec luma4x4BlkIdx = i8x8 * 4 + i4x4 contenant un niveau sur quatre des coefficients de transformée du i8x8-ième bloc luma 8x8 correspondant au décalage i4x4, sont pris comme hypothèse afin de représenter les localisations spatiales données par le processus de balayage inverse de bloc luma 4x4 décrit dans le § 6.4.3.

      – Sinon (si entropy_coding_mode_flag est égal à 1), les 64 niveaux des coefficients de transformée du i8x8-ième bloc sont injectés par analyse syntaxique/sémantique dans la i8x8-ième liste LumaLevel8x8[i8x8].

– Pour chaque composante chromatique, indexée par iCbCr = 0..1, les niveaux apériodiques des coefficients de transformée des 4 * NumC8x8 blocs chroma 4x4 sont injectés par analyse syntaxique/sémantique dans la iCbCr-ième liste ChromaDCLevel[iCbCr].

– Pour chacun des blocs chroma 4x4, indexés par i4x4 = 0..3 et i8x8 = 0..NumC8x8 – 1, de chaque composante chromatique indexée par iCbCr = 0..1, les 15 niveaux périodiques des coefficients de transformée sont injectés par analyse syntaxique/sémantique dans la (i8x8*4 + i4x4)-ième liste de la iCbCr-ième composante chromatique ChromaACLevel[iCbCr][i8x8*4 + i4x4].

7.4.5.3.1 Sémantique du codage CAVLC de bloc résiduel

La fonction TotalCoeff(coeff_token) qui est utilisée au § 7.3.5.3.1 renvoie le nombre de niveaux de coefficient de transformée différents de zéro déduits de coeff_token.

La fonction TrailingOnes(coeff_token) qui est utilisée au § 7.3.5.3.1 renvoie ceux de traîne qui sont déduits de coeff_token.

coeff_token spécifie le nombre total de niveaux de coefficient de transformée différents de zéro et le nombre de niveaux de coefficient de transformée de traîne dans un balayage de niveau de coefficient de transformée. Un niveau de coefficient de transformée de traîne est un des jusqu'à trois niveaux de coefficient de transformée consécutifs différents de zéro ayant une valeur absolue égale à 1 à la fin d'un balayage des niveaux de coefficient de transformée différents de zéro. La gamme de coeff_token est spécifiée au § 9.2.1.

trailing_ones_sign_flag spécifie le signe du niveau de coefficient de transformée de traîne comme suit.

– Si trailing_ones_sign_flag est égal à 0, le niveau de coefficient de transformée correspondant est décodé comme +1.

– Sinon (si trailing_ones_sign_flag égal à 1), le niveau de coefficient de transformée correspondant est décodé comme –1.

level_prefix et **level_suffix** spécifient la valeur d'un niveau de coefficient de transformée différent de zéro. La gamme de level_prefix et level_suffix est spécifiée au § 9.2.2.

total_zeros spécifie le nombre total de niveaux de coefficient de transformée de valeur zéro qui sont localisés avant la position du dernier niveau de coefficient de transformée différent de zéro dans le balayage de niveaux de coefficient de transformée. La gamme de total_zeros est spécifiée au § 9.2.3.

run_before spécifie le nombre de niveaux de coefficient de transformée consécutifs dans le balayage avec la valeur zéro avant un niveau de coefficient de transformée de valeur différente de zéro. La gamme de run_before est spécifiée au § 9.2.3.

coeffLevel contient maxNumCoeff niveaux de coefficient de transformée pour la liste en cours de niveaux de coefficient de transformée.

7.4.5.3.2 Sémantique du codage CABAC de bloc résiduel

coded_block_flag spécifie si le bloc contient des niveaux de coefficient de transformée différents de zéro comme suit.

– Si coded_block_flag est égal à 0, le bloc ne contient aucun niveau de coefficient de transformée différent de zéro.

– Sinon (si coded_block_flag est égal à 1), le bloc contient au moins un niveau de coefficient de transformée différent de zéro.

significant_coeff_flag[i **]** spécifie si le niveau de coefficient de transformée à la position de balayage i est différent de zéro comme suit.

– Si significant_coeff_flag[i] est égal à 0, le niveau de coefficient de transformée à la position de balayage i est mis égal à 0.

– Sinon (si significant_coeff_flag[i] est égal à 1), le niveau de coefficient de transformée à la position de balayage i a une valeur différente de zéro.

last_significant_coeff_flag[i **]** spécifie comme suit, pour la position de balayage i, s'il y a des niveaux de coefficient de transformée différents de zéro pour les i + 1 jusqu'à maxNumCoeff – 1 positions de balayage suivantes.

– Si last_significant_coeff_flag[i] est égal à 1, tous les niveaux de coefficient de transformée suivants (dans l'ordre de balayage) du bloc ont leur valeur égale à 0.

– Sinon (si last_significant_coeff_flag[i] est égal à 0), il y a d'autres niveaux de coefficient de transformée différents de zéro sur le chemin de balayage.

coeff_abs_level_minus1[i **]** est la valeur absolue d'un niveau de coefficient de transformée moins 1. La valeur de coeff_abs_level_minus1 est contrainte par les limites fixées au § 8.5.

coeff_sign_flag[i **]** spécifie le signe d'un niveau de coefficient de transformée comme suit.

– Si coeff_sign_flag est égal à 0, le niveau de coefficient de transformée correspondant a une valeur positive.

– Sinon (si coeff_sign_flag est égal à 1), le niveau de coefficient de transformée correspondant a une valeur négative.

coeffLevel contient maxNumCoeff niveaux de coefficient de transformée pour la liste en cours des niveaux de coefficient de transformée.

8 Processus de décodage

Les résultats de ce processus sont des échantillons décodés de l'image en cours (parfois désignée par la variable CurrPic).

Le présent paragraphe décrit le processus de décodage, étant donnés les éléments syntaxiques et les variables en majuscules du § 7.

Le processus de décodage est spécifié de telle sorte que tous les décodeurs doivent produire des résultats numériquement identiques. Tout processus de décodage qui produit des résultats identiques à la fin du processus décrit ici se conforme aux exigences du processus de décodage de la présente Recommandation | Norme internationale.

Chaque image mentionnée dans le présent paragraphe est une image primaire. Chaque tranche mentionnée dans le présent paragraphe est une tranche d'une image primaire. Chaque subdivision de données de tranche mentionnée dans le présent paragraphe est une subdivision de données de tranche d'une image primaire.

Un aperçu général du processus de décodage peut être donnée comme suit.

- Le décodage des unités NAL est spécifié au § 8.1.
- Les processus du § 8.2 spécifient les processus de décodage utilisant des éléments syntaxiques dans la couche de tranche et au-dessus.
  - Les variables et fonctions se rapportant au compte d'ordre d'image sont décrites au § 8.2.1 (elles ne doivent être invoquées que pour une seule tranche d'une image).
  - Les variables et fonctions se rapportant au mappage de macrobloc à groupe de tranches sont décrites au § 8.2.2 (elles ne doivent être invoquées que pour une seule tranche d'une image).
  - La méthode de combinaison des diverses subdivisions, lorsque la subdivision de données de tranche est utilisée, est décrite au § 8.2.3.
  - Lorsque l'élément frame_num de l'image actuelle n'est pas égal à PrevRefFrameNum et n'est pas égal à (PrevRefFrameNum + 1) % MaxFrameNum, le processus de décodage pour lacunes dans frame_num est effectué conformément au § 8.2.5.2 avant le décodage de toutes les tranches de l'image actuelle.
  - Au début du processus de décodage pour chaque tranche de type P, SP ou B, le processus de décodage pour construction de listes d'images de référence, spécifié dans le § 8.2.4, est effectué en vue de la déduction de la liste 0 d'images de référence (RefPicList0) et, lors du décodage d'une tranche de type B, en vue de la déduction de la liste 1 d'images de référence (RefPicList1).
  - Lorsque l'image en cours est une image de référence et après que toutes les tranches de l'image en cours ont été décodées, le processus de marquage d'image de référence décodée du § 8.2.5 spécifie comment l'image en cours est utilisée dans le processus de décodage de l'interprédiction dans les images décodées ultérieurement.
- Les processus des § 8.3, 8.4, 8.5, 8.6 et 8.7 spécifient les processus de décodage utilisant des éléments syntaxiques dans la couche de macrobloc et au-dessus.
  - Le processus d'intraprédiction pour les macroblocs I et SI, excepté pour les macroblocs I_PCM comme spécifié au § 8.3, a pour résultat les échantillons d'intraprédiction. Pour les macroblocs I_PCM, le § 8.3 spécifie directement un processus de construction d'image. Les résultats sont les échantillons construits avant le processus de filtrage de démontage de blocs.
  - Le processus d'interprédiction pour les macroblocs P et B est spécifié au § 8.4 et les échantillons d'interprédiction en sont le résultat.
  - Le processus de décodage de coefficient de transformée et de construction d'image avant le processus de filtrage de démontage de blocs est spécifié au § 8.5. Ce processus déduit les échantillons pour les macroblocs I et B ainsi que pour les macroblocs P dans les tranches P. Les résultats sont les échantillons construits avant le processus de filtrage de démontage de blocs.
  - Le processus de décodage pour les macroblocs P dans les tranches SP ou les macroblocs SI est spécifié au § 8.6. Ce processus déduit les échantillons pour les macroblocs P dans les tranches SP et pour les macroblocs SI. Les résultats sont les échantillons construits avant le processus de filtrage de démontage de blocs.
  - Les échantillons construits avant le processus de filtrage de démontage de blocs qui sont contre les bordures des blocs et macroblocs sont traités par un filtre de démontage de blocs comme spécifié au § 8.7, dont les résultats sont les échantillons décodés.

8.1 Processus de décodage d'unité NAL

Les entrées dans ce processus sont des unités NAL.

Les résultats de ce processus sont des structures syntaxiques de charge utile RBSP encapsulées au sein des unités NAL.

Le processus de décodage extrait pour chaque unité NAL la structure syntaxique de charge utile RBSP de l'unité NAL et effectue alors les processus de décodage spécifiés pour la structure syntaxique de charge utile RBSP dans l'unité NAL comme suit.

Le paragraphe 8.2 décrit le processus de décodage pour des unités NAL avec nal_unit_type égal à 1 jusqu'à 5.

Le paragraphe 8.3 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec nal_unit_type égal à 1, 2 et 5.

Le paragraphe 8.4 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec nal_unit_type égal à 1 et 2.

Le paragraphe 8.5 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec nal_unit_type égal à 1 et 3 jusqu'à 5.

Le paragraphe 8.6 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec nal_unit_type égal à 1 et 3 jusqu'à 5.

Le paragraphe 8.7 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec nal_unit_type égal à 1 jusqu'à 5.

Les unités NAL avec nal_unit_type égal à 7 et 8 contiennent respectivement des ensembles de paramètres de séquence et des ensembles de paramètres d'image. Les ensembles de paramètres d'image sont utilisés dans les processus de décodage des autres unités NAL comme déterminé par référence à un ensemble de paramètres d'image au sein des en-têtes de tranche de chaque image. Les ensembles de paramètres de séquence sont utilisés dans les processus de décodage des autres unités NAL comme déterminé par référence à un ensemble de paramètres de séquence au sein des ensembles de paramètres d'image de chaque séquence.

Il n'y a pas de processus de décodage normalisé spécifié pour des unités NAL avec nal_unit_type égal à 6, 9, 10, 11 et 12.

8.2 Processus de décodage de tranche

8.2.1 Processus de décodage pour le compte d'ordre d'image

Les résultats de ce processus sont TopFieldOrderCnt (si applicable) et BottomFieldOrderCnt (si applicable).

Les comptes d'ordre d'image sont utilisés afin de déterminer les arrangements d'image initiaux pour les images de référence dans le décodage de tranches B (voir les § 8.2.4.2.3 et 8.2.4.2.4), afin de représenter les différences d'ordre d'image entre les trames ou sous-trames pour le calcul du vecteur cinétique dans le mode temporel direct (voir le § 8.4.1.2.3), pour la prédiction pondérée en mode implicite dans les tranches B (voir le § 8.4.2.3.2) et pour la vérification de conformité du décodeur (voir le § C.4).

Les informations de compte d'ordre d'image sont déduites comme suit pour chaque trame, sous-trame (qu'elle soit décodée à partir d'une sous-trame codée ou qu'elle fasse partie d'une trame décodée) ou paire de sous-trames complémentaires:

– chaque trame codée est associée à deux comptes d'ordre d'image, appelés respectivement TopFieldOrderCnt et BottomFieldOrderCnt pour sa sous-trame supérieure et sa sous-trame inférieure;

– chaque sous-trame codée est associée à un compte d'ordre d'image, appelé TopFieldOrderCnt pour une sous-trame supérieure codée et BottomFieldOrderCnt pour une sous-trame inférieure;

– chaque paire de sous-trames complémentaires est associée à deux comptes d'ordre d'image, qui sont respectivement TopFieldOrderCnt pour sa sous-trame supérieure codée et BottomFieldOrderCnt pour sa sous-trame inférieure codée.

TopFieldOrderCnt et BottomFieldOrderCnt indiquent l'ordre d'image de la sous-trame supérieure ou de la sous-trame inférieure correspondantà la première sous-trame résultant de la précédenteimage à rafraîchissement IDR ou de la précédente image de référence incluant un élément memory_management_control_operation égal à 5 dans l'ordre de décodage.

TopFieldOrderCnt et BottomFieldOrderCnt sont déduits par invocation d'un des processus de décodage pour le type de compte d'ordre d'image 0, 1 et 2, respectivement aux § 8.2.1.1, 8.2.1.2 et 8.2.1.3. Lorsque l'image en cours inclut une

opération de commande de gestion de mémoire égale à 5, après le décodage de l'image en cours, tempPicOrderCnt est mis égal à PicOrderCnt(CurrPic), TopFieldOrderCnt de l'image en cours (le cas échéant) est mis égal à TopFieldOrderCnt – tempPicOrderCnt et BottomFieldOrderCnt de l'image en cours (le cas échéant) est mis égal à BottomFieldOrderCnt – tempPicOrderCnt.

Le flux binaire ne doit pas contenir de données qui auraient pour résultat un élément Min(TopFieldOrderCnt, BottomFieldOrderCnt) qui ne soit pas égal à 0 pour une trame IDR codée, un élément TopFieldOrderCnt qui ne soit pas égal à 0 pour une sous-trame supérieure IDR codée ou un élément BottomFieldOrderCnt qui ne soit pas égal à 0 pour une sous-trame inférieure IDR codée. Ainsi, au moins un des deux éléments TopFieldOrderCnt et BottomFieldOrderCnt doit être égal à 0 pour les sous-trames d'une trame IDR codée.

Lorsque l'image en cours n'est pas à rafraîchissement IDR, on applique ce qui suit.

– Considérer la liste de variables listD contenant comme éléments les valeurs TopFieldOrderCnt et BottomFieldOrderCnt associées à la liste des images et incluant tout ce qui suit:
  – la première image dans la liste est l'image précédente de n'importe lequel des types suivants:
    – uneimage à rafraîchissement IDR;
    – une image contenant un élément memory_management_control_operation égal à 5;
  – les images supplémentaires suivantes:
    – Si pic_order_cnt_type est égal à 0, toutes les autres images qui suivent dans l'ordre de décodage après la première image dans la liste et qui ne sont pas des trames "non existantes" déduites par le processus de décodage pour les lacunes dans frame_num spécifié au § 8.2.5.2 et qui précèdent l'image en cours dans l'ordre de décodage ou constituent l'image en cours. Lorsque pic_order_cnt_type est égal à 0 et que l'image en cours n'est pas une trame "non existante" déduite par le processus de décodage pour les lacunes dans frame_num spécifié au § 8.2.5.2, l'image en cours est incluse dans listD avant l'invocation du processus de marquage d'image de référence décodée.
    – Sinon (si pic_order_cnt_type est différent de 0), toutes les autres images qui suivent dans l'ordre de décodage après la première image dans la liste et qui précèdent l'image en cours dans l'ordre de décodage ou constituent l'image en cours. Lorsque pic_order_cnt_type est différent de 0, l'image en cours est incluse dans listD avant l'invocation du processus de marquage d'image de référence décodée.
– Soit la liste de variables listo qui contient les éléments de listD triés en ordre ascendant. listo ne doit contenir aucun des éléments suivants:
  – une paire de TopFieldOrderCnt et BottomFieldOrderCnt pour une trame ou paire de sous-trames complémentaires qui ne soient pas à des positions consécutives dans listO;
  – un TopFieldOrderCnt qui ait une valeur égale à un autre TopFieldOrderCnt;
  – un BottomFieldOrderCnt qui ait une valeur égale à un autre BottomFieldOrderCnt;
  – un BottomFieldOrderCnt qui ait une valeur égale à un TopFieldOrderCnt à moins que le BottomFieldOrderCnt et le TopFieldOrderCnt n'appartiennent à la même trame codée ou paire de sous-trames complémentaires.

Le flux binaire ne doit pas contenir de données qui résulteraient en des valeurs de TopFieldOrderCnt, BottomFieldOrderCnt, PicOrderCntMsb ou FrameNumOffset utilisées dans le processus de décodage, comme spécifié aux § 8.2.1.1 à 8.2.1.3, qui excèdent la gamme de valeurs de -2^{31} à $2^{31} - 1$ inclus.

La fonction PicOrderCnt(picX) est spécifiée comme suit:

```
si( picX est une trame ou une paire de sous-trames complémentaires )
    PicOrderCnt( picX ) = Min( TopFieldOrderCnt, BottomFieldOrderCnt ) de la trame ou paire
de sous-trames complémentaires picX
ou si( picX est une sous-trame supérieure )
    PicOrderCnt( picX ) = TopFieldOrderCnt de sous-trame picX                    (8-1)
ou si( picX est une sous-trame inférieure )
    PicOrderCnt( picX ) = BottomFieldOrderCnt de sous-trame picX
```

Alors DiffPicOrderCnt(picA, picB) est spécifié comme suit:

```
DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) – PicOrderCnt( picB )    (8-2)
```

Le flux binaire ne doit pas contenir de données ayant pour résultat des valeurs de DiffPicOrderCnt(picA, picB), utilisées dans le processus de décodage, qui soient hors de la gamme de -2^{15} à $2^{15} - 1$ inclus.

NOTE 1 – Soit X l'image en cours et Y et Z deux autres images dans la même séquence. Y et Z sont considérées comme étant dans le même sens d'ordre de sortie que X lorsque DiffPicOrderCnt(X, Y) et DiffPicOrderCnt(X, Z) sont tous deux positifs ou tous deux négatifs.

NOTE 2 – De nombreuses applications allouent PicOrderCnt(X) proportionnel au temps d'échantillonnage de l'image X par rapport au temps d'échantillonnage d'uneimage à rafraîchissement IDR.

Lorsque l'image en cours inclut un élément memory_management_control_operation égal à 5, PicOrderCnt(CurrPic) doit être supérieur à PicOrderCnt (toute autre image de listD).

8.2.1.1 Processus de décodage pour le type 0 de compte d'ordre d'image

Ce processus est invoqué lorsque pic_order_cnt_type est égal à 0.

L'entrée dans ce processus est le PicOrderCntMsb de la précédente image de référence dans l'ordre de décodage comme spécifié au présent paragraphe.

Les résultats de ce processus sont TopFieldOrderCnt ou BottomFieldOrderCnt ou les deux.

Les variables prevPicOrderCntMsb et prevPicOrderCntLsb sont déduites comme suit.

– Si l'image en cours est à rafraîchissement IDR, les variables prevPicOrderCntMsb et prevPicOrderCntLsb sont mises égales à 0.

– Sinon (si l'image en cours n'est pas à rafraîchissement IDR), on applique ce qui suit.

  – Si la précédente image de référence dans l'ordre de décodage inclut un élément memory_management_control_operation égal à 5, on applique ce qui suit.

    – Si la précédente image de référence dans l'ordre de décodage n'est pas une sous-trame inférieure, prevPicOrderCntMsb est mis égal à 0 et prevPicOrderCntLsb est mis égal à la valeur de TopFieldOrderCnt pour la précédente image de référence dans l'ordre de décodage.

    – Sinon (si la précédente image de référence dans l'ordre de décodage est une sous-trame inférieure de référence), les variables prevPicOrderCntMsb et prevPicOrderCntLsb sont mises égales à 0.

  – Sinon (si la précédente image de référence dans l'ordre de décodage n'incluait pas d'élément memory_management_control_operation égal à 5), la variable prevPicOrderCntMsb est mise égale à PicOrderCntMsb de l'image précédente de référence dans l'ordre de décodage et prevPicOrderCntLsb est mise égale à la valeur de pic_order_cnt_lsb de l'image de référence précédente dans l'ordre de décodage.

PicOrderCntMsb de l'image en cours est déduit comme suit:

```
si( ( pic_order_cnt_lsb  <  prevPicOrderCntLsb )  &&
    ( ( prevPicOrderCntLsb – pic_order_cnt_lsb )  >=  ( MaxPicOrderCntLsb / 2 ) ) )
    PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb                    (8-3)
ou si( ( pic_order_cnt_lsb  >  prevPicOrderCntLsb )  &&
        ( ( pic_order_cnt_lsb – prevPicOrderCntLsb )  >  ( MaxPicOrderCntLsb / 2 ) ) )
    PicOrderCntMsb = prevPicOrderCntMsb – MaxPicOrderCntLsb
sinon
    PicOrderCntMsb = prevPicOrderCntMsb
```

Lorsque l'image en cours n'est pas une sous-trame inférieure, TopFieldOrderCnt est déduit comme suit:

```
si( !field_pic_flag  ||  !bottom_field_flag )
    TopFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb                      (8-4)
```

Lorsque l'image en cours n'est pas une sous-trame supérieure, BottomFieldOrderCnt est déduit comme suit:

```
si( !field_pic_flag )
    BottomFieldOrderCnt = TopFieldOrderCnt + delta_pic_order_cnt_bottom
ou si( bottom_field_flag )                                                     (8-5)
    BottomFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb
```

8.2.1.2 Processus de décodage pour le type 1 de compte d'ordre d'image

Ce processus est invoqué lorsque pic_order_cnt_type est égal à 1.

L'entrée dans ce processus est FrameNumOffset de l'image précédente dans l'ordre de décodage comme spécifié au présent paragraphe.

Les résultats de ce processus sont TopFieldOrderCnt ou BottomFieldOrderCnt ou les deux.

Les valeurs de TopFieldOrderCnt et BottomFieldOrderCnt sont déduites comme spécifié au présent paragraphe. Soit prevFrameNum égal au frame_num de l'image précédente dans l'ordre de décodage.

Lorsque l'image en cours n'est pas à rafraîchissement IDR, la variable prevFrameNumOffset est déduite comme suit.

– Si l'image précédente dans l'ordre de décodage inclut un élément memory_management_control_operation égal à 5, prevFrameNumOffset est mis égal à 0.

– Sinon (si l'image précédente dans l'ordre de décodage n'incluait pas d'élément memory_management_control_operation égal à 5), prevFrameNumOffset est mis égal à la valeur de FrameNumOffset de l'image précédente dans l'ordre de décodage.

NOTE – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, l'image précédente dans l'ordre de décodage peut être une trame "non existante" déduite par le processus de décodage pour les lacunes dans frame_num, spécifié au § 8.2.5.2.

La déduction se fait selon les étapes suivantes.

1. La variable FrameNumOffset est déduite comme suit:

```
si( nal_unit_type  = =  5 )
    FrameNumOffset = 0
ou si( prevFrameNum > frame_num )                                   (8-6)
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum
ou
    FrameNumOffset = prevFrameNumOffset
```

2. La variable absFrameNum est déduite comme suit:

```
si( num_ref_frames_in_pic_order_cnt_cycle  !=  0 )
    absFrameNum = FrameNumOffset + frame_num
ou                                                                  (8-7)
    absFrameNum = 0
si( nal_ref_idc  = =  0  &&  absFrameNum  >  0 )
    absFrameNum = absFrameNum – 1
```

3. Lorsque absFrameNum > 0, picOrderCntCycleCnt et frameNumInPicOrderCntCycle sont déduits comme suit:

```
si( absFrameNum  >  0 ) {
    picOrderCntCycleCnt = ( absFrameNum – 1 ) / num_ref_frames_in_pic_order_cnt_cycle
    frameNumInPicOrderCntCycle = ( absFrameNum – 1 ) % num_ref_frames_in_pic_order_cnt_cycle   (8-8)
}
```

4. La variable expectedDeltaPerPicOrderCntCycle est déduite comme suit:

```
expectedDeltaPerPicOrderCntCycle = 0
pour( i = 0;  i < num_ref_frames_in_pic_order_cnt_cycle;  i++ )
    expectedDeltaPerPicOrderCntCycle += offset_for_ref_frame[ i ]   (8-9)
```

5. La variable expectedPicOrderCnt est déduite comme suit:

```
si( absFrameNum > 0 ){
    expectedPicOrderCnt = picOrderCntCycleCnt * expectedDeltaPerPicOrderCntCycle
    pour( i = 0; i <= frameNumInPicOrderCntCycle; i++ )
        expectedPicOrderCnt = expectedPicOrderCnt + offset_for_ref_frame[ i ]
} ou
    expectedPicOrderCnt = 0
```

```
si( nal_ref_idc  ==  0 )                                                    (8-10)
    expectedPicOrderCnt = expectedPicOrderCnt + offset_for_non_ref_pic
```

6. Les variables TopFieldOrderCnt ou BottomFieldOrderCnt sont déduites comme suit:

```
si( !field_pic_flag ) {
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
    BottomFieldOrderCnt = TopFieldOrderCnt +
            offset_for_top_to_bottom_field + delta_pic_order_cnt[ 1 ]       (8-11)
} ou si( !bottom_field_flag )
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
ou
    BottomFieldOrderCnt = expectedPicOrderCnt + offset_for_top_to_bottom_field + delta_pic_order_cnt[ 0 ]
```

8.2.1.3 Processus de décodage pour le type 2 de compte d'ordre d'image

Ce processus est invoqué lorsque pic_order_cnt_type est égal à 2.

Les résultats de ce processus sont TopFieldOrderCnt ou BottomFieldOrderCnt ou les deux.

Soit prevFrameNum égal au frame_num de l'image précédente dans l'ordre de décodage.

Lorsque l'image en cours n'est pas à rafraîchissement IDR, la variable prevFrameNumOffset est déduite comme suit.

– Si l'image précédente dans l'ordre de décodage inclut un élément memory_management_control_operation égal à 5, prevFrameNumOffset est mis égal à 0.

– Sinon (si l'image précédente dans l'ordre de décodage n'inclut pas d'élément memory_management_control_operation égal à 5), prevFrameNumOffset est mis égal à la valeur de FrameNumOffset de l'image précédente dans l'ordre de décodage.

NOTE 1 – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, l'image précédente dans l'ordre de décodage peut être une trame "non existante" déduite par le processus de décodage pour les lacunes dans frame_num, spécifié au § 8.2.5.2.

La variable FrameNumOffset est déduite comme suit.

```
si( nal_unit_type  ==  5 )
    FrameNumOffset = 0
ou si( prevFrameNum  >  frame_num )                                         (8-12)
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum
ou
    FrameNumOffset = prevFrameNumOffset
```

La variable tempPicOrderCnt est déduite comme suit:

```
si( nal_unit_type  ==  5 )
    tempPicOrderCnt = 0
ou si( nal_ref_idc  ==  0 )                                                 (8-13)
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num ) – 1
ou
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num )
```

Les variables TopFieldOrderCnt ou BottomFieldOrderCnt sont déduites comme suit:

```
si( !field_pic_flag ) {
    TopFieldOrderCnt = tempPicOrderCnt
    BottomFieldOrderCnt = tempPicOrderCnt                                   (8-14)
} ou si( bottom_field_flag )
    BottomFieldOrderCnt = tempPicOrderCnt
ou
    TopFieldOrderCnt = tempPicOrderCnt
```

NOTE 2 – Le type 2 de compte d'ordre d'image ne peut pas être utilisé dans une séquence vidéo codée qui contient des images non référentielles consécutives qui auraient pour résultat plus d'une de ces images ayant la même valeur de TopFieldOrderCnt ou plus d'une de ces images ayant la même valeur de BottomFieldOrderCnt.

NOTE 3 – Le type 2 de compte d'ordre d'image a pour résultat un ordre de sortie qui est le même que l'ordre de décodage.

8.2.2 Mappage de processus de décodage pour macrobloc avec groupe de tranches

Les entrées dans ce processus sont l'ensemble actif de paramètres d'image et l'en-tête de la tranche à décoder.

Le résultat de ce processus est un mappage de macrobloc à groupe de tranche MbToSliceGroupMap.

Ce processus est invoqué au début de chaque tranche.

NOTE – Le résultat de ce processus est égal pour toutes les tranches d'une image.

Lorsque num_slice_groups_minus1 est égal à 1 et slice_group_map_type est égal à 3, 4 ou 5, les groupes de tranches 0 et 1 ont une taille et une forme déterminées par slice_group_change_direction_flag comme indiqué au Tableau 8-1 et spécifié aux § 8.2.2.4-8.2.2.6.

Tableau 8-1 – Type précis de mappage de groupe de tranches

slice_group_map_type	slice_group_change_direction_flag	Type précis de mappage de groupe de tranches
3	0	Sortie de boîte dans le sens des aiguilles d'une montre
3	1	Sortie de boîte dans le sens contraire des aiguilles d'une montre
4	0	Balayage matriciel
4	1	Balayage matriciel inverse
5	0	Balayage à droite
5	1	Balayage à gauche

Dans un tel cas, les unités de mappage de groupe de tranches MapUnitsInSliceGroup0, dans l'ordre croissant spécifié, sont allouées pour le groupe de tranches 0 et les unités de mappage de groupe de tranches PicSizeInMapUnits – MapUnitsInSliceGroup0 restantes de l'image sont allouées pour le groupe de tranches 1.

Lorsque num_slice_groups_minus1 est égal à 1 et slice_group_map_type est égal à 4 ou 5, la variable sizeOfUpperLeftGroup est définie comme suit:

$$\text{sizeOfUpperLeftGroup} = (\text{slice\_group\_change\_direction\_flag} ? (\text{PicSizeInMapUnits} - \text{MapUnitsInSliceGroup0}) : \text{MapUnitsInSliceGroup0}) \quad (8\text{-}15)$$

La variable mapUnitToSliceGroupMap est déduite comme suit.

– Si num_slice_groups_minus1 est égal à 0, le mappage d'unité de mappage au groupe de tranches est produit pour tous les i de 0 à PicSizeInMapUnits – 1 inclus, comme spécifié par:

$$\text{mapUnitToSliceGroupMap}[i] = 0 \quad (8\text{-}16)$$

– Sinon (si num_slice_groups_minus1 n'est pas égal à 0), mapUnitToSliceGroupMap est déduit comme suit.

  – Si slice_group_map_type est égal à 0, la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.1 s'applique.

  – Sinon, si slice_group_map_type est égal à 1, la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.2 s'applique.

  – Sinon, si slice_group_map_type est égal à 2, la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.3 s'applique.

  – Sinon, si slice_group_map_type est égal à 3, la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.4 s'applique.

  – Sinon, si slice_group_map_type est égal à 4, la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.5 s'applique.

  – Sinon, si slice_group_map_type est égal à 5, la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.6 s'applique.

  – Sinon (si slice_group_map_type est égal à 6), la déduction de mapUnitToSliceGroupMap comme spécifié au § 8.2.2.7 s'applique.

Après déduction de mapUnitToSliceGroupMap, le processus spécifié au § 8.2.2.8 est invoqué afin de convertir l'unité de mappage en mappage de groupe de tranches mapUnitToSliceGroupMap avec le mappage de macrobloc à groupe de tranches MbToSliceGroupMap. Après déduction du mappage de macrobloc à groupe de tranches comme spécifié au § 8.2.2.8, la fonction NextMbAddress(n) est définie comme la valeur de la variable nextMbAddress déduite comme spécifié par:

```
i = n + 1
tant que( i < PicSizeInMbs  &&  MbToSliceGroupMap[ i ]  !=  MbToSliceGroupMap[ n ] )
    i++;
nextMbAddress = i                                                                    (8-17)
```

8.2.2.1 Spécification pour le type de mappage de groupe de tranches entrelacées

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 0.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

```
i = 0
faire
    pour( iGroup = 0; iGroup <= num_slice_groups_minus1 && i < PicSizeInMapUnits;
          i += run_length_minus1[ iGroup++ ] + 1 )
        pour( j = 0; j <= run_length_minus1[ iGroup ] && i + j < PicSizeInMapUnits; j++ )
            mapUnitToSliceGroupMap[ i + j ] = iGroup                                 (8-18)
tant que( i < PicSizeInMapUnits )
```

8.2.2.2 Spécification pour le type de mappage de groupe de tranches dispersées

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 1.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

```
pour( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = ( ( i % PicWidthInMbs ) +
                                    ( ( ( i / PicWidthInMbs ) * ( num_slice_groups_minus1 + 1 ) ) / 2 ) )
                                  % ( num_slice_groups_minus1 + 1 )                  (8-19)
```

8.2.2.3 Spécification pour le type de mappage de groupe de tranches de premier plan avec surplus

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 2.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

```
pour( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = num_slice_groups_minus1
pour( iGroup = num_slice_groups_minus1 – 1; iGroup >= 0; iGroup-- ) {
    yTopLeft = top_left[ iGroup ] / PicWidthInMbs
    xTopLeft = top_left[ iGroup ] % PicWidthInMbs
    yBottomRight = bottom_right[ iGroup ] / PicWidthInMbs
    xBottomRight = bottom_right[ iGroup ] % PicWidthInMbs
    pour( y = yTopLeft; y <= yBottomRight; y++ )
        pour( x = xTopLeft; x <= xBottomRight; x++ )
            mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = iGroup                 (8-20)
}
```

NOTE – Les rectangles peuvent se chevaucher. Le groupe de tranches 0 contient les macroblocs qui sont au sein du rectangle spécifié par top_left[0] et bottom_right[0]. Un groupe de tranches ayant un ID de groupe de tranches supérieur à 0 et inférieur à num_slice_groups_minus1 contient les macroblocs qui sont au sein du rectangle spécifié pour ce groupe de tranches qui ne sont pas au sein du rectangle spécifié pour tout groupe de tranches ayant un ID de groupe de tranches plus petit. Le groupe de tranches avec un ID de groupe de tranches égal à num_slice_groups_minus1 contient les macroblocs qui ne sont pas dans les autres groupes de tranches.

8.2.2.4 Spécification pour les types de mappage de groupe de tranches en sortie de boîte

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 3.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

```
pour( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = 1
x = ( PicWidthInMbs – slice_group_change_direction_flag ) / 2
y = ( PicHeightInMapUnits – slice_group_change_direction_flag ) / 2
( leftBound, topBound ) = ( x, y )
( rightBound, bottomBound ) = ( x, y )
( xDir, yDir ) = ( slice_group_change_direction_flag – 1, slice_group_change_direction_flag )
pour( k = 0; k < MapUnitsInSliceGroup0; k += mapUnitVacant ) {
    mapUnitVacant = ( mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] == 1 )
    si( mapUnitVacant )
        mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = 0                    (8-21)
    si( xDir == –1 && x == leftBound ) {
        leftBound = Max( leftBound – 1, 0 )
        x = leftBound
        ( xDir, yDir ) = ( 0, 2 * slice_group_change_direction_flag – 1 )
    } ou si( xDir == 1 && x == rightBound ) {
        rightBound = Min( rightBound + 1, PicWidthInMbs – 1 )
        x = rightBound
        ( xDir, yDir ) = ( 0, 1 – 2 * slice_group_change_direction_flag )
    } ou si( yDir == –1 && y == topBound ) {
        topBound = Max( topBound – 1, 0 )
        y = topBound
        ( xDir, yDir ) = ( 1 – 2 * slice_group_change_direction_flag, 0 )
    } ou si( yDir == 1 && y == bottomBound ) {
        bottomBound = Min( bottomBound + 1, PicHeightInMapUnits – 1 )
        y = bottomBound
        ( xDir, yDir ) = ( 2 * slice_group_change_direction_flag – 1, 0 )
    } ou
        ( x, y ) = ( x + xDir, y + yDir )
}
```

8.2.2.5 Spécification pour les types de mappage de groupe de tranches à balayage matriciel

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 4.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

```
pour( i = 0; i < PicSizeInMapUnits; i++ )
    si( i < sizeOfUpperLeftGroup )
        mapUnitToSliceGroupMap[ i ] = slice_group_change_direction_flag
    ou                                                                         (8-22)
        mapUnitToSliceGroupMap[ i ] = 1 – slice_group_change_direction_flag
```

8.2.2.6 Spécification pour les types de mappage de groupe de tranches de balayage

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 5.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

```
k = 0;
pour( j = 0; j < PicWidthInMbs; j++ )
    pour( i = 0; i < PicHeightInMapUnits; i++ )
        si( k++ < sizeOfUpperLeftGroup )
            mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = slice_group_change_direction_flag
        ou                                                                     (8-23)
            mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = 1 – slice_group_change_direction_flag
```

8.2.2.7 Spécification pour type de mappage explicite de groupe de tranches

Les spécifications du présent paragraphe s'appliquent lorsque slice_group_map_type est égal à 6.

Le mappage d'unité de mappage à groupe de tranches est produit comme spécifié par:

$$mapUnitToSliceGroupMap[i] = slice\_group\_id[i] \quad (8\text{-}24)$$

pour tout i dans la gamme de 0 à PicSizeInMapUnits – 1 inclus.

8.2.2.8 Spécification pour conversion d'unité de mappage de groupe de tranches à macrobloc

Pour chaque valeur de i dans la gamme de 0 à PicSizeInMbs – 1 inclus, le mappage de macrobloc à groupe de tranches est spécifié comme suit.

– Si frame_mbs_only_flag est égal à 1 ou field_pic_flag est égal à 1, le mappage de macrobloc à groupe de tranches est spécifié par:

$$MbToSliceGroupMap[i] = mapUnitToSliceGroupMap[i] \quad (8\text{-}25)$$

– Sinon, si MbaffFrameFlag est égal à 1, le mappage de macrobloc à groupe de tranches est spécifié par:

$$MbToSliceGroupMap[i] = mapUnitToSliceGroupMap[i / 2] \quad (8\text{-}26)$$

– Sinon (si frame_mbs_only_flag est égal à 0 et mb_adaptive_frame_field_flag est égal à 0 et field_pic_flag est égal à 0), le mappage de macrobloc à groupe de tranches est spécifié par:

$$MbToSliceGroupMap[i] = mapUnitToSliceGroupMap[(i / (2 * PicWidthInMbs)) * PicWidthInMbs + (i \% PicWidthInMbs)] \quad (8\text{-}27)$$

8.2.3 Processus de décodage pour subdivision de données de tranche

Les entrées dans ce processus sont:

– une charge utile RBSP de couche de subdivision A de données de tranche;

– lorsque des éléments syntaxiques de catégorie 3 sont présents dans les données de tranche, une charge utile RBSP de couche de subdivision A de données de tranche ayant le même slice_id que dans la charge utile RBSP de couche de subdivision A de données de tranche;

– lorsque des éléments syntaxiques de catégorie 4 sont présents dans les données de tranche, une charge utile RBSP de couche de subdivision C de données de tranche ayant le même slice_id que dans la charge utile RBSP de couche de subdivision A de données de tranche.

NOTE 1 – La charge utile RBSP de couche de subdivision B de données de tranche et la charge utile RBSP de couche de subdivision C de données de tranche ne sont pas nécessairement présentes.

Le résultat de ce processus est une tranche codée.

Lorsque la subdivision de données de tranche n'est pas utilisée, les tranches codées sont représentées par une couche de tranche sans subdivision de charge utile RBSP contenant un en-tête de tranche suivi par une structure syntaxique de données de tranche qui contient tous les éléments syntaxiques des catégories 2, 3 et 4 (voir la colonne catégorie au § 7.3) des données de macrobloc pour les macroblocs de la tranche.

Lorsque la subdivision de données de tranche est utilisée, les données de macrobloc d'une tranche sont subdivisées en une, deux ou trois subdivisions contenues dans des unités NAL séparées. La subdivision A contient un en-tête de subdivision A de données de tranche et tous les éléments syntaxiques de catégorie 2. La subdivision B, lorsqu'elle est présente, contient un en-tête de subdivision B de données de tranche et tous les éléments syntaxiques de catégorie 3. La subdivision C, lorsqu'elle est présente, contient un en-tête de subdivision C de données de tranche et tous les éléments syntaxiques de catégorie 4.

Lorsque la subdivision de données de tranche est utilisée, les éléments syntaxiques de chaque catégorie sont analysés syntaxiquement à partir d'une unité NAL séparée, qui ne doit pas nécessairement être présente lorsque n'existe aucun symbole de leurs catégories respectives. Le processus de décodage doit traiter les subdivisions de données de tranche d'une tranche codée d'une manière équivalente au traitement d'une couche de tranche correspondante sans subdivision de charge utile RBSP en extrayant chaque élément syntaxique de la subdivision de données de tranche dans lequel apparaît l'élément syntaxique en fonction de l'allocation de subdivision de données de tranche dans les tableaux syntaxiques du § 7.3.

NOTE 2 – Les éléments syntaxiques de catégorie 3 sont pertinents pour le décodage des données résiduelles des types de macrobloc I et SI. Les éléments syntaxiques de catégorie 4 sont pertinents pour le décodage des données résiduelles des types de

macrobloc P et B. La catégorie 2 englobe tous les autres éléments syntaxiques relatifs au décodage de macroblocs et leurs informations sont souvent mentionnées comme informations d'en-tête. L'en-tête de subdivision A de données de tranche contient tous les éléments syntaxiques de l'en-tête de tranche et en plus les slice_id qui sont utilisés pour associer les subdivisions B et C de données de tranche avec la subdivision A de données de tranche. Les en-têtes de subdivision B et C de données de tranche contiennent l'élément syntaxique slice_id qui établit leur association avec la subdivision de données de tranche A de la tranche.

8.2.4 Processus de décodage pour la construction des listes d'images de référence

Ce processus est invoqué au début du décodage de chaque tranche P, SP ou B.

Les images décodées de référence sont marquées "utilisée comme référence à court terme" ou "utilisée comme référence à long terme" comme spécifié par le flux binaire et décrit au § 8.2.5. Les images de référence à court terme sont identifiées par la valeur de frame_num. Un indice de trame à long terme est alloué aux images de référence à long terme comme spécifié par le flux binaire et indiqué au § 8.2.5.

Le paragraphe 8.2.4.1est invoqué afin de spécifier:

– l'allocation des variables FrameNum, FrameNumWrap, et PicNum à chacune des images de référence à court terme;

– l'allocation de la variable LongTermPicNum à chacune des images de référence à long terme.

Les images de référence sont indexées par des indices de référence comme spécifié au § 8.4.2.1. Un indice de référence est un indice qui pointe dans une liste d'images de référence. Lors du décodage d'une tranche P ou SP, il y a une seule liste d'images de référence RefPicList0. Lors du décodage d'une tranche B, il y a une seconde liste indépendante d'images de référence RefPicList1 en plus de RefPicList0.

Au début du décodage de chaque tranche, la liste d'images de référence RefPicList0 et RefPicList1 pour les tranches B, est déduite comme suit.

– Une liste initiale d'images de référence RefPicList0 et RefPicList1 pour les tranches B, est déduite comme spécifié au § 8.2.4.2.

– La liste initiale d'images de référence RefPicList0 et RefPicList1 pour les tranches B, est modifiée comme spécifié au § 8.2.4.3.

  NOTE – Le processus de réordonnancement pour listes d'images de référence qui est spécifié dans le § 8.2.4.3 permet de modifier de façon flexible le contenu des listes RefPicList0 et, pour les tranches B, le contenu des listes RefPicList1. En particulier, il est possible d'insérer une image actuellement marquée "utilisée comme référence" dans une liste RefPicList0 et, pour les tranches B, dans une liste RefPicList1 même lorsque l'image n'est pas dans la liste initiale d'images de référence calculée comme spécifié dans le § 8.2.4.2.

Le nombre des entrées dans la liste modifiée d'images de référence RefPicList0 est num_ref_idx_l0_active_minus1 + 1 et pour les tranches B le nombre des entrées dans la liste modifiée d'images de référence RefPicList1 est num_ref_idx_l1_active_minus1 + 1. Une image de référence peut apparaître à plus d'un indice dans les listes modifiées d'images de référence RefPicList0 ou RefPicList1.

8.2.4.1 Processus de décodage pour les numéros d'image

Ce processus est invoqué lorsque le processus de décodage pour la construction des listes d'images de référence spécifié au § 8.2.4 ou le processus de marquage d'image de référence décodée spécifié au § 8.2.5 est invoqué.

Les variables FrameNum, FrameNumWrap, PicNum, LongTermFrameIdx et LongTermPicNum sont utilisées pour le processus d'initialisation pour les listes d'images de référence du § 8.2.4.2, le processus de modification pour les listes d'images de référence du § 8.2.4.3 et pour le processus de marquage d'image de référence décodée du § 8.2.5.

Les variables FrameNum et FrameNumWrap de chaque image de référence à court terme sont allouées comme suit. D'abord, FrameNum est mis égal à l'élément syntaxique frame_num qui a été décodé dans l'en-tête de tranche(s) de l'image de référence à court terme correspondante. Puis la variable FrameNumWrap est déduite comme

```
si( FrameNum > frame_num )
    FrameNumWrap = FrameNum – MaxFrameNum                    (8-28)
ou
    FrameNumWrap = FrameNum
```

où la valeur de frame_num utilisée dans l'équation 8-28 est le frame_num de l'en-tête de tranche(s) pour l'image en cours.

Chaque image de référence à long terme possède une valeur associée de l'élément LongTermFrameIdx(qui lui a été attribuée comme spécifié au § 8.2.5).

Une variable PicNum est allouée à chaque image de référence à court terme et une variable LongTermPicNum est allouée à chaque image de référence à long terme. Les valeurs de ces variables dépendent de la valeur de field_pic_flag et bottom_field_flag pour l'image en cours et elles sont établies comme suit.

– Si field_pic_flag est égal à 0, on applique ce qui suit.
  – Pour chaque trame de référence à court terme ou paire de sous-trames de référence complémentaires:

$$PicNum = FrameNumWrap \tag{8-29}$$

  – Pour chaque trame de référence à long terme ou paire de sous-trames de référence complémentaires à long terme:

$$LongTermPicNum = LongTermFrameIdx \tag{8-30}$$

NOTE – Lors du décodage d'une trame, la valeur de MbaffFrameFlag n'a pas d'influence sur les déductions des § 8.2.4.2, 8.2.4.3 et 8.2.5.

– Sinon (si field_pic_flag est égal à 1), on applique ce qui suit.
  – Pour chaque sous-trame de référence à court terme, on applique ce qui suit.
    – Si la sous-trame de référence a la même parité que le sous-trame en cours

$$PicNum = 2 * FrameNumWrap + 1 \tag{8-31}$$

    – Sinon (si la sous-trame de référence a la parité opposée à celle de la sous-trame en cours)

$$PicNum = 2 * FrameNumWrap \tag{8-32}$$

  – Pour chaque sous-trame de référence à long terme, on applique ce qui suit.
    – Si la sous-trame de référence a la même parité que la sous-trame en cours

$$LongTermPicNum = 2 * LongTermFrameIdx + 1 \tag{8-33}$$

    – Sinon (si la sous-trame de référence a la parité opposée à celle de la sous-trame en cours)

$$LongTermPicNum = 2 * LongTermFrameIdx \tag{8-34}$$

8.2.4.2 Processus d'initialisation pour les listes d'images de référence

Ce processus d'initialisation est invoqué lors du décodage d'un en-tête de tranche P, SP ou B.

RefPicList0 et RefPicList1 ont des entrées initiales comme spécifié aux § 8.2.4.2.1 à 8.2.4.2.5.

Lorsque le nombre d'entrées dans les RefPicList0 ou RefPicList1 initiales produites comme spécifié aux § 8.2.4.2.1 à 8.2.4.2.5 est supérieur à respectivement, num_ref_idx_l0_active_minus1 + 1 ou num_ref_idx_l1_active_minus1 + 1, les anciennes positions d'entrées num_ref_idx_l0_active_minus1 ou num_ref_idx_l1_active_minus1 sont détruites de la liste initiale d'image de référence.

Lorsque le nombre d'entrées dans la RefPicList0 ou RefPicList1 initiale produites comme spécifié aux § 8.2.4.2.1 à 8.2.4.2.5 est inférieur respectivement à num_ref_idx_l0_active_minus1 + 1 ou num_ref_idx_l1_active_minus1 + 1, les entrées restantes dans la liste initiale d'image de référence sont mises égales à "pas d'image de référence".

8.2.4.2.1 Processus d'initialisation pour la liste d'images de référence pour les tranches P et SP en trames

Ce processus d'initialisation est invoqué lors du décodage d'une tranche P ou SP dans une trame codée.

Lorsque ce processus est invoqué, il doit y avoir au moins une trame de référence ou une paire de sous-trames de référence complémentaires qui soit actuellement marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme".

La liste d'images de référence RefPicList0 est ordonnée de telle sorte que les trames de référence à court terme et les paires de sous-trames de référence complémentaires à court terme aient des indices inférieurs à ceux des trames de référence à long terme et des paires de sous-trames de référence complémentaires à long terme.

Les trames de référence et les paires de sous-trames de référence complémentaires à court terme sont ordonnées à partir de la trame ou paire de sous-trames complémentaires avec la plus forte valeur de PicNum et ainsi de suite en ordre décroissant jusqu'à la trame ou paire de sous-trames complémentaires avec la plus faible valeur de PicNum.

Les trames de référence et paires de sous-trames de référence complémentaires à long terme sont ordonnées à partir de la trame ou paire de sous-trames complémentaires avec la plus faible valeur de LongTermPicNum value et ainsi de suite en ordre croissant jusqu'à la trame ou paire de sous-trames complémentaires avec la plus forte valeur de LongTermPicNum.

NOTE – Une sous-trame de référence non appariée n'est pas utilisée pour l'interprédiction pour le décodage d'une trame, indépendamment de la valeur de MbaffFrameFlag.

Par exemple, lorsque trois trames de référence sont marquées "utilisée comme référence à court terme" avec PicNum égal à 300, 302 et 303 et que deux trames de référence sont marquées "utilisée comme référence à long terme" avec LongTermPicNum égal à 0 et 3, l'ordre initial des indices est:

– RefPicList0[0] est mis égal à l'image de référence à court terme pour PicNum = 303;

– RefPicList0[1] est mis égal à l'image de référence à court terme pour PicNum = 302;

– RefPicList0[2] est mis égal à l'image de référence à court terme pour PicNum = 300;

– RefPicList0[3] est mis égal à l'image de référence à long terme pour LongTermPicNum = 0;

– RefPicList0[4] est mis égal à l'image de référence à long terme pour LongTermPicNum = 3.

8.2.4.2.2 Processus d'initialisation pour la liste d'images de référence pour des tranches P et SP en sous-trames

Ce processus d'initialisation est invoqué lors du décodage d'une tranche P ou SP dans une sous-trame codée.

Chaque sous-trame incluse dans la liste d'images de référence RefPicList0 a un indice séparé dans la liste d'images de référence RefPicList0.

NOTE – Lors du décodage d'une sous-trame, il y a effectivement au moins deux fois autant d'images disponibles à référencer qu'il y en aurait lors du décodage d'une trame à la même position dans l'ordre de décodage.

Deux listes ordonnées de trames de référence, refFrameList0ShortTerm et refFrameList0LongTerm, sont déduites comme suit. Pour les besoins de la formation de cette liste de trames, les trames de référence décodées, les paires de sous-trames de référence complémentaires, les sous-trames de référence non appariées et les trames de référence dans lesquelles une seule sous-trame est marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme" sont toutes considérées comme des trames de référence.

– Toutes les trames ayant une ou plusieurs sous-trames marquées "utilisée comme référence à court terme" sont incluses dans la liste de trames de référence à court terme refFrameList0ShortTerm. Lorsque la sous-trame en cours est la seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames de référence complémentaires et que la première sous-trame est marquée "utilisée comme référence à court terme", cette première sous-trame est incluse dans la liste de trames de référence à court terme refFrameList0ShortTerm, laquelle est ordonnée à partir de la trame de référence ayant la valeur FrameNumWrap la plus élevée et ainsi de suite en ordre décroissant jusqu'à la trame de référence ayant la valeur FrameNumWrap la plus basse.

– Toutes les trames ayant une ou plusieurs sous-trames marquées "utilisée comme référence à long terme" sont incluses dans la liste des trames de référence à long terme refFrameList0LongTerm. Lorsque la sous-trame en cours est la seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames de référence complémentaires et que la première sous-trame est marquée "utilisée comme référence à long terme", cette première sous-trame est incluse dans la liste de trames de référence à long terme refFrameList0LongTerm, laquelle est ordonnée à partir de la trame de référence ayant la valeur de LongTermFrameIdx la plus basse et ainsi de suite en ordre croissant jusqu'à la trame de référence ayant la valeur de LongTermFrameIdx la plus élevée.

Le processus spécifié au § 8.2.4.2.5 est invoqué avec refFrameList0ShortTerm et refFrameList0LongTerm donnés en entrée et le résultat est alloué à RefPicList0.

8.2.4.2.3 Processus d'initialisation des listes d'images de référence pour les tranches B en trames

Ce processus d'initialisation est invoqué lors du décodage d'une tranche B dans une trame codée.

Aux fins de la formation des listes d'images de référence RefPicList0 et RefPicList1, le terme *entrée de référence* se rapporte ci-dessous à des trames de référence décodées ou à des paires de sous-trames de référence complémentaires.

Lorsque ce processus est invoqué, il doit y avoir au moins une trame de référence ou une entrée de référence qui soit actuellement marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme".

Pour les tranches B, l'ordre des entrées de référence à court terme dans les listes d'images de référence RefPicList0 et RefPicList1 dépend de l'ordre de sortie, tel qu'il est donné par PicOrderCnt( ). Lorsque pic_order_cnt_type est égal à 0, les images de référence qui sont marquées "non existantes" comme spécifié au § 8.2.5.2 ne sont incluses ni dans RefPicList0 ni dans RefPicList1.

NOTE 1 – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, les codeurs devraient utiliser la réorganisation de la liste d'images de référence afin d'assurer le bon fonctionnement du processus de décodage (notamment lorsque pic_order_cnt_type est égal à 0, auquel cas PicOrderCnt( ) n'est pas déduit pour les trames "non existantes").

La liste d'images de référence RefPicList0 est ordonnée de façon que les entrées de référence à court terme aient des indices moins grands que les entrées de référence à long terme. Cette liste est ordonnée comme suit:

– Soit entryShortTerm une variable dont l'étendue contient toutes les entrées de référence qui sont actuellement marquées v"utilisée comme référence à court terme". Lorsque certaines valeurs de entryShortTerm sont présentes et utilisent une valeur PicOrderCnt(entryShortTerm) inférieure à celle de PicOrderCnt(CurrPic), ces valeurs de entryShortTerm sont placées au début de refPicList0 dans l'ordre décroissant de PicOrderCnt(entryShortTerm). Toutes les valeurs restantes de entryShortTerm (lorsqu'elles sont présentes) sont alors ajoutées à refPicList0 dans l'ordre croissant de PicOrderCnt(entryShortTerm).

– Les entrées de référence à long terme sont ordonnées à partir de l'entrée de référence à long terme qui possède la plus basse valeur de LongTermPicNum et ainsi de suite dans l'ordre croissant jusqu'à l'entrée de référence à long terme qui possède la plus haute valeur de LongTermPicNum.

La liste d'images de référence RefPicList1 est ordonnée de façon que les entrées de référence à court terme aient des indices moins grands que les entrées de référence à long terme. Cette liste est ordonnée comme suit.

– Soit entryShortTerm une variable dont l'étendue contient toutes les entrées de référence qui sont actuellement marquées "utilisée comme référence à court terme". Lorsque certaines valeurs de entryShortTerm sont présentes et utilisent une valeur de PicOrderCnt(entryShortTerm) plus grande que celle de PicOrderCnt(CurrPic), ces valeurs de entryShortTerm sont placées au début de la liste refPicList1 dans l'ordre croissant de PicOrderCnt(entryShortTerm). Toutes les valeurs restantes de entryShortTerm (lorsqu'elles sont présentes) sont alors ajoutées à refPicList1 dans l'ordre décroissant de PicOrderCnt(entryShortTerm).

– Les entrées de référence à long terme sont ordonnées à partir de la trame de référence ou paire de sous-trames de référence complémentaires à long terme qui a la plus faible valeur de LongTermPicNum et ainsi de suite en ordre croissant jusqu'à l'entrée de référence à long terme qui a la plus forte valeur de LongTermPicNum.

– Lorsque la liste d'images de référence RefPicList1 a plus d'une entrée et que RefPicList1 est identique à la liste d'images de référence list RefPicList0, les deux premières entrées RefPicList1[0] et RefPicList1[1] sont interverties.

NOTE 2 – Une sous-trame de référence non appariée ne sera pas utilisée pour l'interprédiction de trames indépendamment de la valeur de MbaffFrameFlag.

8.2.4.2.4 Processus d'initialisation des listes d'images de référence pour les tranches B en sous-trames

Ce processus d'initialisation est invoqué lors du décodage d'une tranche B dans une sous-trame codée.

Lors du décodage d'une sous-trame, chaque sous-trame d'une trame de référence mémorisée est identifiée comme une image de référence séparée avec un indice unique. L'ordre des images de référence à court terme dans les listes d'images de référence RefPicList0 et RefPicList1 dépend de l'ordre de sortie, comme donné par PicOrderCnt( ). Lorsque pic_order_cnt_type est égal à 0, les images de référence qui sont marquées "non existantes" comme spécifié au § 8.2.5.2 ne sont incluses ni dans RefPicList0 ni dans RefPicList1.

NOTE 1 – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, les codeurs devraient utiliser la réorganisation de la liste d'images de référence afin d'assurer le bon fonctionnement du processus de décodage (notamment lorsque pic_order_cnt_type est égal à 0, auquel cas PicOrderCnt( ) n'est pas déduit pour les trames "non existantes").

NOTE 2 – Lors du décodage d'une sous-trame, il y a effectivement au moins deux fois autant d'images disponibles pour référence qu'il y en aurait lors du décodage d'une trame à la même position dans l'ordre de décodage.

Trois listes ordonnées de trames de référence, refFrameList0ShortTerm, refFrameList1ShortTerm et refFrameListLongTerm, sont déduites comme suit. Pour les besoins de la formation de ces listes de trames, le terme *entrée de référence* désigne dans ce qui suit les trames de référence décodées, les paires de sous-trames de référence complémentaires ou les sous-trames de référence non appariées. Lorsque pic_order_cnt_type est égal à 0, le terme *entrée de référence* ne désigne pas les trames qui sont marquées "non existantes" comme spécifié au § 8.2.5.2.

– Soit entryShortTerm une variable dont l'étendue contient toutes les entrées de référence qui sont actuellement marquées comme étant "utilisée comme référence à court terme". Lorsque certaines valeurs de entryShortTerm sont présentes et utilisent une valeur de PicOrderCnt(entryShortTerm) inférieure ou égale à celle PicOrderCnt(CurrPic), ces valeurs de entryShortTerm sont placées au début de refFrameList0ShortTerm dans l'ordre décroissant de PicOrderCnt(entryShortTerm). Toutes les valeurs restantes de entryShortTerm (lorsqu'elles

sont présentes) sont alors ajoutées à refFrameList0ShortTerm dans l'ordre croissant de PicOrderCnt(entryShortTerm).

NOTE 3 – Lorsque la sous-trame en cours suit dans l'ordre de décodage une sous-trame codée fldPrev avec laquelle elle forme une paire de sous-trames de référence complémentaires, fldPrev est inclus dans la liste refFrameList0ShortTerm au moyen de PicOrderCnt(fldPrev) et on applique la méthode de classement décrite à la phrase précédente.

– Soit entryShortTerm une variable dont l'étendue contient toutes les entrées de référence qui sont actuellement marquées "utilisée comme référence à court terme". Lorsque certaines valeurs de entryShortTerm sont présentes et utilisent une valeur de PicOrderCnt(entryShortTerm) plus grande que celle de PicOrderCnt(CurrPic), ces valeurs de entryShortTerm sont placées au début de la liste refFrameList1ShortTerm dans l'ordre croissant de PicOrderCnt(entryShortTerm). Toutes les valeurs restantes de entryShortTerm (lorsqu'elles sont présentes) sont alors ajoutées à refFrameList1ShortTerm dans l'ordre décroissant de PicOrderCnt(entryShortTerm).

NOTE 4 – Lorsque la sous-trame en cours suit dans l'ordre de décodage une sous-trame codée fldPrev avec laquelle elle forme une paire de sous-trames de référence complémentaires, fldPrev est inclus dans la liste refFrameList1ShortTerm au moyen de PicOrderCnt(fldPrev) et on applique la méthode de classement décrite à la phrase précédente.

– refFrameListLongTerm est ordonnée à partir de l'entrée de référence ayant la plus faible valeur de LongTermFrameIdx et ainsi de suite en ordre croissant jusqu'à l'entrée de référence ayant la plus forte valeur de LongTermFrameIdx.

NOTE 5 – Lorsque la sous-trame complémentaire de l'image en cours est marquée "utilisée comme référence à long terme", elle est incluse dans la liste refFrameListLongTerm. Une entrée de référence dans laquelle une seule sous-trame est marquée "utilisée comme référence à long terme" est incluse dans la liste refFrameListLongTerm.

Le processus spécifié au § 8.2.4.2.5 est invoqué avec refFrameList0ShortTerm et refFrameListLongTerm donnés en entrée et le résultat est alloué à RefPicList0.

Le processus spécifié au § 8.2.4.2.5 est invoqué avec refFrameList1ShortTerm et refFrameListLongTerm donnés en entrée et le résultat est alloué à RefPicList1.

Lorsque la liste d'images de référence RefPicList1 a plus d'une entrée et que RefPicList1 est identique à la liste d'images de référence RefPicList0, les deux premières entrées RefPicList1[0] et RefPicList1[1] sont interverties.

8.2.4.2.5 Processus d'initialisation pour les listes d'images de référence en sous-trames

Les entrées dans ce processus sont les listes de trame de référence refFrameListXShortTerm (X pouvant être 0 ou 1) et refFrameListLongTerm.

La liste d'images de référence RefPicListX est une liste ordonnée de telle sorte que les sous-trames de référence à court terme aient de plus faibles indices que les sous-trames de référence à long terme. Etant donné les listes de trames de référence refFrameListXShortTerm et refFrameListLongTerm, elle est déduite comme suit.

– Les sous-trames de référence à court terme sont ordonnées en choisissant des sous-trames de référence à partir de la liste ordonnée de trames refFrameListXShortTerm et en alternant entre sous-trames de parité différente, à partir d'une sous-trame qui a la même parité que la sous-trame en cours lorsque celle-ci est présente. Lorsqu'une sous-trame d'une trame de référence n'a pas été décodée ou n'est pas marquée "utilisée comme référence à court terme", la sous-trame manquante est ignorée et on insère à sa place dans RefPicListX la prochaine sous-trame de référence enregistrée qui est disponible avec la parité choisie dans la liste ordonnée de trames refFrameListXShortTerm. Lorsqu'il n'y a plus de sous-trame de référence à court terme de parité alternée dans la liste ordonnée de trames refFrameListXShortTerm, les prochaines sous-trames non encore indexées de la parité disponible sont insérées dans RefPicListX dans l'ordre de leur apparition dans la liste ordonnée de trames refFrameListXShortTerm.

– Les sous-trames de référence à long terme sont ordonnées en choisissant des sous-trames de référence à partir de la liste ordonnée de trames refFrameListLongTerm et en alternant entre sous-trames de différente parité, à partir d'une sous-trame qui a la même parité que la sous-trame en cours lorsque celle-ci est présente. Lorsqu'une sous-trame d'une trame de référence n'a pas été décodée ou n'est pas marquée "utilisée comme référence à long terme", la sous-trame manquante est ignorée et on insère à sa place dans RefPicListX la prochaine sous-trame de référence enregistrée qui est disponible avec la parité choisie dans la liste ordonnée de trames refFrameListLongTerm. Lorsqu'il n'y a plus de sous-trame de référence à long terme de parité alternée dans la liste ordonnée de trames refFrameListLongTerm, les prochaines sous-trames non encore indexées de la parité disponible sont insérées dans RefPicListX dans l'ordre de leur apparition dans la liste ordonnée de trames refFrameListLongTerm.

8.2.4.3 Processus de remise en ordre des listes d'images de référence

Lorsque ref_pic_list_reordering_flag_l0 est égal à 1, on applique ce qui suit.

– Soit refIdxL0 un indice dans la liste d'images de référence RefPicList0. Il est au départ mis égal à 0.

– Les éléments syntaxiques reordering_of_pic_nums_idc correspondants sont traités dans l'ordre dans lequel ils surviennent dans le flux binaire. Pour chacun de ces éléments syntaxiques, on applique ce qui suit.

  – Si reordering_of_pic_nums_idc est égal à 0 ou à 1, le processus spécifié au § 8.2.4.3.1 est invoqué avec refIdxL0 en entrée et le résultat est alloué à refIdxL0.

  – Sinon, si reordering_of_pic_nums_idc est égal à 2, le processus spécifié au § 8.2.4.3.2 est invoqué avec refIdxL0 en entrée et le résultat est alloué à refIdxL0.

  – Sinon (si reordering_of_pic_nums_idc est égal à 3), le processus de remise en ordre de la liste d'images de référence RefPicList0 est terminé.

Lorsque ref_pic_list_reordering_flag_l1 est égal à 1, on applique ce qui suit.

– Soit refIdxL1 un indice dans la liste d'images de référence RefPicList1. Il est au départ mis égal à 0.

– Les éléments syntaxiques reordering_of_pic_nums_idc correspondants sont traités dans l'ordre dans lequel ils surviennent dans le flux binaire. Pour chacun de ces éléments syntaxiques, on applique ce qui suit.

  – Si reordering_of_pic_nums_idc est égal à 0 ou à 1, le processus spécifié au § 8.2.4.3.1 est invoqué avec refIdxL1 en entrée et le résultat est alloué à refIdxL1.

  – Sinon, si reordering_of_pic_nums_idc est égal à 2, le processus spécifié au § 8.2.4.3.2 est invoqué avec refIdxL1 en entrée et le résultat est alloué à refIdxL1.

  – Sinon (si reordering_of_pic_nums_idc est égal à 3), le processus de remise en ordre de la liste d'images de référence RefPicList1 est terminé.

8.2.4.3.1 Processus de remise en ordre des listes d'images de référence pour images à court terme

L'entrée dans ce processus est un indice refldxLX (X étant 0 ou 1).

Les résultats de ce processus est un indice refIdxLX incrémenté.

La variable picNumLXNoWrap est déduite comme suit:

– Si reordering_of_pic_nums_idc est égal à 0

si(picNumLXPred – (abs_diff_pic_num_minus1 + 1) < 0)
picNumLXNoWrap = picNumLXPred – (abs_diff_pic_num_minus1 + 1) + MaxPicNum (8-35)
ou
picNumLXNoWrap = picNumLXPred – (abs_diff_pic_num_minus1 + 1)

– Sinon (si reordering_of_pic_nums_idc est égal à 1)

si(picNumLXPred + (abs_diff_pic_num_minus1 + 1) >= MaxPicNum)
picNumLXNoWrap = picNumLXPred + (abs_diff_pic_num_minus1 + 1) – MaxPicNum (8-36)
ou
picNumLXNoWrap = picNumLXPred + (abs_diff_pic_num_minus1 + 1)

picNumLXPred est la valeur de prédiction pour la variable picNumLXNoWrap. Lorsque le processus spécifié au présent paragraphe est invoqué la première fois pour une tranche (c'est-à-dire, pour la première occurrence de reordering_of_pic_nums_idc égal à 0 ou 1 dans la syntaxe ref_pic_list_reordering()), picNumL0Pred et picNumL1Pred sont au départ mis égaux à CurrPicNum. Après chaque allocation de picNumLXNoWrap, la valeur de picNumLXNoWrap est allouée à picNumLXPred.

La variable picNumLX est déduite comme suit

si(picNumLXNoWrap > CurrPicNum)
picNumLX = picNumLXNoWrap – MaxPicNum (8-37)
ou
picNumLX = picNumLXNoWrap

picNumLX doit être égal au PicNum d'une image de référence qui est marquée "utilisée comme référence à court terme" et doit être différent du PicNum d'une image de référence à court terme qui est marquée "non existante".

La procédure suivante est appliquée afin de placer l'image avec le numéro d'image à court terme picNumLX dans la position d'indice refIdxLX, déplacer la position de toutes autres images restantes plus loin dans la liste et incrémenter la valeur de refIdxLX.

```
pour( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx – 1]
RefPicListX[ refIdxLX++ ] = short-term reference picture with PicNum equal to picNumLX
nIdx = refIdxLX
pour( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )                    (8-38)
    si( PicNumF( RefPicListX[ cIdx ] ) != picNumLX )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

où la fonction PicNumF(RefPicListX[cIdx]) est déduite comme suit:

– si l'image RefPicListX[cIdx] est marquée "utilisée comme référence à court terme", PicNumF(RefPicListX[cIdx]) est le PicNum de l'image RefPicListX[cIdx];

– Sinon (si l'image RefPicListX[cIdx] n'est pas marquée "utilisée comme référence à court terme"), PicNumF(RefPicListX[cIdx]) est égal à MaxPicNum.

NOTE 1 – Une valeur de MaxPicNum ne peut jamais être égale à picNumLX.

NOTE 2 – Au sein de cette procédure de pseudocode, la longueur de la liste RefPicListX est temporairement allongée d'un élément par rapport à la longueur nécessaire pour la liste finale. Après l'exécution de cette procédure, seuls les éléments de 0 à num_ref_idx_lX_active_minus1 de la liste doivent être retenus.

8.2.4.3.2 Processus de remise en ordre des listes d'images de référence pour les images à long terme

L'entrée dans ce processus est un indice refIdxLX (X étant 0 ou 1).

Le résultat de ce processus est un indice incrémenté refIdxLX.

La procédure suivante est appliquée afin de placer l'image avec le numéro d'image à long terme long_term_pic_num dans la position d'indice refIdxLX, déplacer la position de toutes autres images restantes plus loin dans la liste et incrémenter la valeur de refIdxLX.

```
pour( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
      RefPicListX[ cIdx ] = RefPicListX[ cIdx – 1]
RefPicListX[ refIdxLX++ ] = image de référence à court terme avec LongTermPicNum égal à
long_term_pic_num
nIdx = refIdxLX
pour( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )                    (8-39)
      si( LongTermPicNumF( RefPicListX[ cIdx ] ) != long_term_pic_num )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

où la fonction LongTermPicNumF(RefPicListX[cIdx]) est déduite comme suit:

– Si l'image RefPicListX[cIdx] est marquée "utilisée comme référence à long terme", LongTermPicNumF(RefPicListX[cIdx]) est le LongTermPicNum de l'image RefPicListX[cIdx].

– Sinon (si l'image RefPicListX[cIdx] n'est pas marquée "utilisée comme référence à long terme"), LongTermPicNumF(RefPicListX[cIdx]) est égal à 2 * (MaxLongTermFrameIdx + 1).

NOTE 1 – Une valeur de 2 * (MaxLongTermFrameIdx + 1) ne peut jamais être égale à long_term_pic_num.

NOTE 2 – Au sein de cette procédure de pseudo-code, la longueur de la liste RefPicListX est temporairement allongée d'un élément par rapport à la longueur nécessaire pour la liste finale. Après l'exécution de cette procédure, seuls les éléments de 0 à num_ref_idx_lX_active_minus1 de la liste doivent être retenus.

8.2.5 Processus de marquage d'image de référence décodée

Ce processus est invoqué pour des images décodées lorsque nal_ref_idc n'est pas égal à 0.

NOTE – Le processus de décodage pour les lacunes dans frame_num, qui est spécifié au § 8.2.5.2, peut aussi être invoqué lorsque nal_ref_idc est égal à 0, comme spécifié au § 8.

Une image décodée avec nal_ref_idc différent de 0, mentionnée comme image de référence, est marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme". Pour une trame de référence décodée, ses

deux sous-trames sont marquées de la même façon qu'une trame. Pour une paire de sous-trames de référence complémentaires, la paire est marquée de la même façon que ses deux sous-trames. Une image qui est marquée "utilisée comme référence à court terme" est identifiée par son FrameNum et, lorsque c'est une sous-trame, par sa parité. Une image qui est marquée "utilisée comme référence à long terme" est identifiée par son LongTermFrameIdx et, lorsque c'est une sous-trame, par sa parité.

Les trames ou paires de sous-trames complémentaires marquées "utilisée comme référence à court terme" ou "utilisée comme référence à long terme" peuvent être utilisée comme référence pour l'interprédiction lors du décodage d'une trame jusqu'à ce que la trame, la paire de sous-trames complémentaires ou une de ses sous-trames constituantes soit marquée "inutilisée comme référence". Une sous-trame marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme" peut être utilisée comme référence pour l'interprédiction lors du décodage d'une sous-trame jusqu'à ce qu'elle soit marquée "inutilisée comme référence".

Une image peut être marquée "inutilisée comme référence" par le processus de marquage d'image de référence en fenêtre glissante, qui est un mécanisme de "premier entré, premier sorti" spécifié au § 8.2.5.3 ou par le processus de marquage d'image de référence à commande adaptative de mémoire, qui est une opération de marquage adaptatif personnalisée spécifiée au § 8.2.5.4.

Une image de référence à court terme est identifiée afin d'être utilisée dans le processus de décodage par ses variables FramNum et FrameNumWrap et par son numéro d'image PicNum et une image de référence à long terme est identifiée afin d'être utilisée dans le processus de décodage par son numéro d'image à long terme LongTermPicNum. Lorsque l'image en cours n'est pas à rafraîchissement IDR, le § 8.2.4.1 est invoqué afin de spécifier l'allocation des variables FrameNum, FrameNumWrap, PicNum et LongTermPicNum.

8.2.5.1 Séquence des opérations pour le processus de marquage d'image de référence décodée

Le marquage d'image de référence décodée s'effectue suivant les étapes ordonnées ci-après.

1. Toutes les tranches de l'image en cours sont décodées.
2. Selon que l'image en cours est ou non à rafraîchissement IDR, on applique ce qui suit.
   - Si l'image en cours est à rafraîchissement IDR, on applique ce qui suit.
     - Toutes les images de référence sont marquées "inutilisée comme référence".
     - En fonction de long_term_reference_flag, on applique ce qui suit.
       - Si long_term_reference_flag est égal à 0, l'image à rafraîchissement IDR est marquée "utilisée comme référence à court terme" et MaxLongTermFrameIdx doit être mis égal à "pas d'indice de trame à long terme".
       - Sinon (si long_term_reference_flag est égal à 1), l'image à rafraîchissement IDR est marquée "utilisée comme référence à long terme", l'indice LongTermFrameIdx pour l'image à rafraîchissement IDR est mis égal à 0 et MaxLongTermFrameIdx est mis égal à 0.
   - Sinon (si l'image en cours n'est pas à rafraîchissement IDR), on applique ce qui suit.
     - Si adaptive_ref_pic_marking_mode_flag est égal à 0, le processus spécifié au § 8.2.5.3 est invoqué.
     - Sinon (si adaptive_ref_pic_marking_mode_flag est égal à 1), le processus spécifié au § 8.2.5.4 est invoqué.
3. Lorsque l'image en cours n'est pas uneimage à rafraîchissement IDR et qu'elle n'était pas marquée "utilisée comme référence à long terme" par memory_management_control_operation égal à 6, elle est marquée "utilisée comme référence à court terme".

Après le marquage de l'image de référence décodée en cours, le nombre total de trames avec au moins une sous-trame marquée "utilisée comme référence", plus le nombre de paires de sous-trames complémentaires avec au moins une sous-trame marquée "utilisée comme référence", plus le nombre de sous-trames non appariées marquées "utilisée comme référence", ne doit pas être supérieur à num_ref_frames.

8.2.5.2 Processus de décodage pour les lacunes dans frame_num

Ce processus est invoqué lorsque frame_num n'est pas égal à PrevRefFrameNum et n'est pas égal à (PrevRefFrameNum + 1) % MaxFrameNum.

NOTE 1 – Bien que ce processus soit spécifié dans un paragraphe du § 8.2.5 (qui définit un processus invoqué uniquement lorsque nal_ref_idc est différent de 0), ce processus peut aussi être invoqué lorsque nal_ref_idc est égal à 0 (comme spécifié à l'article 8). Les raisons de la présence de ce paragraphe dans la structure de la présente Recommandation | Norme internationale sont historiques.

NOTE 2 – Ce processus ne peut être invoqué que pour un flux binaire conforme lorsque gaps_in_frame_num_value_allowed_flag est égal à 1. Lorsque gaps_in_frame_num_value_allowed_flag est égal à 0 et frame_num n'est pas égal à PrevRefFrameNum et n'est pas égal à (PrevRefFrameNum + 1) % MaxFrameNum, le processus de décodage devrait en déduire une perte d'images non intentionnelle.

Lorsque ce processus est invoqué, un ensemble de valeurs de frame_num appartenant à des images "non existantes" est déduit comme étant toutes les valeurs prises par UnusedShortTermFrameNum dans l'équation 7-21 excepté la valeur de frame_num pour l'image en cours.

Le processus de décodage produit et marque une trame pour chacune des valeurs de frame_num appartenant aux images "non existantes", dans l'ordre selon lequel les valeurs de UnusedShortTermFrameNum sont produites par l'équation 7-21, au moyen du processus de marquage d'image à "fenêtre glissante" comme spécifié au § 8.2.5.3. Les trames produites sont aussi marquées "non existantes" et "utilisée comme référence à court terme". Les valeurs d'échantillon des trames produites peuvent être mises à n'importe quelle valeur. Le flux binaire ne doit pas contenir de données se traduisant par une référence à ces trames produites avec le marquage "non existantes" dans le processus d'interprédiction, par une référence à ces trames dans les commandes de réordonnancement de listes d'images de référence pour images de référence à court terme (§ 8.2.4.3.1), ou par une référence à ces trames dans le processus d'attribution d'un indice LongTermFrameIdx à une imae de référence à court terme (§ 8.2.5.4.3).Lorsque pic_order_cnt_type est différent de 0, TopFieldOrderCnt et BottomFieldOrderCnt sont calculés pour chacune des trames "non existantes" en invoquant le processus de décodage pour le compte d'ordre d'image visé au § 8.2.1. En cas d'invocation du processus visé au § 8.2.1 pour une trame "non existante" donnée, l'image en cours est considérée comme étant une image dont frame_num est supposé être égal à UnusedShortTermFrameNum, nal_ref_idc est supposé être égal à 0, nal_unit_type est supposé être égal à 5, field_pic_flag est supposé être égal à 0, adaptive_ref_pic_marking_mode_flag est supposé être égal à 0, delta_pic_order_cnt[0] (si nécessaire) est supposé être égal à 0 et delta_pic_order_cnt[1] (si nécessaire) est supposé être égal à 0.

NOTE 3 – Le processus de décodage devrait conclure à une perte d'image non intentionnelle lorsque toute valeur de frame_num appartenant à une image "non existante" est mentionnée dans le processus d'interprédiction, dans les commandes de remise en ordre des listes d'images de référence pour les images de référence à court terme (§ 8.2.4.3.1) ou dans le processus d'allocation d'un indice LongTermFrameIdx à une image de référence à court terme (§ 8.2.5.4.3). Le processus de décodage ne devrait pas conclure à une perte d'image non intentionnelle lorsqu'une opération de commande de gestion de mémoire non égale à 3 est appliquée à une trame marquée "non existante".

8.2.5.3 Processus de marquage d'image de référence décodée en fenêtre glissante

Ce processus est invoqué lorsque adaptive_ref_pic_marking_mode_flag est égal à 0.

En fonction des propriétés de l'image en cours spécifiée ci-dessous, on applique ce qui suit.

– Si l'image en cours est une sous-trame codée qui est la seconde sous-trame dans l'ordre de décodage d'une paire de sous-trames de référence complémentaires et que la première sous-trame ait été marquée "utilisée comme référence à court terme", l'image en cours est aussi marquée "utilisée comme référence à court terme".

– Sinon, on applique ce qui suit.

  – Soit numShortTerm le nombre total de trames de référence, de paires de sous-trames de référence complémentaires et de sous-trames de référence non appariées pour lesquelles au moins une sous-trame est marquée "utilisée comme référence à court terme". Soit numLongTerm le nombre total de trames de référence, de paires de sous-trames de référence complémentaires et de sous-trames de référence non appariées pour lesquelles au moins une sous-trame est marquée "utilisée comme référence à long terme".

  – Lorsque numShortTerm + numLongTerm est égal à Max(num_ref_frames, 1), la condition que numShortTerm soit supérieur à 0 doit être remplie et la trame de référence à court terme, la paire de sous-trames de référence complémentaires ou la sous-trame de référence non appariée qui a la plus faible valeur de FrameNumWrap est marquée "inutilisée comme référence". Lorsque c'est une trame ou une paire de sous-trames complémentaires, ses deux sous-trames sont aussi marquées "inutilisée comme référence".

8.2.5.4 Processus de marquage d'image de référence décodée en commande de mémoire adaptative

Ce processus est invoqué lorsque adaptive_ref_pic_marking_mode_flag est égal à 1.

Les commandes memory_management_control_operation avec des valeurs de 1 à 6 sont traitées dans l'ordre dans lequel elles surviennent dans le flux binaire après que l'image en cours a été décodée. Pour chacune de ces commandes d'opération memory_management_control_operation, un des processus spécifiés aux § 8.2.5.4.1 à 8.2.5.4.5 est invoqué en fonction de la valeur de memory_management_control_operation. La commande memory_management_control_operation avec la valeur 0 spécifie la fin des commandes memory_management_control_operation.

Les opérations de commande de gestion de mémoire sont appliquées comme suit aux images.

– Si field_pic_flag est égal à 0, les commandes memory_management_control_operation sont appliquées aux trames ou paires de sous-trames de référence complémentaires spécifiées.

– Sinon (si field_pic_flag est égal à 1), les commandes memory_management_control_operation sont appliquées aux sous-trames de référence individuelles spécifiées.

8.2.5.4.1 Processus de marquage d'image de référence à court terme "inutilisée comme référence"

Ce processus est invoqué lorsque memory_management_control_operation est égal à 1.

Soit picNumX spécifié par:

$$picNumX = CurrPicNum - (difference\_of\_pic\_nums\_minus1 + 1) \quad (8\text{-}40)$$

En fonction de field_pic_flag la valeur de picNumX est utilisée afin de marquer une image de référence à court terme comme étant "inutilisée comme référence" comme suit.

– Si field_pic_flag est égal à 0, la trame de référence à court terme ou la paire de sous-trames de référence complémentaires à court terme spécifiée par picNumX et ses deux sous-trames sont marquées "inutilisée comme référence".

– Sinon (si field_pic_flag est égal à 1), la sous-trame de référence à court terme spécifiée par picNumX est marquée comme étant "inutilisée comme référence". Lorsque cette sous-trame de référence fait partie d'une trame de référence ou d'une paire de sous-trames de référence complémentaires, la trame ou paire de sous-trames complémentaires est aussi marquée "inutilisée comme référence", mais le marquage de l'autre sous-trame n'est pas changé.

8.2.5.4.2 Processus de marquage d'une image à long terme comme étant "inutilisée comme référence"

Ce processus est invoqué lorsque memory_management_control_operation est égal à 2.

En fonction de field_pic_flag la valeur de LongTermPicNum est utilisée afin de marquer une image de référence à long terme comme étant "inutilisée comme référence" comme suit.

– Si field_pic_flag est égal à 0, la trame de référence à long terme ou la paire de sous-trames de référence complémentaires à long terme ayant LongTermPicNum égal à long_term_pic_num et ses deux sous-trames sont marquées "inutilisée comme référence".

– Sinon (si field_pic_flag est égal à 1), la sous-trame de référence à long terme spécifiée par LongTermPicNum égal à long_term_pic_num est marquée "inutilisée comme référence". Lorsque cette sous-trame de référence fait partie d'une trame de référence ou d'une paire de sous-trames de référence complémentaires, la trame ou paire de sous-trames complémentaires est aussi marquée "inutilisée comme référence", mais le marquage de l'autre sous-trame n'est pas changé.

8.2.5.4.3 Processus d'allocation d'un LongTermFrameIdx à une image de référence à court terme

Ce processus est invoqué lorsque memory_management_control_operation est égal à 3.

Etant donné l'élément syntaxique difference_of_pic_nums_minus1, la variable picNumX est obtenue comme spécifié au § 8.2.5.4.1. picNumX doit faire référence à une trame ou à une paire de sous-trames de référence complémentaires ou à une sous-trame de référence non appariée et marquée "utilisée comme référence à court terme" et non marquée "non existante".

Lorsque LongTermFrameIdx égal à long_term_frame_idx est déjà alloué à une trame de référence à long terme ou à une paire de sous-trames de référence complémentaires à long terme, cette trame ou paire de sous-trames complémentaires et ses deux sous-trames sont marquées "inutilisée comme référence". Lorsque LongTermFrameIdx est déjà alloué à une sous-trame de référence non appariée et que cette sous-trame n'est pas la sous-trame complémentaire de l'image spécifiée par picNumX, cette sous-trame est marquée "inutilisée comme référence".

En fonction de field_pic_flag la valeur de LongTermFrameIdx est utilisée afin de marquer une image de "utilisée comme référence à court terme" à "utilisée comme référence à long terme" comme suit.

– Si field_pic_flag est égal à 0, le marquage de la trame de référence à court terme ou de la paire de sous-trames de référence complémentaires à court terme spécifiée par picNumX et ses deux sous-trames est changé de "utilisée comme référence à court terme" à "utilisée comme référence à long terme" et le LongTermFrameIdx est alloué égal à long_term_frame_idx.

– Sinon (si field_pic_flag est égal à 1), le marquage de la sous-trame de référence à court terme spécifiée par picNumX est changé de "utilisée comme référence à court terme" en "utilisée comme référence à long terme" et le

LongTermFrameIdx est alloué égal à long_term_frame_idx. Lorsque la sous-trame fait partie d'une trame de référence ou d'une paire de sous-trames de référence complémentaires, et que l'autre sous-trame de la même trame de référence ou paire de sous-trames de référence complémentaires est également marquée "utilisée comme référence à long terme", la trame de référence ou paire de sous-trames de référence complémentaires est également marquée "utilisée comme référence à long terme" et attribuée à LongTermFrameIdx égal à long_term_frame_idx.

8.2.5.4.4 Processus de décodage pour MaxLongTermFrameIdx

Ce processus est invoqué lorsque memory_management_control_operation est égal à 4.

Toutes les images pour lesquelles LongTermFrameIdx est supérieur à max_long_term_frame_idx_plus1 – 1 et qui sont marquées "utilisée comme référence à long terme" sont marquées "inutilisée comme référence".

La variable MaxLongTermFrameIdx est déduite comme suit.

– Si max_long_term_frame_idx_plus1 est égal à 0, MaxLongTermFrameIdx est mis égal à "pas d'indice de trame à long terme".

– Sinon (si max_long_term_frame_idx_plus1 est supérieur à 0), MaxLongTermFrameIdx est mis égal à max_long_term_frame_idx_plus1 – 1.

NOTE – La commande memory_management_control_operation égale à 4 peut être utilisée afin de marquer des images de référence à long terme "inutilisée comme référence". La fréquence de transmission max_long_term_frame_idx_plus1 n'est pas spécifiée par la présente Recommandation | Norme internationale. Cependant, le codeur ne devrait pas envoyer de commande memory_management_control_operation égale à 4 à réception d'un message d'erreur, tel qu'un message de demande de rafraîchissement interne.

8.2.5.4.4.1 Processus de marquage de toutes les images de référence "inutilisée comme référence" et mise de MaxLongTermFrameIdx à "pas d'indice de trame à long terme"

Ce processus est invoqué lorsque memory_management_control_operation est égal à 5.

Toutes les images de référence sont marquées "inutilisée comme référence" et la variable MaxLongTermFrameIdx est mise égale à "pas d'indice de trame à long terme".

8.2.5.4.5 Processus afin d'allouer un indice de trame à long terme à l'image en cours

Ce processus est invoqué lorsque memory_management_control_operation est égal à 6.

Lorsqu'une variable LongTermFrameIdx égale à long_term_frame_idx est déjà allouée à une trame de référence à long terme ou à une paire de sous-trames de référence complémentaires à long terme, cette trame ou paire de sous-trames complémentaires et ses deux sous-trames sont marquées "inutilisée comme référence". Lorsque LongTermFrameIdx est déjà alloué à une sous-trame de référence non appariée et que cette sous-trame n'est pas la sous-trame complémentaire de l'image en cours, cette sous-trame est marquée "inutilisée comme référence".

L'image en cours est marquée "utilisée comme référence à long terme" et LongTermFrameIdx lui est alloué égal à long_term_frame_idx.

Lorsque field_pic_flag est égal à 0, ses deux sous-trames sont aussi marquées "utilisée comme référence à long terme" et LongTermFrameIdx leur est alloué égal à long_term_frame_idx.

Lorsque field_pic_flag est égal à 1 et que l'image en cours est la seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames de référence complémentaires, et lorsque la première sous-trame de la paire de sous-trames de référence complémentaires est également marquée actuellement "utilisée comme référence à long terme", la paire de sous-trames de référence complémentaires est aussi marquée "utilisée comme référence à long terme" et LongTermFrameIdx lui est alloué égal à long_term_frame_idx.

Après le marquage de l'image de référence décodée en cours, le nombre total de trames avec au moins une sous-trame marquée "utilisée comme référence", plus le nombre de paires de sous-trames complémentaires avec au moins une sous-trame marquée "utilisée comme référence", plus le nombre de sous-trames non appariées marquées "utilisée comme référence", ne doit pas être supérieur à Max(num_ref_frames, 1).

NOTE – Dans certaines circonstances, l'énoncé ci-dessus peut imposer une contrainte relative à l'ordre dans lequel un élément syntaxique memory_management_control_operation égal à 6 peut figurer dans la syntaxe de marquage d'image de référence décodée par rapport à un élément syntaxique memory_management_control_operation égal à 1, 2 ou 4.

8.3 Processus d'intraprédiction

Ce processus est invoqué pour les types de macrobloc I et SI.

Les entrées dans ce processus sont les échantillons construits avant le processus de filtre de démontage de bloc et, pour les modes de prédiction Intra_NxN (où NxN est égal à 4x4 ou 8x8), les valeurs de IntraNxNPredMode, extraites des macroblocs voisins.

Les résultats de ce processus sont spécifiés comme suit.

– Si le mode de prédiction de macrobloc est Intra_4x4 ou Intra_8x8, les sorties sont les échantillons luma construits avant le processus de filtre de démontage de bloc et (lorsque chroma_format_idc n'est pas égal à 0) les échantillons chroma de prédiction du macrobloc $pred_C$, où C est égal à Cb et Cr.

– Sinon, si mb_type n'est pas égal à I_PCM, les sorties sont les échantillons luma de prédiction du macrobloc $pred_L$ et (lorsque chroma_format_idc n'est pas égal à 0) les échantillons chroma de prédiction du macrobloc $pred_C$, où C est égal à Cb et Cr.

– Sinon (si mb_type est égal à I_PCM), les sorties sont les échantillons luma et (lorsque chroma_format_idc n'est pas égal à 0) échantillons chroma construits avant le processus de filtre de démontage de bloc.

La variable MvCnt est mise égale à 0.En fonction de la valeur de mb_type on applique ce qui suit.

– Si mb_type est égal à I_PCM, le processus spécifié au § 8.3.5 est invoqué.

– Sinon (si mb_type n'est pas égal à I_PCM), on applique ce qui suit.

  – Les processus de décodage pour les modes d'intraprédiction sont décrits comme suit pour la composante luma.
    – Si le mode de prédiction de macrobloc est égal à Intra_4x4, la spécification du § 8.3.1 s'applique.
    – Sinon, si le mode de prédiction de macrobloc est égal à Intra_8x8, la spécification du § 8.3.2 s'applique.
    – Sinon, (si le mode de prédiction de macrobloc est égal à Intra_16x16), la spécification du § 8.3.3 s'applique.
  – Les processus de décodage pour les modes d'intraprédiction pour les composantes chroma sont décrits au § 8.3.4. Ce processus n'est invoqué que lors que chroma_format_idc n'est pas égal à 0 (format monochrome).

Les échantillons utilisés dans le processus d'intraprédiction sont des valeurs d'échantillon antérieures à l'altération par toute opération de filtrage de démontage de blocs.

8.3.1 Processus de prédiction Intra_4x4 pour les échantillons luma

Ce processus est invoqué lorsque le mode de prédiction de macrobloc est égal à Intra_4x4.

Les entrées dans ce processus sont les valeurs de Intra4x4PredMode (si disponible) ou Intra8x8PredMode (si disponible) de macroblocs ou paires de macroblocs voisins.

La composante luma d'un macrobloc consiste en 16 blocs d'échantillons luma 4x4. Ces blocs subissent un balayage inverse au moyen du processus de balayage inverse de bloc luma 4x4 comme spécifié au § 6.4.3.

Pour tous les blocs luma 4x4 de la composante luma d'un macrobloc avec luma4x4BlkIdx = 0..15, le processus de déduction pour le mode Intra4x4PredMode spécifié dans le § 8.3.1.1 est invoqué avec, en entrée, luma4x4BlkIdx ainsi que les modes Intra4x4PredMode et Intra8x8PredMode déjà calculés (dans l'ordre du décodage) pour les macroblocs adjacents et, en sortie, la variable Intra4x4PredMode[luma4x4BlkIdx].

Pour chaque bloc luma d'échantillons 4x4 indexé au moyen de luma4x4BlkIdx = 0..15,

1. Le processus de prédiction d'échantillon Intra_4x4 du § 8.3.1.2 est invoqué avec luma4x4BlkIdx et les échantillons construits antérieurement (dans l'ordre de décodage) au processus de filtrage de démontage de blocs à partir des blocs luma adjacents en entrée, le résultat étant les échantillons de prédiction luma Intra_4x4 $pred4x4_L[x, y]$ avec x, y = 0..3.

2. La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx à l'intérieur du macrobloc en cours est déduite en invoquant le processus de balayage de bloc luma 4x4 inverse décrit au § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO) et x, y = 0..3.

$$pred_L[xO + x, yO + y] = pred4x4_L[x, y] \quad (8\text{-}41)$$

3. Le processus de décodage du coefficient de transformée et le processus de construction d'image avant le processus de filtrage de démontage de blocs du § 8.5 sont invoqués avec $pred_L$ et luma4x4BlkIdx comme entrée et les échantillons S'_L construits pour le bloc luma 4x4 en cours comme résultat.

8.3.1.1 Processus de déduction pour Intra4x4PredMode

Les entrées dans ce processus sont l'indice du bloc luma 4x4 luma4x4BlkIdx et les matrices de variables Intra4x4PredMode (si disponible) et Intra8x8PredMode (si disponible) qui ont été déduites précédemment (dans l'ordre de décodage) pour les macroblocs adjacents.

Le résultat de ce processus est la variable Intra4x4PredMode[luma4x4BlkIdx].

Le Tableau 8-2 spécifie les valeurs pour Intra4x4PredMode[luma4x4BlkIdx] et les noms associés.

Tableau 8-2 – Spécification de Intra4x4PredMode[luma4x4BlkIdx] et des noms associés

Intra4x4PredMode[luma4x4BlkIdx]	Nom de Intra4x4PredMode[luma4x4BlkIdx]
0	Intra_4x4_Vertical (mode de prédiction)
1	Intra_4x4_Horizontal (mode de prédiction)
2	Intra_4x4_DC (mode de prédiction)
3	Intra_4x4_Diagonal_Down_Left (mode de prédiction)
4	Intra_4x4_Diagonal_Down_Right (mode de prédiction)
5	Intra_4x4_Vertical_Right (mode de prédiction)
6	Intra_4x4_Horizontal_Down (mode de prédiction)
7	Intra_4x4_Vertical_Left (mode de prédiction)
8	Intra_4x4_Horizontal_Up (mode de prédiction)

Intra4x4PredMode[luma4x4BlkIdx] étiqueté 0, 1, 3, 4, 5, 6, 7 et 8 représente les directions de prédiction comme illustré à la Figure 8-1.

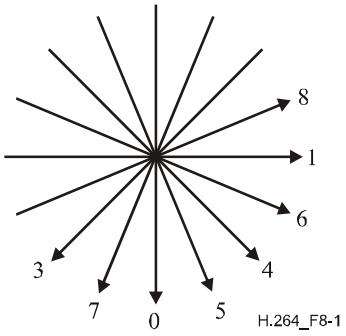


Figure 8-1 – Directions du mode de prédiction Intra_4x4 (pour information)

Intra4x4PredMode[luma4x4BlkIdx] est déduit comme suit.

– Le processus spécifié au § 6.4.8.3 est invoqué avec luma4x4BlkIdx donné en entrée et le résultat alloué à mbAddrA, luma4x4BlkIdxA, mbAddrB et luma4x4BlkIdxB.

– La variable dcPredModePredictedFlag est déduite comme suit.

  – Si une quelconque des conditions suivantes est vraie, dcPredModePredictedFlag est mis égal à 1

    – le macrobloc avec l'adresse mbAddrA n'est pas disponible

    – le macrobloc avec l'adresse mbAddrB n'est pas disponible

    – le macrobloc avec l'adresse mbAddrA est disponible et codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1

    – le macrobloc avec l'adresse mbAddrB est disponible et codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1

  – Sinon, dcPredModePredictedFlag est mis égal à 0.

– Pour N remplacé par A ou B, les variables intraMxMPredModeN sont déduites comme suit.

– Si dcPredModePredictedFlag est égal à 1 ou que le macrobloc avec l'adresse mbAddrN ne soit pas codé en mode de prédiction de macrobloc Intra_4x4 ou Intra_8x8, intraMxMPredModeN est mis égal à 2 (mode de prédiction Intra_4x4_DC).

– Sinon (si dcPredModePredictedFlag est égal à 0 (et que le macrobloc avec l'adresse mbAddrN soit codé en mode de prédiction de macrobloc Intra_4x4 ou que le macrobloc avec l'adresse mbAddrN soit codé en mode de prédiction de macrobloc Intra_8x8)), ce qui suit s'applique.

  – Si le macrobloc avec l'adresse mbAddrN est codé en mode de prédiction de macrobloc Intra_4x4, la variable intraMxMPredModeN est mise égale à Intra4x4PredMode[luma4x4BlkIdxN], où Intra4x4PredMode est la matrice de variables allouée au macrobloc mbAddrN.

  – Sinon (si le macrobloc ayant l'adresse mbAddrN est codé dans le mode Intra_8x8), intraMxMPredModeN est mis à Intra8x8PredMode[luma4x4BlkIdxN >> 2], où Intra8x8PredMode est la matrice de variables allouée au macrobloc mbAddrN.

– Intra4x4PredMode[luma4x4BlkIdx] est déduit par application de la procédure suivante:

```
predIntra4x4PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
si( prev_intra4x4_pred_mode_flag[ luma4x4BlkIdx ] )
    Intra4x4PredMode[ luma4x4BlkIdx ] = predIntra4x4PredMode
ou                                                                    (8-42)
    si( rem_intra4x4_pred_mode[ luma4x4BlkIdx ] < predIntra4x4PredMode )
        Intra4x4PredMode[ luma4x4BlkIdx ] = rem_intra4x4_pred_mode[ luma4x4BlkIdx ]
    ou
        Intra4x4PredMode[ luma4x4BlkIdx ] = rem_intra4x4_pred_mode[ luma4x4BlkIdx ] + 1
```

8.3.1.2 Prédiction d'échantillon Intra_4x4

Ce processus est invoqué pour chaque bloc luma 4x4 d'un macrobloc avec mode de prédiction égal à Intra_4x4 suivi par le processus de décodage de la transformée et le processus de construction d'image avant le démontage de bloc pour chaque bloc luma 4x4.

L'entrée dans ce processus est l'indice luma4x4BlkIdx d'un bloc luma 4x4.

Les résultats de ce processus sont les échantillons de prédiction $pred4x4_L[x, y]$, avec x, y = 0..3 pour le bloc luma 4x4 avec l'indice luma4x4BlkIdx.

La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx à l'intérieur du macrobloc en cours est déduite en invoquant le processus de balayage de bloc luma 4x4 inverse du § 6.4.3 avec luma4x4BlkIdx en entrée et le résultat étant alloué à (xO, yO).

Les 13 échantillons voisins p[x, y] qui sont des échantillons luma construits avant le processus de filtrage de démontage de blocs, avec x = –1, y = –1..3 et x = 0..7, y = –1, sont déduits comme suit.

– La localisation luma (xN, yN) est spécifié par

$$xN = xO + x \tag{8-43}$$

$$yN = yO + y \tag{8-44}$$

– Le processus de déduction pour les localisations du voisinage du § 6.4.9 est invoqué pour les localisations luma avec (xN, yN) en entrée et mbAddrN et (xW, yW) comme résultat.

– Chaque échantillon p[x, y] avec x = –1, y = –1..3 et x = 0..7, y = –1 est déduit comme suit.

  – Si une des conditions suivantes est vraie, l'échantillon p[x, y] est marqué "indisponible pour la prédiction Intra_4x4"

    – mbAddrN n'est pas disponible,

    – le macrobloc mbAddrN est codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1,

    – le macrobloc mbAddrN a mb_type égal à SI et constrained_intra_pred_flag est égal à 1 et le macrobloc en cours n'a pas mb_type égal à SI,

    – x est supérieur à 3 et luma4x4BlkIdx est égal à 3 ou 11.

– Sinon, l'échantillon p[x, y] est marqué "disponible pour la prédiction Intra_4x4" et l'échantillon luma à la localisation luma (xW, yW) à l'intérieur du macrobloc mbAddrN est alloué à p[x, y].

Lorsque les échantillons p[x, –1], avec x = 4..7, sont marqués "indisponible pour la prédiction Intra_4x4" et que l'échantillon p[3, –1] est marqué "disponible pour la prédiction Intra_4x4," la valeur d'échantillon de p[3, –1] est substituée aux valeurs d'échantillon p[x, –1], avec x = 4..7 et les échantillons p[x, –1], avec x = 4..7, sont marqués "disponible pour la prédiction Intra_4x4".

NOTE – Chaque bloc est supposé être construit dans une série matricielle d'images avant de décoder le bloc suivant.

En fonction de Intra4x4PredMode[luma4x4BlkIdx], un des modes de prédiction Intra_4x4 spécifiés aux § 8.3.1.2.1 à 8.3.1.2.9 est invoqué.

8.3.1.2.1 Spécification du mode de prédiction Intra_4x4_Vertical

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 0.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..3 sont marqués "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées par

$$pred4x4_L[x, y] = p[x, -1], \text{ avec } x, y = 0..3 \tag{8-45}$$

8.3.1.2.2 Spécification du mode de prédiction Intra_4x4_Horizontal

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 1.

Ce mode ne doit être utilisé que lorsque les échantillons p[–1, y], avec y = 0..3, sont marqués "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées par

$$pred4x4_L[x, y] = p[-1, y], \text{ avec } x,y = 0..3 \tag{8-46}$$

8.3.1.2.3 Spécification du mode de prédiction Intra_4x4_DC

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 2.

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées comme suit.

– Si tous les échantillons p[x, –1], avec x = 0..3 et p[–1, y], avec y = 0..3, sont marqués "disponible pour la prédiction Intra_4x4", les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées par

$$pred4x4_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 4) >> 3 \tag{8-47}$$

– Sinon, si des échantillons p[x, –1], avec x = 0..3 sont marqués "indisponible pour la prédiction Intra_4x4" et tous les échantillons p[–1, y], avec y = 0..3 sont marqués "disponible pour la prédiction Intra_4x4", les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées par

$$pred4x4_L[x, y] = (p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 2) >> 2 \tag{8-48}$$

– Sinon, si des échantillons p[–1, y], avec y = 0..3 sont marqués "indisponible pour la prédiction Intra_4x4" et tous les échantillons p[x, –1], avec x = 0..3 sont marqués "disponible pour la prédiction Intra_4x4", les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3 sont calculées par

$$pred4x4_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + 2) >> 2 \tag{8-49}$$

– Sinon (si certains échantillons p[x, –1], avec x = 0..3 et certains échantillons p[–1, y], avec y = 0..3, sont marqués "indisponible pour la prédiction Intra_4x4"), les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées par

$$pred4x4_L[x, y] = (1 << (BitDepth_Y - 1)) \tag{8-50}$$

NOTE – Un bloc luma 4x4 peut toujours être prédit au moyen de ce mode.

8.3.1.2.4 Spécification du mode de prédiction Intra_4x4_Diagonal_Down_Left

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 3.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..7 sont marqués "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées comme suit.

– Si x est égal à 3 et y est égal à 3

$$pred4x4_L[x, y] = (p[6, -1] + 3 * p[7, -1] + 2) >> 2 \quad (8\text{-}51)$$

– Sinon (si x n'est pas égal à 3 ou si y n'est pas égal à 3)

$$pred4x4_L[x, y] = (p[x + y, -1] + 2 * p[x + y + 1, -1] + p[x + y + 2, -1] + 2) >> 2 \quad (8\text{-}52)$$

8.3.1.2.5 Spécification du mode de prédiction Intra_4x4_Diagonal_Down_Right

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 4.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..3 et p[–1, y] avec y = –1..3 sont marqués "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées comme suit.

– Si x est supérieur à y,

$$pred4x4_L[x, y] = (p[x - y - 2, -1] + 2 * p[x - y - 1, -1] + p[x - y, -1] + 2) >> 2 \quad (8\text{-}53)$$

– Sinon, si x est inférieur à y,

$$pred4x4_L[x, y] = (p[-1, y - x - 2] + 2 * p[-1, y - x - 1] + p[-1, y - x] + 2) >> 2 \quad (8\text{-}54)$$

– Sinon (si x est égal à y),

$$pred4x4_L[x, y] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8\text{-}55)$$

8.3.1.2.6 Spécification du mode de prédiction Intra_4x4_Vertical_Right

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 5.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..3 et p[–1, y] avec y = –1..3 sont marqués "disponible pour la prédiction Intra_4x4".

Soit la variable zVR mise égale à 2 * x – y.

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3, sont calculées comme suit.

– Si zVR est égal à 0, 2, 4 ou 6,

$$pred4x4_L[x, y] = (p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 1) >> 1 \quad (8\text{-}56)$$

– Sinon, si zVR est égal à 1, 3 ou 5,

$$pred4x4_L[x, y] = (p[x - (y >> 1) - 2, -1] + 2 * p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 2) >> 2 \quad (8\text{-}57)$$

– Sinon, si zVR est égal à –1,

$$pred4x4_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8\text{-}58)$$

– Sinon (si zVR est égal à –2 ou –3),

$$pred4x4_L[x, y] = (p[-1, y - 1] + 2 * p[-1, y - 2] + p[-1, y - 3] + 2) >> 2 \quad (8\text{-}59)$$

8.3.1.2.7 Spécification du mode de prédiction Intra_4x4_Horizontal_Down

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 6.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..3 et p[–1, y] avec y = –1..3 sont marqués "disponible pour la prédiction Intra_4x4".

Soit la variable zHD mise égale à 2 * y – x.

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3 sont calculées comme suit.

– Si zHD est égal à 0, 2, 4 ou 6,

$$pred4x4_L[x, y] = (p[-1, y -(x >> 1) - 1] + p[-1, y - (x >> 1)] + 1) >> 1 \quad (8\text{-}60)$$

– Sinon, si zHD est égal à 1, 3 ou 5,

$$pred4x4_L[x, y] = (p[-1, y - (x >> 1) - 2] + 2 * p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 2) >> 2 \quad (8\text{-}61)$$

– Sinon, si zHD est égal à –1,

$$pred4x4_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8\text{-}62)$$

– Sinon (si zHD est égal à –2 ou –3),

$$pred4x4_L[x, y] = (p[x - 1, -1] + 2 * p[x - 2, -1] + p[x - 3, -1] + 2) >> 2 \quad (8\text{-}63)$$

8.3.1.2.8 Spécification du mode de prédiction Intra_4x4_Vertical_Left

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 7.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..7 sont marqués "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3 sont calculées comme suit.

– Si y est égal à 0 ou 2,

$$pred4x4_L[x, y] = (p[x + (y >> 1), -1] + p[x + (y >> 1) + 1, -1] + 1) >> 1 \quad (8\text{-}64)$$

– Sinon (si y est égal à 1 ou 3),

$$pred4x4_L[x, y] = (p[x + (y >> 1), -1] + 2 * p[x + (y >> 1) + 1, -1] + p[x + (y >> 1) + 2, -1] + 2) >> 2 \quad (8\text{-}65)$$

8.3.1.2.9 Spécification du mode de prédiction Intra_4x4_Horizontal_Up

Ce mode de prédiction Intra_4x4 est invoqué lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 8.

Ce mode ne doit être utilisé que lorsque les échantillons p[–1, y] avec y = 0..3 sont marqués "disponible pour la prédiction Intra_4x4".

Soit la variable zHU mise égale à x + 2 * y.

Les valeurs des échantillons de prédiction $pred4x4_L$[x, y], avec x, y = 0..3 sont calculées comme suit:

– Si zHU est égal à 0, 2 ou 4

$$pred4x4_L[x, y] = (p[-1, y + (x >> 1)] + p[-1, y + (x >> 1) + 1] + 1) >> 1 \quad (8\text{-}66)$$

– Sinon, si zHU est égal à 1 ou 3

$$pred4x4_L[x, y] = (p[-1, y + (x >> 1)] + 2 * p[-1, y + (x >> 1) + 1] + p[-1, y + (x >> 1) + 2] + 2) >> 2 \quad (8\text{-}67)$$

– Sinon, si zHU est égal à 5,

$$pred4x4_L[x, y] = (p[-1, 2] + 3 * p[-1, 3] + 2) >> 2 \quad (8\text{-}68)$$

– Sinon (si zHU est supérieur à 5),

$$pred4x4_L[x, y] = p[-1, 3] \quad (8\text{-}69)$$

8.3.2 Processus de prédiction Intra_8x8 pour les échantillons luma

Ce processus est invoqué lorsque le mode de prédiction de macrobloc est égal à Intra_8x8.

Les entrées dans ce processus sont les valeurs de Intra4x4PredMode (si disponible) ou Intra8x8PredMode (si disponible) des macroblocs ou paires de macroblocs voisins.

Les résultats de ce processus sont les séries matricielles d'échantillons luma 8x8 qui font partie de la série matricielle d'échantillons luma 16x16 d'échantillons de prédiction du macrobloc $pred_L$.

La composante luma d'un macrobloc se compose de 4 blocs d'échantillons luma 8x8. Ces blocs sont balayés en sens inverse au moyen du processus de balayage inverse de blocs luma 8x8 comme spécifié dans le § 6.4.4.

Pour tous les blocs luma 8x8 de la composante luma d'un macrobloc avec luma8x8BlkIdx = 0..3, le processus de calcul pour Intra8x8PredMode, spécifié dans le § 8.3.2.1, est invoqué avec, en entrée, luma8x8BlkIdx ainsi que Intra4x4PredMode et Intra8x8PredMode qui ont déjà été calculés (dans l'ordre du décodage) pour les macroblocs adjacents et avec, en sortie, la variable Intra8x8PredMode[luma8x8BlkIdx].

Pour chaque bloc d'échantillons luma 8x8 indexés au moyen de luma8x8BlkIdx = 0..3, ce qui suit est applicable:

– Le processus de prédiction d'échantillon Intra_8x8 décrit dans le § 8.3.2.2 est invoqué avec en entrée luma8x8BlkIdx et les échantillons construits avant (dans l'ordre du décodage) le processus de filtre de démontage de blocs luma adjacents et avec, en sortie, les échantillons luma du mode de prédiction Intra_8x8 du macrobloc $pred8x8_L[x, y]$ avec x, y = 0..7.

– La position de l'échantillon supérieur gauche d'un bloc luma 8x8 ayant l'indice luma8x8BlkIdx à l'intérieur du macrobloc actuel est calculée par invocation du processus de balayage inverse de blocs luma 8x8 dans le § 6.4.4 avec luma8x8BlkIdx comme entrée et la sortie étant attribuée à (xO, yO) et x, y = 0..7.

$$pred_L[xO + x, yO + y] = pred8x8_L[x, y] \quad (8\text{-}70)$$

– Le processus de décodage des coefficients de transformée et le processus de construction d'image avant processus de filtre de démontage de bloc dans le § 8.5 est invoqué avec $pred_L$ et luma8x8BlkIdx comme entrée et les échantillons construits pour le bloc actuel d'échantillons luma 8x8 S'_L comme sortie.

8.3.2.1 Processus de déduction pour Intra8x8PredMode

Les entrées dans ce processus sont l'indice de bloc luma 8x8 luma8x8BlkIdx et les matrices de variables Intra4x4PredMode (si disponible) et Intra8x8PredMode (si disponible) qui ont déjà été (dans l'ordre du décodage) calculées pour les macroblocs adjacents.

La sortie de ce processus est la variable Intra8x8PredMode[luma8x8BlkIdx].

Le Tableau 8-3 spécifie les valeurs pour Intra8x8PredMode[luma8x8BlkIdx] et les noms mnémoniques associés.

Tableau 8-3 – Spécification de Intra8x8PredMode[luma8x8BlkIdx] et des noms associés

Intra8x8PredMode[luma8x8BlkIdx]	Nom de Intra8x8PredMode[luma8x8BlkIdx]
0	Intra_8x8_Vertical (mode de prédiction)
1	Intra_8x8_Horizontal (mode de prédiction)
2	Intra_8x8_DC (mode de prédiction)
3	Intra_8x8_Diagonal_Down_Left (mode de prédiction)
4	Intra_8x8_Diagonal_Down_Right (mode de prédiction)
5	Intra_8x8_Vertical_Right (mode de prédiction)
6	Intra_8x8_Horizontal_Down (mode de prédiction)
7	Intra_8x8_Vertical_Left (mode de prédiction)
8	Intra_8x8_Horizontal_Up (mode de prédiction)

L'indice Intra8x8PredMode[luma8x8BlkIdx] est calculé comme suit:

– Le processus spécifié dans le § 6.4.8.2 est invoqué avec luma8x8BlkIdx donné en entrée et la sortie est attribuée à mbAddrA, luma8x8BlkIdxA, mbAddrB, et luma8x8BlkIdxB.

– La variable dcPredModePredictedFlag est calculée comme suit:

  – Si la totalité des conditions suivantes sont vraies, dcPredModePredictedFlag est mis à 1

    – le macrobloc ayant l'adresse mbAddrA est indisponible

    – le macrobloc ayant l'adresse mbAddrB est indisponible

    – le macrobloc ayant l'adresse mbAddrA est disponible et codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1

    – le macrobloc ayant l'adresse mbAddrB est disponible et codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1

  – Sinon, dcPredModePredictedFlag est mis à 0.

– Pour N remplacé soit par A soit par B, les variables intraMxMPredModeN sont calculées comme suit:

  – Si dcPredModePredictedFlag est égal à 1 (ou si le macrobloc ayant l'adresse mbAddrN n'est pas codé dans le mode de prédiction de macrobloc Intra_4x4 et si le macrobloc ayant l'adresse mbAddrN n'est pas codé dans le mode de prédiction de macrobloc Intra_8x8), intraMxMPredModeN est mis à 2 (mode de prédiction Intra_8x8_DC).

  – Sinon (si dcPredModePredictedFlag est égal à 0 (et si le macrobloc ayant l'adresse mbAddrN est codé dans le mode de prédiction de macrobloc Intra_4x4 ou si le macrobloc ayant l'adresse mbAddrN est codé dans le mode de prédiction de macrobloc Intra_8x8)), ce qui suit est applicable:

    – Si le macrobloc ayant l'adresse mbAddrN est codé dans le mode Intra_8x8, intraMxMPredModeN est mis à Intra8x8PredMode[luma8x8BlkIdxN], où Intra8x8PredMode est la matrice de variables attribuée au macrobloc mbAddrN.

    – Sinon (si le macrobloc ayant l'adresse mbAddrN est codé dans le mode Intra_4x4), intraMxMPredModeN est calculé par la procédure suivante, où Intra4x4PredMode est la matrice de variables attribuée au macrobloc mbAddrN.

      intraMxMPredModeN = Intra4x4PredMode[luma8x8BlkIdxN * 4 + n] (8-71)

      où la variable n est calculée comme suit:

      – Si N est égal à A, selon la variable MbaffFrameFlag, la variable luma8x8BlkIdx, le macrobloc actuel, et le macrobloc mbAddrN, ce qui suit est applicable:

        – Si MbaffFrameFlag est égal à 1, si le macrobloc actuel est un macrobloc codé en trames, si le macrobloc mbAddrN est un macrobloc codé en sous-trames, et si luma8x8BlkIdx est égal à 2, n est mis à 3.

        – Sinon (si MbaffFrameFlag est égal à 0 ou si le macrobloc actuel est un macrobloc codé en sous-trames ou si le macrobloc mbAddrN est un macrobloc codé en trames ou si luma8x8BlkIdx n'est pas égal à 2), n est mis à 1.

      – Sinon (si N est égal à B), n est mis à 2.

– Finalement, en présence de MxMPredModeA et intraMxMPredModeB, la variable Intra8x8PredMode[luma8x8BlkIdx] est calculée par application de la procédure suivante.

```
predIntra8x8PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
si( prev_intra8x8_pred_mode_flag[ luma8x8BlkIdx ] )
    Intra8x8PredMode[ luma8x8BlkIdx ] = predIntra8x8PredMode
sinon                                                                    (8-72)
    si( rem_intra8x8_pred_mode[ luma8x8BlkIdx ] < predIntra8x8PredMode )
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ]
    sinon
        Intra8x8PredMode[ luma8x8BlkIdx ] = rem_intra8x8_pred_mode[ luma8x8BlkIdx ] + 1
```

8.3.2.2 Prédiction d'échantillon Intra_8x8

Ce processus est invoqué pour chaque bloc luma 8x8 d'un macrobloc avec mode de prédiction égal à Intra_8x8 suivi par le processus de décodage de transformée et par le processus de construction d'image avant démontage de bloc pour chaque bloc luma 8x8.

L'entrée dans ce processus est l'indice d'un bloc luma 8x8: luma8x8BlkIdx.

La sortie de ce processus est constituée par les échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 pour le bloc luma 8x8 ayant l'indice luma8x8BlkIdx.

La position de l'échantillon supérieur gauche d'un bloc luma 8x8 ayant l'indice luma8x8BlkIdx à l'intérieur du macrobloc actuel est calculée par invocation du processus de balayage inverse de blocs luma 8x8 dans le § 6.4.4 avec luma8x8BlkIdx comme entrée et la sortie étant attribuée à (xO, yO).

Les 25 échantillons voisins p[x, y] qui sont les échantillons luma construits avant le processus de filtre de démontage de bloc, avec x = -1, y = -1..7 et x = 0..15, y = -1, sont calculés comme suit:

– La localisation d'échantillon luma (xN, yN) est spécifiée par:

$$xN = xO + x \tag{8-73}$$

$$yN = yO + y \tag{8-74}$$

– Le processus de calcul pour les localisations voisines dans le § 6.4.9 est invoqué pour localisation d'échantillons luma avec (xN, yN) en entrée et mbAddrN et (xW, yW) en sortie.

– Chaque échantillon p[x, y] avec x = -1, y = -1..7 et x = 0..15, y = -1 est calculé comme suit:

  – Si toutes les conditions suivantes sont vraies, l'échantillon p[x, y] est marqué "indisponible pour prédiction Intra_8x8"

    – mbAddrN est indisponible,

    – le macrobloc mbAddrN est codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1,

  – Sinon, l'échantillon p[x, y] est marqué "disponible pour prédiction Intra_8x8" et l'échantillon luma à la localisation d'échantillon luma (xW, yW) à l'intérieur du macrobloc mbAddrN est attribuée à p[x, y].

Lorsque des échantillons p[x, -1], avec x = 8..15 sont marqués "indisponible pour prédiction Intra_8x8," et lorsque l'échantillon p[7, -1] est marqué "disponible pour prédiction Intra_8x8," la valeur d'échantillon de la position p[7, -1] est substituée aux valeurs d'échantillon p[x, -1], avec x = 8..15 et les échantillons p[x, -1], avec x = 8..15 sont marqués "disponible pour prédiction Intra_8x8".

NOTE – Chaque bloc est censé être construit dans une série matricielle d'images avant décodage du bloc suivant.

Le processus de filtrage d'échantillon de référence pour prédiction d'échantillons Intra_8x8 dans le § 8.3.2.2.1 est invoqué avec les échantillons p[x, y] avec x = -1, y = -1..7 et x = 0..15, y = -1 (si disponible) en entrée et p'[x, y] avec x = -1, y = -1..7 et x = 0..15, y = -1 en sortie.

Selon Intra8x8PredMode[luma8x8BlkIdx], un seul des modes de prédiction Intra_8x8 spécifiés dans les § 8.3.2.2.2 à 8.3.2.2.10 est invoqué.

8.3.2.2.1 Processus de filtrage d'échantillon de référence pour prédiction d'échantillon Intra_8x8

Les entrées dans ce processus sont les échantillons de référence p[x, y] avec x = -1, y = -1..7 et x = 0..15, y = -1 (si disponible) pour prédiction d'échantillon Intra_8x8.

Les résultats de ce processus sont les échantillons de référence filtrés p'[x, y] avec x = -1, y = -1..7 et x = 0..15, y = -1 pour prédiction d'échantillon Intra_8x8.

Lorsque tous les échantillons p[x, -1] avec x = 0..7 sont marqués "disponible pour prédiction Intra_8x8", ce qui suit est applicable:

– La valeur de p'[0, -1] est calculée comme suit:

  – Si p[-1, -1] est marqué "disponible pour prédiction Intra_8x8", p'[0, -1] est calculé par:

$$p'[0, -1] = (p[-1, -1] + 2 * p[0, -1] + p[1, -1] + 2) >> 2 \quad (8\text{-}75)$$

  – Sinon (si p[-1, -1] est marqué "indisponible pour prédiction Intra_8x8"), p'[0, -1] est calculé par:

$$p'[0, -1] = (3 * p[0, -1] + p[1, -1] + 2) >> 2 \quad (8\text{-}76)$$

– Les valeurs de p'[x, -1], avec x = 1..7 sont calculées par:

$$p'[x, -1] = (p[x-1, -1] + 2 * p[x, -1] + p[x+1, -1] + 2) >> 2 \quad (8\text{-}77)$$

Lorsque tous les échantillons p[x, -1] avec x = 7..15 sont marqués "disponible pour prédiction Intra_8x8", ce qui suit est applicable:

– Les valeurs de p'[x, -1], avec x = 8..14 sont calculées par:

$$p'[x, -1] = (p[x-1, -1] + 2 * p[x, -1] + p[x+1, -1] + 2) >> 2 \quad (8\text{-}78)$$

– La valeur de p'[15, -1] est calculée par:

$$p'[15, -1] = (p[14, -1] + 3 * p[15, -1] + 2) >> 2 \quad (8\text{-}79)$$

Lorsque l'échantillon p[-1, -1] est marqué "disponible pour prédiction Intra_8x8", la valeur de p'[-1, -1] est calculée comme suit:

– Si l'échantillon p[0, -1] est marqué "indisponible pour prédiction Intra_8x8" ou si l'échantillon p[-1, 0] est marqué "indisponible pour prédiction Intra_8x8", ce qui suit est applicable:

  – Si l'échantillon p[0, -1] est marqué "disponible pour prédiction Intra_8x8", p'[-1, -1] est calculé par:

$$p'[-1, -1] = (3 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8\text{-}80)$$

  – Sinon (si l'échantillon p[0, -1] est marqué "indisponible pour prédiction Intra_8x8" et si l'échantillon p[-1, 0] est marqué "disponible pour prédiction Intra_8x8"), p'[-1, -1] est calculé par

$$p'[-1, -1] = (3 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8\text{-}81)$$

– Sinon (si l'échantillon p[0, -1] est marqué "disponible pour prédiction Intra_8x8" et si l'échantillon p[-1, 0] est marqué "disponible pour prédiction Intra_8x8"), p'[-1, -1] est calculé par:

$$p'[-1, -1] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8\text{-}82)$$

Lorsque tous les échantillons p[-1, y] avec y = 0..7 sont marqués "disponible pour prédiction Intra_8x8", ce qui suit est applicable:

– La valeur de p'[-1, 0] est calculée comme suit:

  – Si p[-1, -1] est marqué "disponible pour prédiction Intra_8x8", p'[-1, 0] est calculé par:

$$p'[\,-1, 0\,] = (\,p[\,-1, -1\,] + 2 * p[\,-1, 0\,] + p[\,-1, 1\,] + 2\,) >> 2 \quad (8\text{-}83)$$

– Sinon (si p[-1, -1] est marqué "indisponible pour prédiction Intra_8x8"), p'[-1, 0] est calculé par:

$$p'[\,-1, 0\,] = (\,3 * p[\,-1, 0\,] + p[\,-1, 1\,] + 2\,) >> 2 \quad (8\text{-}84)$$

– Les valeurs de p'[-1, y], avec y = 1..6 sont calculées par:

$$p'[\,-1, y\,] = (\,p[\,-1, y\text{-}1\,] + 2 * p[\,-1, y\,] + p[\,-1, y\text{+}1\,] + 2\,) >> 2 \quad (8\text{-}85)$$

– La valeur de p'[-1, 7] est calculée par:

$$p'[\,-1,\ 7\,] = (\,p[\,-1, 6\,] + 3 * p[\,-1, 7\,] + 2\,) >> 2 \quad (8\text{-}86)$$

8.3.2.2.2 Spécification du mode de prédiction Intra_8x8_Vertical

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 0.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..7 sont marqués "disponible pour prédiction Intra_8x8".

Les valeurs des échantillons de prédiction $pred8x8_L[\,x, y\,]$, avec x, y = 0..7 sont calculées par:

$$pred8x8_L[\,x, y\,] = p'[\,x, -1\,], \text{ avec } x, y = 0..7 \quad (8\text{-}87)$$

8.3.2.2.3 Spécification du mode de prédiction Intra_8x8_Horizontal

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 1.

Ce mode ne doit être utilisé que lorsque les échantillons p[-1, y], avec y = 0..7 sont marqués "disponible pour prédiction Intra_8x8".

Les valeurs des échantillons de prédiction $pred8x8_L[\,x, y\,]$, avec x, y = 0..7 sont calculées par:

$$pred8x8_L[\,x, y\,] = p'[\,-1, y\,], \text{ avec } x, y = 0..7 \quad (8\text{-}88)$$

8.3.2.2.4 Spécification du mode de prédiction Intra_8x8_DC

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 2.

Les valeurs des échantillons de prédiction $pred8x8_L[\,x, y\,]$, avec x, y = 0..7 sont calculées comme suit:

– Si tous les échantillons p[x, -1], avec x = 0..7 et p[-1, y], avec y = 0..7 sont marqués "disponible pour prédiction Intra_8x8," les valeurs des échantillons de prédiction $pred8x8_L[\,x, y\,]$, avec x, y = 0..7 sont calculées par:

$$pred8x8_L[\,x, y\,] = \left(\sum_{x'=0}^{7} p'[x', -1] + \sum_{y'=0}^{7} p'[-1, y'] + 8\right) >> 4 \quad (8\text{-}89)$$

– Sinon, si tous les échantillons p[x, -1], avec x = 0..7 sont marqués "indisponible pour prédiction Intra_8x8" et si tous les échantillons p[-1, y], avec y = 0..7 sont marqués "disponible pour prédiction Intra_8x8", les valeurs des échantillons de prédiction $pred8x8_L[\,x, y\,]$, avec x, y = 0..7 sont calculées par:

$$pred8x8_L[\,x, y\,] = \left(\sum_{y'=0}^{7} p'[-1, y'] + 4\right) >> 3 \quad (8\text{-}90)$$

– Sinon, si tous les échantillons p[-1, y], avec y = 0..7 sont marqués "indisponible pour prédiction Intra_8x8" et si tous les échantillons p[x, -1], avec x = 0..7 sont marqués "disponible pour prédiction Intra_8x8", les valeurs des échantillons de prédiction $pred8x8_L[\,x, y\,]$, avec x, y = 0..7 sont calculées par:

$$pred8x8_L[\,x, y\,] = \left(\sum_{x'=0}^{7} p'[x', -1] + 4\right) >> 3 \quad (8\text{-}91)$$

– Sinon (si certains échantillons p[x, -1], avec x = 0..7 et si certains échantillons p[-1, y], avec y = 0..7 sont marqués "indisponible pour prédiction Intra_8x8"), les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées par:

$$pred8x8_L[x, y] = (1 << (BitDepth_Y - 1)) \quad (8\text{-}92)$$

NOTE – Un bloc luma 8x8 peut toujours être prédit au moyen de ce mode.

8.3.2.2.5 Spécification du mode de prédiction Intra_8x8_Diagonal_Down_Left

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 3.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..15 sont marqués "disponible pour prédiction Intra_8x8".

Les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées comme suit:

– Si x est égal à 7 et si y est égal à 7,

$$pred8x8_L[x, y] = (p'[14, -1] + 3 * p'[15, -1] + 2) >> 2 \quad (8\text{-}93)$$

– Sinon (si x n'est pas égal à 7 ou si y n'est pas égal à 7),

$$pred8x8_L[x, y] = (p'[x + y, -1] + 2 * p'[x + y + 1, -1] + p'[x + y + 2, -1] + 2) >> 2 \quad (8\text{-}94)$$

8.3.2.2.6 Spécification du mode de prédiction Intra_8x8_Diagonal_Down_Right

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 4.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..7 et p[-1, y] avec y = -1..7 sont marqués "disponible pour prédiction Intra_8x8".

Les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées comme suit:

– Si x est plus grand que y,

$$pred8x8_L[x, y] = (p'[x - y - 2, -1] + 2 * p'[x - y - 1, -1] + p'[x - y, -1] + 2) >> 2 \quad (8\text{-}95)$$

– Sinon si x est inférieur à y,

$$pred8x8_L[x, y] = (p'[-1, y - x - 2] + 2 * p'[-1, y - x - 1] + p'[-1, y - x] + 2) >> 2 \quad (8\text{-}96)$$

– Sinon (si x est égal à y),

$$pred8x8_L[x, y] = (p'[0, -1] + 2 * p'[-1, -1] + p'[-1, 0] + 2) >> 2 \quad (8\text{-}97)$$

8.3.2.2.7 Spécification du mode de prédiction Intra_8x8_Vertical_Right

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 5.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..7 et p[-1, y] avec y = -1..7 sont marqués "disponible pour prédiction Intra_8x8".

Soit la variable zVR mise égale à 2 * x – y.

Les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées comme suit:

– Si zVR est égal à 0, 2, 4, 6, 8, 10, 12, ou 14

$$pred8x8_L[x, y] = (p'[x - (y >> 1) - 1, -1] + p'[x - (y >> 1), -1] + 1) >> 1 \quad (8\text{-}98)$$

– Sinon, si zVR est égal à 1, 3, 5, 7, 9, 11, ou 13

$$pred8x8_L[x, y] = (p'[x - (y >> 1) - 2, -1] + 2 * p'[x - (y >> 1) - 1, -1] + p'[x - (y >> 1), -1] + 2) >> 2 \quad (8\text{-}99)$$

– Sinon, si zVR est égal à -1,

$$pred8x8_L[x, y] = (p'[-1, 0] + 2 * p'[-1, -1] + p'[0, -1] + 2) >> 2 \quad (8\text{-}100)$$

– Sinon (si zVR est égal à -2, -3, -4, -5, -6, ou -7),

$$pred8x8_L[x, y] = (p'[-1, y - 2*x - 1] + 2 * p'[-1, y - 2*x - 2] + p'[-1, y - 2*x - 3] + 2) >> 2 \quad (8\text{-}101)$$

8.3.2.2.8 Spécification de Intra_8x8_Horizontal_Down

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 6.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..7 et p[-1, y] avec y = -1..7 sont marqués "disponible pour prédiction Intra_8x8".

Soit la variable zHD mise égale à 2 * y – x.

Les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées comme suit:

– Si zHD est égal à 0, 2, 4, 6, 8, 10, 12, ou 14

$$pred8x8_L[x, y] = (p'[-1, y -(x >> 1) - 1] + p'[-1, y - (x >> 1)] + 1) >> 1 \quad (8\text{-}102)$$

– Sinon, si zHD est égal à 1, 3, 5, 7, 9, 11, ou 13

$$pred8x8_L[x, y] = (p'[-1, y - (x >> 1) - 2] + 2 * p'[-1, y - (x >> 1) - 1] + p'[-1, y - (x >> 1)] + 2) >> 2 \quad (8\text{-}103)$$

– Sinon, si zHD est égal à -1,

$$pred8x8_L[x, y] = (p'[-1, 0] + 2 * p'[-1, -1] + p'[0, -1] + 2) >> 2 \quad (8\text{-}104)$$

– Sinon (si zHD est égal à –2, -3, -4, -5, -6, -7),

$$pred8x8_L[x, y] = (p'[x - 2*y - 1, -1] + 2 * p'[x - 2*y - 2, -1] + p'[x - 2*y - 3, -1] + 2) >> 2 \quad (8\text{-}105)$$

8.3.2.2.9 Spécification du mode de prédiction Intra_8x8_Vertical_Left

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 7.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..15 sont marqués "disponible pour prédiction Intra_8x8".

Les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées comme suit:

– Si y est égal à 0, 2, 4 ou 6

$$pred8x8_L[x, y] = (p'[x + (y >> 1), -1] + p'[x + (y >> 1) + 1, -1] + 1) >> 1 \quad (8\text{-}106)$$

– Sinon (si y est égal à 1, 3, 5, 7),

$$pred8x8_L[x, y] = (p'[x + (y >> 1), -1] + 2 * p'[x + (y >> 1) + 1, -1] + p'[x + (y >> 1) + 2, -1] + 2) >>2 \quad (8\text{-}107)$$

8.3.2.2.10 Spécification du mode de prédiction Intra_8x8_Horizontal_Up

Ce mode de prédiction Intra_8x8 est invoqué lorsque Intra8x8PredMode[luma8x8BlkIdx] est égal à 8.

Ce mode ne doit être utilisé que lorsque les échantillons p[-1, y] avec y = 0..7 sont marqués "disponible pour prédiction Intra_8x8".

Soit la variable zHU mise égale à x + 2 * y.

Les valeurs des échantillons de prédiction $pred8x8_L[x, y]$, avec x, y = 0..7 sont calculées comme suit:

– Si zHU est égal à 0, 2, 4, 6, 8, 10, ou 12

$$pred8x8_L[x, y] = (p'[-1, y + (x >> 1)] + p'[-1, y + (x >> 1) + 1] + 1) >> 1 \quad (8\text{-}108)$$

– Sinon, si zHU est égal à 1, 3, 5, 7, 9, ou 11

$$pred8x8_L[x, y] = (p'[-1, y + (x >> 1)] + 2 * p'[-1, y + (x >> 1) + 1] + p'[-1, y + (x >> 1) + 2] + 2) >>2 \quad (8\text{-}109)$$

– Sinon, si zHU est égal à 13,

$$pred8x8_L[x, y] = (p'[-1, 6] + 3 * p'[-1, 7] + 2) >> 2 \quad (8\text{-}110)$$

– Sinon (si zHU est plus grand que 13),

$$pred8x8_L[x, y] = p'[-1, 7] \quad (8\text{-}111)$$

8.3.3 Processus de prédiction Intra_16x16 pour échantillons luma

Ce processus est invoqué lorsque le mode de prédiction de macrobloc est égal à Intra_16x16. Il spécifie comment sont déduits les échantillons luma d'intraprédiction pour le macrobloc en cours.

Les résultats de ce processus sont les échantillons luma d'intraprédiction pour le macrobloc en cours $pred_L[x, y]$.

Les 33 échantillons p[x, y] du voisinage qui sont des échantillons luma construits avant le processus de filtrage de démontage de blocs, avec x = –1, y = –1..15 et avec x = 0..15, y = –1, sont déduits comme suit.

– Le processus de déduction pour les localisations du voisinage du § 6.4.9 est invoqué pour les localisations luma avec (x, y) alloué à (xN, yN) comme entrée et mbAddrN et (xW, yW) comme résultat.

– Chaque échantillon p[x, y] avec x = –1, y = –1..15 et avec x = 0..15, y = –1 est déduit comme suit.

  – Si l'une des conditions suivantes est vraie, l'échantillon p[x, y] est marqué "indisponible pour la prédiction Intra_16x16":

    – mbAddrN n'est pas disponible;

    – le macrobloc mbAddrN est codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1;

    – le macrobloc mbAddrN a mb_type égal à SI et constrained_intra_pred_flag est égal à 1.

  – Sinon, l'échantillon p[x, y] est marqué "disponible pour la prédiction Intra_16x16" et l'échantillon luma à la localisation luma (xW, yW) à l'intérieur du macrobloc mbAddrN est mis à p[x, y].

Soit $pred_L[x, y]$ avec x, y = 0..15 afin de noter les échantillons de prédiction pour les échantillons de bloc luma 16x16.

Les modes de prédiction Intra_16x16 sont spécifiés au Tableau 8-4.

Tableau 8-4 – Spécification de Intra16x16PredMode et des noms associés

Intra16x16PredMode	Nom de Intra16x16PredMode
0	Intra_16x16_Vertical (mode de prédiction)
1	Intra_16x16_Horizontal (mode de prédiction)
2	Intra_16x16_DC (mode de prédiction)
3	Intra_16x16_Plane (mode de prédiction)

En fonction de Intra16x16PredMode, un des modes de prédiction Intra_16x16 spécifiés aux § 8.3.3.1 à 8.3.3.4 est invoqué.

8.3.3.1 Spécification du mode de prédiction Intra_16x16_Vertical

Ce mode de prédiction Intra_16x16 ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = 0..15 sont marqués "disponible pour la prédiction Intra_16x16".

$$pred_L[x, y] = p[x, -1], \text{ avec } y = 0..15 \quad (8\text{-}112)$$

8.3.3.2 Spécification du mode de prédiction Intra_16x16_Horizontal

Ce mode de prédiction Intra_16x16 ne doit être utilisé que lorsque les échantillons p[–1, y] avec y = 0..15 sont marqués "disponible pour la prédiction Intra_16x16".

$$pred_L[x, y] = p[-1, y], \text{ avec } x, y = 0..15 \quad (8\text{-}113)$$

8.3.3.3 Spécification du mode de prédiction Intra_16x16_DC

Ce mode de prédiction Intra_16x16 fonctionne comme suit, selon que les échantillons du voisinage sont marqués "disponible pour la prédiction Intra_16x16".

– Si tous les échantillons p[x, –1], avec x = 0..15 et p[–1, y], avec y = 0..15 du voisinage sont marqués "disponible pour la prédiction Intra_16x16", la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$pred_L[x, y] = (\sum_{x'=0}^{15} p[x',-1] + \sum_{y'=0}^{15} p[-1,y'] + 16) >> 5, \text{ avec } x, y = 0..15 \quad (8\text{-}114)$$

– Sinon, si un ou plusieurs des échantillons p[x, –1] du voisinage, avec x = 0..15 sont marqués "indisponible pour la prédiction Intra_16x16" et si tous les échantillons p[–1, y] du voisinage, avec y = 0..15 sont marqués "disponible pour la prédiction Intra_16x16", la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$pred_L[x, y] = (\sum_{y'=0}^{15} p[-1,y'] + 8) >> 4, \text{ avec } x, y = 0..15 \quad (8\text{-}115)$$

– Sinon, si un ou plusieurs des échantillons p[–1, y] du voisinage, avec y = 0..15 sont marqués "indisponible pour la prédiction Intra_16x16 et que tous les échantillons p[x, –1] du voisinage, avec x = 0..15 sont marqués "disponible pour la prédiction Intra_16x16", la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$pred_L[x, y] = (\sum_{x'=0}^{15} p[x',-1] + 8) >> 4, \text{ avec } x, y = 0..15 \quad (8\text{-}116)$$

– Sinon (si certains des échantillons p[x, –1] du voisinage, avec x = 0..15 et certains des échantillons p[–1, y] du voisinage, avec y = 0..15 sont marqués "indisponible pour la prédiction Intra_16x16"), la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$pred_L[x, y] = (1 << (BitDepth_Y - 1)), \text{ avec } x, y = 0..15 \quad (8\text{-}117)$$

8.3.3.4 Spécification du mode de prédiction Intra_16x16_Plane

Ce mode de prédiction Intra_16x16 ne doit être utilisé que lorsque les échantillons p[x, –1] avec x = –1..15 et p[–1, y] avec y = 0..15 sont marqués "disponible pour la prédiction Intra_16x16".

$$pred_L[x, y] = Clip1_Y((a + b * (x - 7) + c * (y - 7) + 16) >> 5), \text{ avec } x, y = 0..15, \quad (8\text{-}118)$$

où:

$$a = 16 * (p[-1, 15] + p[15, -1]) \quad (8\text{-}119)$$

$$b = (5 * H + 32) >> 6 \quad (8\text{-}120)$$

$$c = (5 * V + 32) >> 6 \quad (8\text{-}121)$$

et H et V sont spécifiés aux équations 8-122 et 8-123.

$$H = \sum_{x'=0}^{7} (x'+1) * (p[8 + x', -1] - p[6 - x', -1]) \quad (8\text{-}122)$$

$$V = \sum_{y'=0}^{7} (y'+1) * (p[-1, 8+y'] - p[-1, 6-y']) \quad (8\text{-}123)$$

8.3.4 Processus d'intraprédiction pour les échantillons chroma

Ce processus est invoqué pour les types de macroblocs I et SI. Il spécifie comment sont déduits les échantillons chroma d'intraprédiction pour le macrobloc en cours.

Les résultats de ce processus sont les échantillons chroma d'intraprédiction pour le macrobloc en cours $pred_{Cb}[x, y]$ et $pred_{Cr}[x, y]$.

Les deux blocs chroma (Cb et Cr) du macrobloc utilisent le même mode de prédiction. Le mode de prédiction est appliqué à chaque bloc chroma séparément. Le processus spécifié au présent paragraphe est invoqué pour chaque bloc chroma. Dans le reste du présent paragraphe, bloc chroma se rapporte à un des deux blocs chroma et la mention C est utilisée en remplacement de la mention Cb ou Cr.

Les échantillons voisins p[x, y] qui sont des échantillons chroma construits avant le processus de filtre de démontage de bloc, avec x = -1, y = -1..MbHeightC – 1 et avec x = 0..MbWidthC – 1, y = -1, sont calculés comme suit:

– Le processus de déduction pour les localisations du voisinage du § 6.4.9 est invoqué pour les localisations chroma avec (x, y) alloué à (xN, yN) comme entrée et mbAddrN et (xW, yW) comme résultat.

– Chaque échantillon p[x, y] est déduit comme suit.

  – Si l'une des conditions suivantes est vraie, l'échantillon p[x, y] est marqué "indisponible pour l'intraprédiction chroma":

    – mbAddrN n'est pas disponible;

    – le macrobloc mbAddrN est codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1;

    – le macrobloc mbAddrN a mb_type égal à SI et constrained_intra_pred_flag est égal à 1 et le macrobloc en cours n'a pas mb_type égal à SI.

  – Sinon, l'échantillon p[x, y] est marqué "disponible pour l'intraprédiction chroma" et l'échantillon chroma de composant C à la localisation chroma (xW, yW) au sein du macrobloc mbAddrN est alloué à p[x, y].

Soit $pred_C[x, y]$ avec x = 0..MbWidthC – 1, y = 0..MbHeightC – 1 qui indique les échantillons de prédiction pour les échantillons de bloc chroma.

Les modes d'intraprédiction chroma sont spécifiés au Tableau 8-5.

Tableau 8-5 – Spécification des modes d'intraprédiction chroma et des noms associés

intra_chroma_pred_mode	Nom de intra_chroma_pred_mode
0	Intra_Chroma_DC (mode de prédiction)
1	Intra_Chroma_Horizontal (mode de prédiction)
2	Intra_Chroma_Vertical (mode de prédiction)
3	Intra_Chroma_Plane (mode de prédiction)

En fonction de intra_chroma_pred_mode, un des modes d'intraprédiction chroma spécifiés aux § 8.3.4.1 à 8.3.4.4 est invoqué.

8.3.4.1 Spécification du mode de prédiction Intra_Chroma_DC

Ce mode de mode de prédiction intrachromatique est invoqué lorsque intra_chroma_pred_mode est égal à 0.

Pour chaque bloc chroma de 4x4 échantillons indexés par chroma4x4BlkIdx = 0..(1 << (chroma_format_idc + 1)) – 1, ce qui suit est applicable:

– Selon chroma_format_idc, la position de l'échantillon supérieur gauche d'un bloc chroma 4x4 ayant l'indice chroma4x4BlkIdx est calculé comme suit:

– Si chroma_format_idc est égal à 1 ou 2, ce qui suit est applicable:

xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (8-124)

yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (8-125)

– Sinon (si chroma_format_idc est égal à 3), ce qui suit est applicable:

xO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 0) (8-126)

yO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 1) (8-127)

– Si (xO, yO) est égal à (0, 0) ou si xO et yO sont plus grands que 0, les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$ avec x, y = 0..3 sont calculées comme suit:

– Si tous les échantillons p[x + xO, -1], avec x = 0..3 et p[-1, y +yO], avec y = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$, avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[x + xO, y + yO] = \left(\sum_{x'=0}^{3} p[x'+xO,-1] + \sum_{y'=0}^{3} p[-1, y'+yO] + 4 \right) >> 3 \text{ , avec } x, y = 0..3 \quad (8\text{-}128)$$

– Sinon, si tous les échantillons p[x + xO, -1], avec x = 0..3 sont marqués "indisponible pour prédiction intrachromatique" et si tous les échantillons p[-1, y +yO], avec y = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$, avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[x + xO, y + yO] = \left(\sum_{y'=0}^{3} p[-1, y'+yO] + 2 \right) >> 2 \text{ , avec } x, y = 0..3 \quad (8\text{-}129)$$

– Sinon, si certains échantillons p[-1, y +yO], avec y = 0..3 sont marqués "indisponible pour prédiction intrachromatique" et si tous les échantillons p[x + xO, -1], avec x = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$, avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[x + xO, y + yO] = \left(\sum_{x'=0}^{3} p[x'+xO,-1] + 2 \right) >> 2 \text{ , avec } x, y = 0..3 \quad (8\text{-}130)$$

– Sinon (si certains échantillons p[x + xO, -1], avec x = 0..3 et si certains échantillons p[-1, y +yO], avec y = 0..3 sont marqués "indisponible pour prédiction intrachromatique"), les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$, avec x, y = 0..3 sont des variables calculées comme suit:

$pred_C[x + xO, y + yO] = (1 << (BitDepth_C - 1))$, avec x, y = 0..3 (8-131)

– Sinon, si xO est plus grand que 0 et si yO est égal à 0, les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$ avec x, y = 0..3 sont calculées comme suit:

– Si tous les échantillons p[x + xO, -1], avec x = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C[x + xO, y + yO]$, avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[x + xO, y + yO] = \left(\sum_{x'=0}^{3} p[x'+xO,-1] + 2 \right) >> 2 \text{ , avec } x, y = 0..3 \quad (8\text{-}132)$$

– Sinon, si tous les échantillons p[-1, y +yO], avec y = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C$[x + xO, y + yO], avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[\, x + xO, y + yO \,] = \left(\sum_{y'=0}^{3} p[-1, y'+yO] + 2 \right) >> 2 \text{ , avec } x, y = 0..3 \tag{8-133}$$

– Sinon (si certains échantillons p[x + xO, -1], avec x = 0..3 et certains échantillons p[-1, y +yO], avec y = 0..3 sont marqués "indisponible pour prédiction intrachromatique"), les valeurs des échantillons de prédiction $pred_C$[x + xO, y + yO], avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[\, x + xO, y + yO \,] = (1 << (BitDepth_C - 1)), \text{ avec } x, y = 0..3 \tag{8-134}$$

– Sinon (si xO est égal à 0 et yO est plus grand que 0), les valeurs des échantillons de prédiction $pred_C$[x + xO, y + yO] avec x, y = 0..3 sont calculées comme suit:

– Si tous les échantillons p[-1, y +yO], avec y = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C$[x + xO, y + yO], avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[\, x + xO, y + yO \,] = \left(\sum_{y'=0}^{3} p[-1, y'+yO] + 2 \right) >> 2 \text{ , avec } x, y = 0..3 \tag{8-135}$$

– Sinon, si tous les échantillons p[x + xO, -1], avec x = 0..3 sont marqués "disponible pour prédiction intrachromatique", les valeurs des échantillons de prédiction $pred_C$[x + xO, y + yO], avec x, y = 0..3 sont calculées comme suit:

$$pred_C[\, x + xO, y + yO \,] = \left(\sum_{x'=0}^{3} p[x'+xO, -1] + 2 \right) >> 2 \text{ , avec } x, y = 0..3 \tag{8-136}$$

– Sinon (si certains échantillons p[x + xO, -1], avec x = 0..3 et certains échantillons p[-1, y +yO], avec y = 0..3 sont marqués "indisponible pour prédiction intrachromatique"), les valeurs des échantillons de prédiction $pred_C$[x + xO, y + yO], avec x, y = 0..3 sont des variables calculées comme suit:

$$pred_C[\, x + xO, y + yO \,] = (1 << (BitDepth_C - 1)), \text{ avec } x, y = 0..3 \tag{8-137}$$

8.3.4.2 Spécification du mode de prédiction Intra_Chroma_Horizontal

Ce mode de mode de prédiction intrachromatique est invoqué lorsque intra_chroma_pred_mode est égal à 1.

Ce mode ne doit être utilisé que lorsque les échantillons p[-1, y] avec y = 0..MbHeightC – 1 sont marqués "disponible pour prédiction intrachromatique".

Les valeurs des échantillons de prédiction $pred_C$[x, y] sont calculées comme suit:

$$pred_C[\, x, y \,] = p[\, -1, y \,], \text{ avec } x = 0..MbWidthC - 1 \text{ et } y = 0..MbHeightC - 1 \tag{8-138}$$

8.3.4.3 Spécification du mode de prédiction Intra_Chroma_Vertical

Ce mode de mode de prédiction intrachromatique est invoqué lorsque intra_chroma_pred_mode est égal à 2.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..MbWidthC – 1 sont marqués "disponible pour prédiction intrachromatique".

Les valeurs des échantillons de prédiction $pred_C$[x, y] sont calculées comme suit:

$$pred_C[\, x, y \,] = p[\, x, -1 \,], \text{ avec } x = 0..MbWidthC - 1 \text{ et } y = 0..MbHeightC - 1 \tag{8-139}$$

8.3.4.4 Spécification du mode de prédiction Intra_Chroma_Plane

Ce mode de mode de prédiction intrachromatique est invoqué lorsque intra_chroma_pred_mode est égal à 3.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1], avec x = 0..MbWidthC – 1 et p[-1, y], avec y = -1..MbHeightC – 1 sont marqués "disponible pour prédiction intrachromatique".

Les valeurs des échantillons de prédiction $pred_C$[x, y] sont calculées comme suit:

Soit la variable xCF mise égale à 4 * (chroma_format_idc = = 3) et soit la variable yCF mise égale à 4 * (chroma_format_idc != 1).

$$pred_C[x, y] = Clip1_C((a + b * (x – 3 – xCF) + c * (y – 3 – yCF) + 16) >> 5),$$
avec x = 0..MbWidthC - 1 et y = 0..MbHeightC - 1 (8-140)

où:

$$a = 16 * (p[-1, MbHeightC - 1] + p[MbWidthC - 1, -1]) \quad (8\text{-}141)$$

$$b = ((34 – 29 * (chroma\_format\_idc = = 3)) * H + 32) >> 6 \quad (8\text{-}142)$$

$$c = ((34 – 29 * (chroma\_format\_idc != 1)) * V + 32) >> 6 \quad (8\text{-}143)$$

et où H et V sont spécifiés par:

$$H = \sum_{x'=0}^{3+xCF} (x'+1) * (p[4 + xCF + x', -1] - p[2 + xCF - x', -1]) \quad (8\text{-}144)$$

$$V = \sum_{y'=0}^{3+yCF} (y'+1) * (p[-1, 4 + yCF + y'] - p[-1, 2 + yCF - y']) \quad (8\text{-}145)$$

8.3.5 Processus de construction des échantillons pour macroblocs I_PCM

Ce processus est invoqué lorsque mb_type est égal à I_PCM.

La variable dy est déduite comme suit.

– Si MbaffFrameFlag est égal à 1 et que le macrobloc en cours soit un macrobloc de sous-trame, dy est mise égale à 2.

– Sinon (si MbaffFrameFlag est égal à 0 ou si le macrobloc en cours est un macrobloc de trame), dy est mis égal à 1.

La position de l'échantillon luma supérieur gauche du macrobloc en cours est déduite en invoquant le processus de balayage de macrobloc inverse du § 6.4.1 avec CurrMbAddr comme entrée et le résultat étant alloué à (xP, yP).

Les échantillons luma construits antérieurement au processus de démontage de bloc sont produits comme spécifié par:

```
pour ( i = 0; i < 256; i++ )
    S'L[ xP + ( i % 16 ), yP + dy * ( i / 16 ) ) ] = pcm_sample_luma[ i ]    (8-146)
```

Lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), les échantillons chroma construits avant le processus de démontage de bloc sont produits comme spécifié par:

```
pour( i = 0; i < MbWidthC * MbHeightC; i++ ) {
    S'Cb[ ( xP / SubWidthC ) + ( i % MbWidthC ),
                ( ( yP + SubHeightC – 1 ) / SubHeightC ) + dy * ( i / MbWidthC ) ] =
                  pcm_sample_chroma[ i ]                                    (8-147)
    S'Cr[ ( xP / SubWidthC ) + ( i % MbWidthC ),
                ( ( yP + SubHeightC – 1 ) / SubHeightC ) + dy * ( i / MbWidthC ) ] =
                  pcm_sample_chroma[ i + MbWidthC * MbHeightC ]
}
```

8.4 Processus d'interprédiction

Ce processus est invoqué lors du décodage des types de macrobloc P et B.

Les résultats de ce processus sont des échantillons d'interprédiction pour le macrobloc en cours qui sont une matrice 16x16 $pred_L$ d'échantillons luma et, lorsque chroma_format_idcchroma_format_idc n'est pas égal à 0 (format monochrome), deux matrices 8x8 $pred_{Cr}$ et $pred_{Cb}$ d'échantillons chroma, une pour chacune des composantes chroma Cb et Cr.

La subdivision d'un macrobloc est spécifiée par mb_type. Chaque subdivision de macrobloc est désignée par mbPartIdx. Lorsque la subdivision de macrobloc consiste en partitions qui sont égales à des sous-macroblocs, chaque sous-macrobloc peut encore être partagé en subdivisions de sous-macrobloc comme spécifié par sub_mb_type. Chaque subdivision de sous-macrobloc est désignée par subMbPartIdx. Lorsque la subdivision de macrobloc ne comporte pas de sous-macroblocs, subMbPartIdx est mis égal à 0.

Les étapes suivantes sont spécifiées pour chaque subdivision de macrobloc ou pour chaque subdivision de sous-macrobloc.

Les fonctions MbPartWidth( ), MbPartHeight( ), SubMbPartWidth( ) et SubMbPartHeight( ) décrivant la largeur et la hauteur des subdivisions de macrobloc et subdivisions de sous-macrobloc sont spécifiées aux Tableaux 7-13, 7-14, 7-17 et 7-18.

L'étendue de l'indice de subdivision de macrobloc mbPartIdx est calculée comme suit:

– Si mb_type est égal à B_Skip ou B_Direct_16x16, mbPartIdx saute les valeurs 0..3.

– Sinon (si mb_type n'est pas égal à B_Skip ou à B_Direct_16x16), mbPartIdx saute les valeurs 0..NumMbPart(mb_type) – 1.

Pour chaque valeur de mbPartIdx, les variables partWidth et partHeight pour chaque subdivision de macrobloc ou sous-subdivision de macrobloc dans le macrobloc sont calculées comme suit:

– Si mb_type n'est pas égal à P_8x8, P_8x8ref0, B_Skip, B_Direct_16x16, ou B_8x8, subMbPartIdx est mis à 0, et partWidth et partHeight sont des variables calculées comme suit:

$$partWidth = MbPartWidth(mb\_type) \quad (8\text{-}148)$$

$$partHeight = MbPartHeight(mb\_type) \quad (8\text{-}149)$$

– Sinon, si mb_type est égal à P_8x8 ou à P_8x8ref0, ou si mb_type est égal à B_8x8 et si sub_mb_type[mbPartIdx] n'est pas égal à B_Direct_8x8, subMbPartIdx saute les valeurs 0..NumSubMbPart(sub_mb_type) – 1, et partWidth et partHeight sont des variables calculées comme suit:

$$partWidth = SubMbPartWidth(sub\_mb\_type[mbPartIdx]) \quad (8\text{-}150)$$

$$partHeight = SubMbPartHeight(sub\_mb\_type[mbPartIdx]) \quad (8\text{-}151)$$

– Sinon (si mb_type est égal à B_Skip ou à B_Direct_16x16, ou si mb_type est égal à B_8x8 et si sub_mb_type[mbPartIdx] est égal à B_Direct_8x8), subMbPartIdx saute les valeurs 0..3, et partWidth et partHeight sont des variables calculées comme suit:

$$partWidth = 4 \quad (8\text{-}152)$$

$$partHeight = 4 \quad (8\text{-}153)$$

Lorsque chroma_format_idc n'est pas égal à 0 (format monochrome) les variables partWidthC et partHeightC sont des variables calculées comme suit:

$$partWidthC = partWidth / SubWidthC \quad (8\text{-}154)$$
$$partHeightC = partHeight / SubHeightC \quad (8\text{-}155)$$

Soit la variable MvCnt initialement mise égale à 0 avant toute invocation de § 8.4.1 pour le macrobloc.

Le processus d'interprédiction pour une subdivision de macrobloc mbPartIdx et pour une subdivision de sous-macrobloc subMbPartIdx comporte les étapes ordonnées suivantes:

1. Processus de déduction pour les composantes de vecteur cinétique et les indices de référence comme spécifié au § 8.4.1.

   Les entrées dans ce processus sont:

   – une subdivision de macrobloc mbPartIdx;

   – une subdivision de sous-macrobloc subMbPartIdx.

   Les résultats de ce processus sont:

   – les vecteurs cinétiques luma mvL0 et mvL1 et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), les vecteurs cinétiques chroma mvCL0 et mvCL1;

   – les indices de référence refIdxL0 et refIdxL1;

   – les fanions d'utilisation de liste de prédiction predFlagL0 et predFlagL1;

   – le compte de vecteur cinétique de subdivision de sous-macrobloc subMvCnt.

2. La variable MvCnt est incrémentée de subMvCnt.

3. Processus de décodage pour les échantillons d'interprédiction comme spécifié au § 8.4.2.

   Les entrées dans ce processus sont:

   – une subdivision de macrobloc mbPartIdx;

   – une subdivision de sous-macrobloc subMbPartIdx;

   – les variables spécifiant la largeur et la hauteur de la subdivision pour luma et chroma (si disponibles), partWidth, partHeight, partWidthC (si disponibles), et partHeightC (si disponible);

   – les vecteurs cinétiques luma mvL0 et mvL1 et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), les vecteurs cinétiques chroma mvCL0 et mvCL1;

   – les indices de référence refIdxL0 et refIdxL1;

   – les fanions d'utilisation de liste de prédiction predFlagL0 et predFlagL1.

   Les résultats de ce processus sont:

   – des échantillons d'interprédiction (pred); qui sont une matrice (partWidth)x(partHeight) $predPart_L$ d'échantillons luma de prédiction et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), deux matrices (partWidthC)x(partHeightC) $predPart_{Cr}$ et $predPart_{Cb}$ d'échantillons chroma de prédiction, une pour chacune des composantes chroma Cb et Cr.

Pour l'utilisation dans les processus de déduction de variables invoquées ultérieurement dans le processus de décodage, on fait les allocations suivantes:

MvL0[mbPartIdx][subMbPartIdx] = mvL0 (8-156)

MvL1[mbPartIdx][subMbPartIdx] = mvL1 (8-157)

RefIdxL0[mbPartIdx] = refIdxL0 (8-158)

RefIdxL1[mbPartIdx] = refIdxL1 (8-159)

PredFlagL0[mbPartIdx] = predFlagL0 (8-160)

PredFlagL1[mbPartIdx] = predFlagL1 (8-161)

La localisation de l'échantillon supérieur gauche de la subdivision relative à l'échantillon supérieur gauche du macrobloc est déduite en invoquant le processus de balayage inverse de subdivision de macrobloc comme décrit au § 6.4.2.1 avec mbPartIdx comme entrée et (xP, yP) comme résultat.

La localisation de l'échantillon supérieur gauche de la subdivision de macrobloc relative à l'échantillon supérieur gauche de la subdivision de macrobloc est déduite en invoquant le processus de balayage inverse de division de sous-macrobloc comme décrit au § 6.4.2.2 avec subMbPartIdx comme entrée et (xS, yS) comme résultat.

La prédiction de macrobloc est formée en plaçant comme suit les échantillons de prédiction de subdivision ou de subdivision de sous-macrobloc dans leurs positions relatives correctes dans le macrobloc.

La variable $pred_L[xP + xS + x, yP + yS + y]$, avec x = 0 .. partWidth – 1, y = 0 .. partHeight – 1, est calculée par

$$pred_L[xP + xS + x, yP + yS + y] = predPart_L[x, y] \quad (8\text{-}162)$$

Lorsque chroma_format_idc n'est pas égal à 0 (monochrome) la variable $pred_C$ avec x = 0..partWidthC – 1, y = 0..partHeightC – 1, et avec C dans $pred_C$ et dans $predPart_C$ remplacé par Cb ou Cr est calculée par:

$$pred_C[xP / SubWidthC + xS / SubWidthC + x, yP / SubHeightC + yS / SubHeightC + y] = predPart_C[x, y] \quad (8\text{-}163)$$

8.4.1 Processus de déduction pour composantes de vecteur cinétique et indices de référence

Les entrées dans ce processus sont:

– une subdivision de macrobloc mbPartIdx;

– une subdivision de sous-macrobloc subMbPartIdx.

Les résultats de ce processus sont:

– des vecteurs cinétiques luma mvL0 et mvL1 ainsi que des vecteurs cinétiques chroma mvCL0 et mvCL1;

– des indices de référence refIdxL0 et refIdxL1;

– des fanions d'utilisation de liste de prédiction predFlagL0 et predFlagL1;

– une variable compte de vecteur cinétique de subdivision de sous-macrobloc subMvCnt.

Pour la déduction des variables mvL0 et mvL1 ainsi que refIdxL0 et refIdxL1, on applique ce qui suit:

– si mb_type est égal à P_Skip, le processus de déduction pour les vecteurs cinétiques luma pour des macroblocs sautés dans des tranches P et SP du § 8.4.1.1 est invoqué avec comme résultat des vecteurs cinétiques luma mvL0 et des indices de référence refIdxL0 et predFlagL0 est mis égal à 1. mvL1 et refIdxL1 sont marqués comme indisponibles et predFlagL1 est mis égal à 0. La variable compte de vecteur cinétique de subdivision de sous-macrobloc subMvCnt est mise égale à 1.

– Sinon, si mb_type est égal à B_Skip ou si B_Direct_16x16 ou sub_mb_type[mbPartIdx] est égal à B_Direct_8x8, le processus de déduction pour les vecteurs cinétiques luma pour B_Skip, B_Direct_16x16 et B_Direct_8x8 dans les tranches B du § 8.4.1.2 est invoqué avec mbPartIdx et subMbPartIdx comme entrée, le résultat étant des vecteurs cinétiques luma mvL0, mvL1, des indices de référence refIdxL0, refIdxL1, le compte de vecteur cinétique de subdivision subMvCnt, et les fanions d'utilisation de prédiction predFlagL0 et predFlagL1.

– Sinon, pour X remplacé par 0 ou 1 dans les variables predFlagLX, mvLX, refIdxLX et dans Pred_LX et dans les éléments syntaxiques ref_idx_lX et mvd_lX, on applique ce qui suit.

1. Les variables refIdxLX et predFlagLX sont calculées comme suit.

   – Si MbPartPredMode(mb_type, mbPartIdx) ou SubMbPredMode(sub_mb_type[mbPartIdx]) est égal à Pred_LX ou à BiPred,

   $$refIdxLX = ref\_idx\_lX[mbPartIdx] \quad (8\text{-}164)$$

   $$predFlagLX = 1 \quad (8\text{-}165)$$

   – Sinon, les variables refIdxLX et predFlagLX sont spécifiées par

   $$refIdxLX = -1 \quad (8\text{-}166)$$

predFlagLX = 0 (8-167)

2. La variable subMvCnt pour le compte de vecteur cinétique de subdivision est mise égale à predFlagL0 + predFlagL1.
3. La variable currSubMbType est calculée comme suit:
   – Si le macrobloc type est égal à B_8x8, currSubMbType est mis à sub_mb_type[mbPartIdx].
   – Sinon (si le macrobloc type n'est pas égal à B_8x8), currSubMbType est mis à "na".
4. Lorsque predFlagLX est égal à 1, le processus de déduction pour la prédiction de vecteur cinétique luma du § 8.4.1.3 est invoqué avec mbPartIdx, subMbPartIdx, refIdxLX et currSubMbType comme entrées et le résultat étant mvpLX. Les vecteurs cinétiques luma sont déduits par:

mvLX[0] = mvpLX[0] + mvd_lX[mbPartIdx][subMbPartIdx][0] (8-168)

mvLX[1] = mvpLX[1] + mvd_lX[mbPartIdx][subMbPartIdx][1] (8-169)

Pour la déduction des variables pour les vecteurs cinétiques chroma, on applique ce qui suit. Lorsque predFlagLX (avec X étant 0 ou 1) est égal à 1, le processus de déduction pour des vecteurs cinétiques chroma du § 8.4.1.4 est invoqué avec mvLX et refIdxLX comme entrée et le résultat étant mvCLX.

8.4.1.1 Processus de déduction des vecteurs cinétiques luma pour macroblocs sautés en tranches P et SP

Ce processus est invoqué lorsque mb_type est égal à P_Skip.

Les résultats de ce processus sont le vecteur cinétique mvL0 et l'indice de référence refIdxL0.

L'indice de référence refIdxL0 pour un macrobloc sauté est déduit comme suit.

refIdxL0 = 0. (8-170)

Pour la déduction du vecteur cinétique mvL0 d'un type de macrobloc P_Skip, on applique ce qui suit.

– Le processus spécifié au § 8.4.1.3.2 est invoqué avec mbPartIdx mis égal à 0, subMbPartIdx mis égal à 0, currSubMbType mis égal à "na" et listSuffixFlag mis égal à 0 comme entrées et le résultat est alloué à mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A et refIdxL0B.
– La variable mvL0 est spécifiée comme suit.
  – Si certaines des conditions suivantes sont vraies, les deux composantes du vecteur cinétique mvL0 sont mises égales à 0:
    – mbAddrA n'est pas disponible;
    – mbAddrB n'est pas disponible;
    – refIdxL0A est égal à 0 et les deux composantes de mvL0A sont égales à 0;
    – refIdxL0B est égal à 0 et les deux composantes de mvL0B sont égales à 0.
  – Sinon, le processus de calcul pour la prédiction de vecteur cinétique luma comme spécifié dans le § 8.4.1.3 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, refIdxL0, et currSubMbType = "na" en entrées et la sortie est attribuée à mvL0.

NOTE – Le résultat est directement alloué à mvL0, dans la mesure où le prédicteur est égal au vecteur cinétique réel.

8.4.1.2 Processus de déduction des vecteurs cinétiques luma pour B_Skip, B_Direct_16x16 et B_Direct_8x8

Ce processus est invoqué lorsque mb_type est égal à B_Skip ou B_Direct_16x16 ou que sub_mb_type[mbPartIdx] est égal à B_Direct_8x8.

Les entrées dans ce processus sont mbPartIdx et subMbPartIdx.

Les résultats de ce processus sont les indices de référence refIdxL0, refIdxL1, les vecteurs cinétiques mvL0 et mvL1, le compte vecteur cinétique de subdivision subMvCnt, et les fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1.

Le processus de déduction dépend de la valeur de direct_spatial_mv_pred_flag, qui est présente dans le flux binaire dans la syntaxe d'en-tête de tranche comme spécifié au § 7.3.3 et il est spécifié comme suit.

– Si direct_spatial_mv_pred_flag est égal à 1, le mode dans lequel sont déduits les résultats de ce processus est désigné sous le nom de mode de prédiction direct spatial.

– Sinon (si direct_spatial_mv_pred_flag est égal à 0), le mode dans lequel sont déduits les résultats de ce processus est désigné sous le nom de mode de prédiction direct temporel.

Les modes de prédiction directs spatial et temporel utilisent tous deux les vecteurs cinétiques et indices de référence colocalisés, comme spécifié au § 8.4.1.2.1.

Les vecteurs cinétiques et indices de référence sont déduits comme suit.

– Si le mode de prédiction direct spatial est utilisé, le mode de prédiction direct de vecteur cinétique et d'indice de référence spécifié au § 8.4.1.2.2 est utilisé, avec subMvCnt comme résultat.

– Sinon (si le mode de prédiction direct temporel est utilisé), le mode de prédiction direct de vecteur cinétique et d'indice de référence spécifié au § 8.4.1.2.3 est utilisé, et la variable subMvCnt est calculée comme suit.

  – Si subMbPartIdx est égal à 0, subMvCnt est mis à 2.

  – Sinon (si subMbPartIdx est différent de 0), subMvCnt est mis à 0.

8.4.1.2.1 Processus de déduction pour la subdivision de sous-macroblocs 4x4 colocalisés

Les entrées dans ce processus sont mbPartIdx et subMbPartIdx.

Les résultats de ce processus sont l'image colPic, le macrobloc colocalisé mbAddrCol, le vecteur cinétique mvCol, l'indice de référence refIdxCol et la variable vertMvScale (qui peut être One_To_One, Frm_To_Fld ou Fld_To_Frm).

Lorsque RefPicList1[0] est une trame ou une paire de sous-trames complémentaires, soient respectivement firstRefPicL1Top et firstRefPicL1Bottom les sous-trames supérieure et inférieure de RefPicList1[0] et soient les variables suivantes spécifiées comme suit:

topAbsDiffPOC = Abs(DiffPicOrderCnt(firstRefPicL1Top, CurrPic)) (8-171)

bottomAbsDiffPOC = Abs(DiffPicOrderCnt(firstRefPicL1Bottom, CurrPic)) (8-172)

La variable colPic spécifie l'image qui contient le macrobloc colocalisé comme spécifié au Tableau 8-6.

Tableau 8-6 – Spécification de la variable colPic

field_pic_flag	RefPicList1[0] est …	mb_field_decoding_flag	Condition supplémentaire	colPic
1	une sous-trame d'une trame décodée			la trame contenant RefPicList1[0]
	un sous-trame décodée			RefPicList1[0]
0	une trame décodée			RefPicList1[0]
	une paire de sous-trames complémentaires	0	topAbsDiffPOC < bottomAbsDiffPOC	la sous-trame supérieure de firstRefPicL1Top
			topAbsDiffPOC >= bottomAbsDiffPOC	la sous-trame inférieure de firstRefPicL1Bottom
		1	(CurrMbAddr & 1) == 0	la sous-trame supérieure de firstRefPicL1Top
			(CurrMbAddr & 1) != 0	la sous-trame inférieure de firstRefPicL1Bottom

Lorsque direct_8x8_inference_flag est égal à 1, subMbPartIdx est mis comme suit.

subMbPartIdx = mbPartIdx (8-173)

Soit PicCodingStruct(X) une fonction avec l'argument X CurrPic ou colPic. Elle est spécifiée au Tableau 8-7.

Tableau 8-7 – Spécification de PicCodingStruct(X)

X est codé avec field_pic_flag égal à …	mb_adaptive_frame_field_flag	PicCodingStruct(X)
1		FLD
0	0	FRM
0	1	AFRM

Avec luma4x4BlkIdx = mbPartIdx * 4 + subMbPartIdx, le processus de balayage de bloc luma 4x4 inverse comme spécifié au § 6.4.3 est invoqué avec luma4x4BlkIdx comme entrée et (x, y) alloué à (xCol, yCol) comme résultat.

Le Tableau 8-8 spécifie en deux étapes l'adresse de colocalisation de macrobloc mbAddrCol, yM et la variable vertMvScale:

1. spécification d'une adresse de macrobloc mbAddrX en fonction de PicCodingStruct(CurrPic) et de PicCodingStruct(colPic).

   NOTE – Il n'est pas possible pour les types de codage d'image CurrPic et colPic d'être (FRM, AFRM) ou (AFRM, FRM) parce que ces types de codage d'image doivent être séparés par uneimage à rafraîchissement IDR;

2. spécification de mbAddrCol, yM et vertMvScale en fonction de mb_field_decoding_flag et de la variable fieldDecodingFlagX, qui est déduite comme suit.

   – Si le macrobloc mbAddrX dans l'image colPic est un macrobloc de sous-trame, fieldDecodingFlagX est mis égal à 1.

   – Sinon (si le macrobloc mbAddrX dans l'image colPic est un macrobloc de trame), fieldDecodingFlagX est mis égal à 0.

Les valeurs non spécifiées dans le Tableau 8-8 indiquent que la valeur de la variable correspondante n'est pas pertinente pour la rangée en cours du tableau.

mbAddrCol reçoit la même valeur que CorrMbAddr ou que l'une des valeurs suivantes:

$$mbAddrCol1 = 2 * PicWidthInMbs * (CurrMbAddr / PicWidthInMbs) + (CurrMbAddr \% PicWidthInMbs) + PicWidthInMbs * (yCol / 8) \quad (8\text{-}174)$$

$$mbAddrCol2 = 2 * CurrMbAddr + (yCol / 8) \quad (8\text{-}175)$$

$$mbAddrCol3 = 2 * CurrMbAddr + bottom\_field\_flag \quad (8\text{-}176)$$

$$mbAddrCol4 = PicWidthInMbs * (CurrMbAddr / (2 * PicWidthInMbs)) + (CurrMbAddr \% PicWidthInMbs) \quad (8\text{-}177)$$

$$mbAddrCol5 = CurrMbAddr / 2 \quad (8\text{-}178)$$

$$mbAddrCol6 = 2 * (CurrMbAddr / 2) + ((topAbsDiffPOC < bottomAbsDiffPOC) ? 0 : 1) \quad (8\text{-}179)$$

$$mbAddrCol7 = 2 * (CurrMbAddr / 2) + (yCol / 8) \quad (8\text{-}180)$$

Tableau 8-8 – Spécification de mbAddrCol, yM et vertMvScale

PicCodingStruct(CurrPic)	PicCodingStruct(colPic)	mbAddrX	mb_field_decoding_flag	fieldDecodingFlagX	mbAddrCol	yM	vertMvScale
FLD	FLD				CurrMbAddr	yCol	One_To_One
	FRM				mbAddrCol1	(2 * yCol) % 16	Frm_To_Fld
	AFRM	2*CurrMbAddr		0	mbAddrCol2	(2 * yCol) % 16	Frm_To_Fld
				1	mbAddrCol3	yCol	One_To_One
FRM	FLD				mbAddrCol4	8 * ((CurrMbAddr / PicWidthInMbs) % 2) + 4 * (yCol / 8)	Fld_To_Frm
	FRM				CurrMbAddr	yCol	One_To_One
AFRM	FLD		0		mbAddrCol5	8 * (CurrMbAddr % 2) +4 * (yCol / 8)	Fld_To_Frm
			1		mbAddrCol5	yCol	One_To_One
	AFRM	CurrMbAddr	0	0	CurrMbAddr	yCol	One_To_One
				1	mbAddrCol6	8 * (CurrMbAddr % 2) + 4 * (yCol / 8)	Fld_To_Frm
		CurrMbAddr	1	0	mbAddrCol7	(2 * yCol) % 16	Frm_To_Fld
				1	CurrMbAddr	yCol	One_To_One

Soient mbPartIdxCol l'indice de subdivision de macrobloc de la subdivision colocalisée et subMbPartIdxCol l'indice de subdivision de sous-macrobloc de la subdivision colocalisée de sous-macrobloc. La subdivision dans le macrobloc mbAddrCol au sein de l'image colPic couvrant l'échantillon (xCol, yM) est allouée à mbPartIdxCol et la subdivision de sous-macrobloc au sein de la subdivision mbPartIdxCol couvrant l'échantillon (xCol, yM) dans le macrobloc mbAddrCol au sein de l'image colPic est allouée à subMbPartIdxCol.

Les fanions d'utilisation de prédiction predFlagL0Col et predFlagL1Col sont respectivement mis égaux à PredFlagL0[mbPartIdxCol] et PredFlagL1[mbPartIdxCol], qui sont les fanions d'utilisation de prédiction qui ont été alloués à la subdivision de macrobloc mbAddrCol\mbPartIdxCol au sein de l'image colPic.

Le vecteur cinétique mvCol et l'indice de référence refIdxCol sont déduits comme suit.

– Si le macrobloc mbAddrCol est codé en mode de prédiction intramacrobloc ou que les deux fanions d'utilisation de prédiction, predFlagL0Col et predFlagL1Col, sont égaux à 0, les deux composantes de mvCol sont mises égales à 0 et refIdxCol est mis égal à –1.

– Sinon, on applique ce qui suit.

  – Si predFlagL0Col est égal à 1, le vecteur cinétique mvCol et l'indice de référence refIdxCol sont mis respectivement égaux à MvL0[mbPartIdxCol][subMbPartIdxCol] et RefIdxL0[mbPartIdxCol], qui sont le vecteur cinétique mvL0 et l'indice de référence refIdxL0 qui ont été alloués à la subdivision de (sous-)macrobloc mbAddrCol\mbPartIdxCol\subMbPartIdxCol au sein de l'image colPic.

  – Sinon (si predFlagL0Col est égal à 0 et predFlagL1Col est égal à 1), le vecteur cinétique mvCol et l'indice de référence refIdxCol sont respectivement mis égaux à MvL1[mbPartIdxCol][subMbPartIdxCol] et RefIdxL1[mbPartIdxCol], qui sont le vecteur cinétique mvL1 et l'indice de référence refIdxL1 qui ont été alloués à la subdivision de (sous-)macrobloc mbAddrCol\mbPartIdxCol\subMbPartIdxCol au sein de l'image colPic.

8.4.1.2.2 Processus de déduction pour le mode de prédiction de vecteur cinétique et d'indice de référence luma direct spatial

Ce processus est invoqué lorsque direct_spatial_mv_pred_flag est égal à 1 et qu'une des conditions suivantes est vraie.

– mb_type est égal à B_Skip

– mb_type est égal à B_Direct_16x16

– sub_mb_type[mbPartIdx] est égal à B_Direct_8x8.

Les entrées dans ce processus sont mbPartIdx, subMbPartIdx.

Les résultats de ce processus sont les indices de référence refIdxL0, refIdxL1, les vecteurs cinétiques mvL0 et mvL1, le compte de vecteur cinétique de subdivision subMvCnt, et les fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1.

Les indices de référence refIdxL0 et refIdxL1 et la variable directZeroPredictionFlag sont déduits par application de les étapes ordonnées suivantes:

1. Soit la variable currSubMbType égale à sub_mb_type[mbPartIdx].
2. Le processus spécifié au § 8.4.1.3.2 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, currSubMbType et listSuffixFlag = 0 comme entrées et le résultat est alloué aux vecteurs cinétiques mvL0N et aux indices de référence refIdxL0N avec N remplacé par A, B ou C.
3. Le processus spécifié au § 8.4.1.3.2 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, currSubMbType et listSuffixFlag = 1 comme entrées et le résultat est alloué aux vecteurs cinétiques mvL1N et aux indices de référence refIdxL1N avec N remplacé par A, B ou C.

   NOTE 1 – Les vecteurs cinétiques mvL0N, mvL1N et les indices de référence refIdxL0N, refIdxL1N sont identiques pour toutes les subdivisions de sous-macrobloc 4x4 d'un macrobloc.

4. Les indices de référence refIdxL0, refIdxL1 et directZeroPredictionFlag sont déduits par:

$$refIdxL0 = MinPositive(refIdxL0A, MinPositive(refIdxL0B, refIdxL0C)) \quad (8\text{-}181)$$
$$refIdxL1 = MinPositive(refIdxL1A, MinPositive(refIdxL1B, refIdxL1C)) \quad (8\text{-}182)$$
$$directZeroPredictionFlag = 0 \quad (8\text{-}183)$$

   où:

$$MinPositive(x, y) = \begin{cases} Min(x, y) \text{ si } x > 0 \text{ et } y >= 0 \\ Max(x, y) \text{ autrement} \end{cases} \quad (8\text{-}184)$$

5. Lorsque les deux indices de référence refIdxL0 et refIdxL1 sont inférieurs à 0,

$$refIdxL0 = 0 \quad (8\text{-}185)$$
$$refIdxL1 = 0 \quad (8\text{-}186)$$
$$directZeroPredictionFlag = 1 \quad (8\text{-}187)$$

Le processus spécifié au § 8.4.1.2.1 est invoqué avec mbPartIdx, subMbPartIdx est donné en entrée et le résultat est alloué à refIdxCol et mvCol.

La variable colZeroFlag est déduite comme suit.

– Si toutes les conditions suivantes sont vraies, colZeroFlag est mis égal à 1.
  – RefPicList1[0] est actuellement marquée "utilisée comme référence à court terme".
  – refIdxCol est égal à 0
  – les deux composantes de vecteur cinétique mvCol[0] et mvCol[1] sont dans la gamme de –1 à 1 en unités spécifiées comme suit.
    – si le macrobloc colocalisé est un macrobloc de trame, les unités de mvCol[0] et mvCol[1] sont des unités de quart d'échantillon de trame luma;
    – Sinon (si le macrobloc colocalisé est un macrobloc de sous-trame), les unités de mvCol[0] et mvCol[1] sont des unités de quart d'échantillon de sous-trame luma.

  NOTE 2 – Pour les besoins de détermination de la condition ci-dessus, la valeur mvCol[1] n'est pas normalisée pour une utilisation des unités d'un vecteur cinétique pour le macrobloc en cours dans les cas où le macrobloc en cours est un macrobloc de trame et que le macrobloc colocalisé est un macrobloc de sous-trame ou lorsque le macrobloc en cours est un macrobloc de sous-trame et que le macrobloc colocalisé est un macrobloc de trame. Cet aspect diffère de l'utilisation de mvCol[1] dans le mode direct temporel comme spécifié au § 8.4.1.2.3, qui applique la normalisation au vecteur cinétique du macrobloc colocalisé afin d'utiliser les mêmes unités que celles d'un vecteur cinétique pour le macrobloc en cours, utilisant dans ces cas l'équation 8-190 ou l'équation 8-191.

  – Sinon, colZeroFlag est mis égal à 0.

Les vecteurs cinétiques mvLX (X étant 0 ou 1) sont déduits comme suit.

– Si l'une des conditions suivantes est vraie, les deux composantes du vecteur cinétique mvLX sont mises égales à 0.
  – directZeroPredictionFlag est égal à 1
  – refIdxLX est inférieur à 0
  – refIdxLX est égal à 0 et colZeroFlag est égal à 1
– Sinon, le processus spécifié au § 8.4.1.3 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, refIdxLX et currSubMbType en entrées et le résultat est alloué à mvLX.

NOTE 3 – Le vecteur cinétique mvLXrenvoyé par la procédure du § 8.4.1.3 est identique pour toutes les subdivisions 4x4 de sous-macrobloc d'un macrobloc faisant l'objet de l'invocation du processus.

Les fanions d'utilisation de prédiction predFlagL0 et predFlagL1 sont déduits comme spécifié, au moyen du Tableau 8-9.

Tableau 8-9 – Allocation des fanions d'utilisation de prédiction

refIdxL0	refIdxL1	predFlagL0	predFlagL1
>= 0	>= 0	1	1
>= 0	< 0	1	0
< 0	>= 0	0	1

La variable subMvCnt est calculée comme suit.

– Si subMbPartIdx est différent de 0 ou direct_8x8_inference_flag est égal à 0, subMvCnt est mis égal à 0.
– Sinon (si subMbPartIdx est égal à 0 et si direct_8x8_inference_flag est égal à 1), subMvCnt est mis égal à predFlagL0 + predFLagL1.

8.4.1.2.3 Processus de déduction pour mode de prédiction direct temporel de vecteur cinétique et d'indice de référence luma

Ce processus est invoqué lorsque direct_spatial_mv_pred_flag est égal à 0 et si une des conditions suivantes est vraie:

– mb_type est égal à B_Skip
– mb_type est égal à B_Direct_16x16
– sub_mb_type[mbPartIdx] est égal à B_Direct_8x8.

Les entrées dans ce processus sont mbPartIdx et subMbPartIdx.

Les résultats de ce processus sont les vecteurs cinétiques mvL0 et mvL1, les indices de référence refIdxL0 et refIdxL1 et les fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1.

Le processus spécifié au § 8.4.1.2.1 est invoqué avec mbPartIdx, subMbPartIdx est donné en entrée et le résultat est alloué à colPic, mbAddrCol, mvCol, refIdxCol et vertMvScale.

Les indices de référence refIdxL0 et refIdxL1 sont déduits comme suit.

$$refIdxL0 = ((refIdxCol < 0) ? 0 : MapColToList0(refIdxCol)) \quad (8\text{-}188)$$

$$refIdxL1 = 0 \quad (8\text{-}189)$$

NOTE 1 – Si le macrobloc en cours est un macrobloc de sous-trame, refIdxL0 et refIdxL1 sont les indices d'une liste de sous-trames; Sinon (si le macrobloc en cours est un macrobloc de trame), refIdxL0 et refIdxL1 sont les indices d'une liste de trames ou paire de sous-trames de référence complémentaires.

Soit refPicCol une trame, une sous-trame ou une paire de sous-trames complémentaires qui sont désignées par l'indice de référence refIdxCol lors du décodage du macrobloc colocalisé mbAddrCol au sein de l'image colPic. La fonction MapColToList0(refIdxCol) est spécifiée comme suit:

– Si vertMvScale est égal à One_To_One, on applique ce qui suit.
  – Si field_pic_flag est égal à 0 et si le macrobloc en cours est un macrobloc de sous-trame, on applique ce qui suit.

– Soit refIdxL0Frm la plus faible valeur d'indice de référence dans la liste d'images de référence en cours RefPicList0 qui mentionne la trame ou paire desous-trames complémentaires qui contient la sous-trame refPicCol. RefPicList0 doit contenir une trame ou paire de sous-trames complémentaires qui contient refPicCol. La valeur retournée de MapColToList0( ) est spécifiée comme suit.

  – Si la sous-trame à laquelle se rapporte refIdxCol a la même parité que le macroblocen cours, MapColToList0(refIdxCol) retourne l'indice de référence (refIdxL0Frm << 1).

  – Sinon (si la sous-trame à laquelle se rapporte refIdxCol a la parité opposée à celle du macrobloc en cours), MapColToList0(refIdxCol) retourne l'indice de référence ((refIdxL0Frm << 1) + 1).

– Sinon (si field_pic_flag est égal à 1 ou le macrobloc en cours soit un macrobloc de trame), MapColToList0(refIdxCol) retourne la plus faible valeur d'indice de référence refIdxL0 dans la liste d'images de référence en cours RefPicList0 qui mentionne refPicCol. RefPicList0 doit contenir refPicCol.

– Sinon, si vertMvScale est égal à Frm_To_Fld, on applique ce qui suit.

  – Si field_pic_flag est égal à 0, soit refIdxL0Frm la plus faible valeur d'indice de référence dans la liste d'images de référence en cours RefPicList0 qui mentionne refPicCol. MapColToList0(refIdxCol) retourne l'indice de référence (refIdxL0Frm << 1). RefPicList0 doit contenir refPicCol.

  – Sinon (si field_pic_flag est égal à 1), MapColToList0(refIdxCol) retourne la plus faible valeur d'indice de référence refIdxL0 dans la liste d'images de référence en cours RefPicList0 qui mentionne lla sous-trame de refPicCol avec la même parité que l'image en cours CurrPic. RefPicList0 doit contenir la sous-trame de refPicCol avec la même parité que l'image en cours CurrPic.

– Sinon (si vertMvScale est égal à Fld_To_Frm), MapColToList0(refIdxCol) retourne la plus faible valeur d'indice de référence refIdxL0 dans la liste d'images de référence en cours RefPicList0 qui mentionne la trame ou paire de sous-trames complémentaires qui contient refPicCol. RefPicList0 doit contenir une trame ou paire de sous-trames complémentaires qui contient la sous-trame refPicCol.

NOTE 2 – Une image décodée de référence, qui était marquée "utilisée comme référence à court terme" lorsqu'elle était mentionnée dans le processus de décodage de l'image contenant le macrobloc colocalisé, peut avoir été modifiée afin d'être marquée "utilisée comme référence à long terme" avant d'être utilisée comme référence pour l'interprédiction utilisant le mode de prédiction direct pour le macrobloc en cours.

En fonction de la valeur de vertMvScale, la composante verticale de mvCol est modifiée comme suit.

– Si vertMvScale est égal à Frm_To_Fld

$$mvCol[1] = mvCol[1] / 2 \quad (8\text{-}190)$$

– Sinon, si vertMvScale est égal à Fld_To_Frm

$$mvCol[1] = mvCol[1] * 2 \quad (8\text{-}191)$$

– Sinon (si vertMvScale est égal à One_To_One), mvCol[1] reste inchangée.

Les variables currPicOrField, pic0 et pic1, sont calculées comme suit:

– Si field_pic_flag est égal à 0 et si le macrobloc en cours est un macrobloc de sous-trame, on applique ce qui suit.

  – currPicOrField est la sous-trame de l'image en cours CurrPic qui a la même parité que le macrobloc en cours.

  – pic1 est la sous-trame de RefPicList1[0] qui a la même parité que le macrobloc en cours.

  – La variable pic0 est calculée comme suit.

    – Si refIdxL0 % 2 est égal à 0, pic0 est la sous-trame de RefPicList0[refIdxL0 / 2] qui a la même parité que le macrobloc en cours.

    – Sinon (si refIdxL0 % 2 est différent de 0), pic0 est la sous-trame de RefPicList0[refIdxL0 / 2] qui a la parité opposée à celle du macrobloc en cours.

– Sinon (si field_pic_flag est égal à 1 ou que le macrobloc en cours soit un macrobloc de trame), currPicOrField est l'image en cours CurrPic, pic1 est l'image de référence décodée RefPicList1[0], et pic0 est l'image de référence décodée RefPicList0[refIdxL0].

Les deux vecteurs cinétiques mvL0 et mvL1 sont déduits comme suit pour chaque subdivision 4x4 de sous-macrobloc du macrobloc en cours.

NOTE 3 – Il arrive fréquemment que de nombreuses subdivisions 4x4 de sous-macroblocs partagent les mêmes vecteurs cinétiques et images de référence. Dans ces cas, la compensation de mouvement de mode direct temporel peut calculer les valeurs d'échantillon d'interprédiction dans des unités plus grandes que les blocs d'échantillon luma 4x4. Par exemple, lorsque direct_8x8_inference_flag est égal à 1, au moins chaque quadrant d'échantillon 8x8 luma du macrobloc partage les mêmes vecteurs cinétiques et images de référence.

– Si l'indice de référence refIdxL0 se rapporte à une image de référence à long terme ou si DiffPicOrderCnt(pic1, pic0) est égal à 0, les vecteurs cinétiques mvL0, mvL1 pour la subdivision de mode direct sont déduits par

$$mvL0 = mvCol \tag{8-192}$$

$$mvL1 = 0 \tag{8-193}$$

– Sinon, les vecteurs cinétiques mvL0, mvL1 sont déduits comme versions normalisées du vecteur cinétique mvCol de la subdivision colocalisée de sous-macrobloc comme spécifié ci-dessous (voir la Figure 8-2)

$$tx = (16\,384 + Abs(td / 2)) / td \tag{8-194}$$

$$DistScaleFactor = Clip3(-1024, 1023, (tb * tx + 32) >> 6) \tag{8-195}$$

$$mvL0 = (DistScaleFactor * mvCol + 128) >> 8 \tag{8-196}$$

$$mvL1 = mvL0 – mvCol \tag{8-197}$$

où tb et td sont calculés comme suit.

$$tb = Clip3(-128, 127, DiffPicOrderCnt(currPicOrField, pic0)) \tag{8-198}$$

$$td = Clip3(-128, 127, DiffPicOrderCnt(pic1, pic0)) \tag{8-199}$$

NOTE 4 – mvL0 et mvL1 ne peuvent pas excéder les gammes spécifiées à l'Annexe A.

Les fanions d'utilisation de prédiction predFlagL0 et predFlagL1 sont tous deux mis égaux à 1.

La Figure 8-2 illustre la déduction du vecteur cinétique temporel en mode direct lorsque l'image en cours est temporellement entre l'image de référenceissue de la liste 0 d'images de référence et l'image de référence issue de la liste 1 d'images de référence.

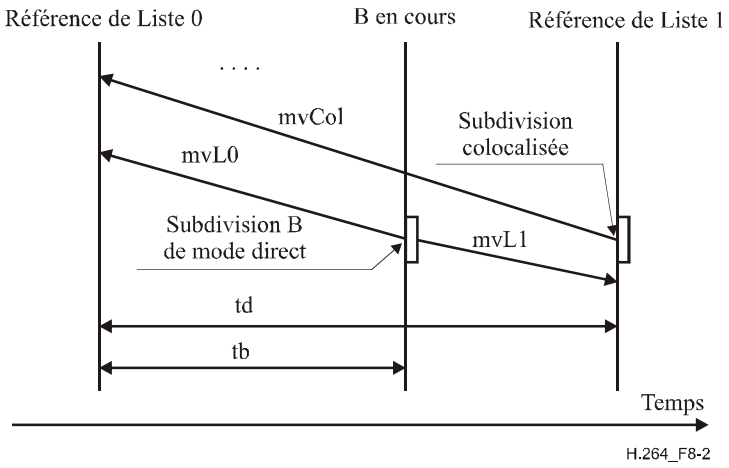


Figure 8-2 – Exemple de déduction de vecteur cinétique temporel en mode direct (pour information)

8.4.1.3 Processus de déduction pour la prédiction de vecteur cinétique luma

Les entrées dans ce processus sont:

– l'indice de subdivision de macrobloc mbPartIdx;

– l'indice de subdivision de sous-macrobloc subMbPartIdx;

– l'indice de référence de la subdivision en cours refIdxLX (avec X égal à 0 ou 1);

– la variable currSubMbType.

Le résultat de ce processus est la prédiction mvpLX du vecteur cinétique mvLX (avec X égal à 0 ou 1).

Le processus de déduction pour les blocs du voisinage pour les données de mouvement du § 8.4.1.3.2 est invoqué avec mbPartIdx, subMbPartIdx, currSubMbType etlistSuffixFlag = X (X étant égal à 0 ou 1 pour refIdxLX égal, selon le cas, à refIdxL0 ou refIdxL1) comme entrées et avec mbAddrN\mbPartIdxN\subMbPartIdxN, les indices de référence refIdxLXN et les vecteurs cinétiques mvLXN avec N remplacé par A, B ou C comme résultat.

Le processus de déduction pour la prédiction de vecteur cinétique luma médian du § 8.4.1.3.1 est invoqué avec mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN avec N remplacé par A, B ou C et refIdxLX comme entrée et mvpLX comme résultat, à moins qu'une des conditions suivantes ne soit vraie:

– MbPartWidth(mb_type) est égal à 16, MbPartHeight(mb_type) est égal à 8, mbPartIdx est égal à 0 et refIdxLXB est égal à refIdxLX,

$$mvpLX = mvLXB \tag{8-200}$$

– MbPartWidth(mb_type) est égal à 16, MbPartHeight(mb_type) est égal à 8, mbPartIdx est égal à 1 et refIdxLXA est égal à refIdxLX,

$$mvpLX = mvLXA \tag{8-201}$$

– MbPartWidth(mb_type) est égal à 8, MbPartHeight(mb_type) est égal à 16, mbPartIdx est égal à 0 et refIdxLXA est égal à refIdxLX,

$$mvpLX = mvLXA \tag{8-202}$$

– MbPartWidth(mb_type) est égal à 8, MbPartHeight(mb_type) est égal à 16, mbPartIdx est égal à 1 et refIdxLXC est égal à refIdxLX,

$$mvpLX = mvLXC \tag{8-203}$$

La Figure 8-3 illustre la prédiction non médiane comme décrit ci-dessus.

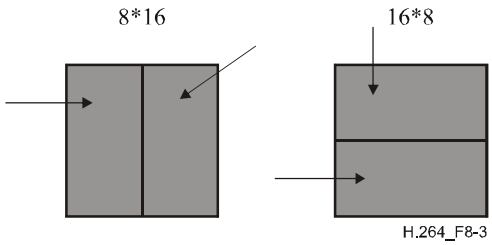


Figure 8-3 – Prédiction de segmentation directionnelle (pour information)

8.4.1.3.1 Processus de déduction pour la prédiction de vecteur cinétique luma médian

Les entrées dans ce processus sont:

– les subdivisions voisines mbAddrN\mbPartIdxN\subMbPartIdxN (avec N remplacé par A, B ou C);

– les vecteurs cinétiques mvLXN (avec N remplacé par A, B ou C) des subdivisions voisines;

– les indices de référence refIdxLXN (avec N remplacé par A, B ou C) des subdivisions voisines;

– l'indice de référence refIdxLX de la subdivision en cours.

Le résultat de ce processus est la prédiction de vecteur cinétique mvpLX.

La variable mvpLX est déduite comme suit:

– Lorsque les deux subdivisions mbAddrB\mbPartIdxB\subMbPartIdxB et mbAddrC\mbPartIdxC\subMbPartIdxC sont indisponibles et que mbAddrA\mbPartIdxA\subMbPartIdxA est disponible

$$mvLXB = mvLXA \tag{8-204}$$

$$mvLXC = mvLXA \tag{8-205}$$

$$refIdxLXB = refIdxLXA \tag{8-206}$$

$$refIdxLXC = refIdxLXA \tag{8-207}$$

– En fonction des indices de référence refIdxLXA, refIdxLXB ou refIdxLXC, on applique ce qui suit.

  – Si un et seulement un des indices de référence refIdxLXA, refIdxLXB ou refIdxLXC est égal à l'indice de référence refIdxLX de la subdivision en cours, on applique ce qui suit. Soit refIdxLXN l'indice de référence qui est égal à refIdxLX, le vecteur cinétique mvLXN est alloué à la prédiction de vecteur cinétique mvpLX:

$$mvpLX = mvLXN \tag{8-208}$$

  – Sinon, chaque composant de la prédiction de vecteur cinétique mvpLX est donné par la médiane des composantes mvLXA, mvLXB et mvLXC du vecteur correspondant du vecteur cinétique:

$$mvpLX[\ 0\ ] = Median(\ mvLXA[\ 0\ ], mvLXB[\ 0\ ], mvLXC[\ 0\ ]\ ) \tag{8-209}$$

$$mvpLX[\ 1\ ] = Median(\ mvLXA[\ 1\ ], mvLXB[\ 1\ ], mvLXC[\ 1\ ]\ ) \tag{8-210}$$

8.4.1.3.2 Processus de déduction pour les données de mouvement des subdivisions du voisinage

Les entrées pourentrées dans ce processus sont:

– l'indice de subdivision de macrobloc mbPartIdx;

– l'indice de subdivision de sous macrobloc subMbPartIdx;

– le type de sous macrobloc en cours currSubMbType;

– le fanion de suffixe de liste LX listSuffixFlag.

Les résultats de ce processus sont (avec N remplacé par A, B ou C)

– mbAddrN\mbPartIdxN\subMbPartIdxN spécifiant les subdivisions du voisinage;

– les vecteurs cinétiques mvLXN des subdivisions du voisinage;

– les indices de référence refIdxLXN des subdivisions du voisinage.

Les noms de variable qui contiennent la chaîne "LX" sont interprétés comme si le X était égal à listSuffixFlag.

Les subdivisions mbAddrN\mbPartIdxN\subMbPartIdxN avec N étant A, B ou C sont déduites selon les étapes ordonnées suivantes:

1. Soit mbAddrD\mbPartIdxD\subMbPartIdxD des variables spécifiant une subdivision supplémentaire du voisinage.
2. Le processus du § 6.4.8.5 est invoqué avec mbPartIdx, currSubMbType et subMbPartIdx comme entrées et le résultat est alloué à mbAddrN\mbPartIdxN\subMbPartIdxN avec N remplacé par A, B, C ou D.
3. Lorsque la subdivision mbAddrC\mbPartIdxC\subMbPartIdxC n'est pas disponible, on applique ce qui suit:

$$mbAddrC = mbAddrD \tag{8-211}$$

mbPartIdxC = mbPartIdxD (8-212)

subMbPartIdxC = subMbPartIdxD (8-213)

Les vecteurs cinétiques mvLXN et les indices de référence refIdxLXN (avec N étant A, B ou C) sont déduits comme suit.

– Si la subdivision de macrobloc ou subdivision de sous macrobloc mbAddrN\mbPartIdxN\subMbPartIdxN n'est pas disponible ou que mbAddrN est codé en mode d'intraprédiction ou que predFlagLX de mbAddrN\mbPartIdxN\subMbPartIdxN est égal à 0, les deux composantes de mvLXN sont mises égales à 0 et refIdxLXN est mis égal à –1.

– Sinon, on applique ce qui suit.

  – Le vecteur cinétique mvLXN et l'indice de référence refIdxLXN sont respectivement mis égaux à MvLX[mbPartIdxN][subMbPartIdxN] et RefIdxLX[mbPartIdxN], qui sont le vecteur cinétique mvLX et l'indice de référence refIdxLX qui ont été alloués à la subdivision de (sous-)macrobloc mbAddrN\mbPartIdxN\subMbPartIdxN.

  – Les variables mvLXN[1] et refIdxLXN sont ensuite traitées comme suit.

    – Si le macrobloc en cours est un macrobloc de sous-trame et que le macrobloc mbAddrN soit un macrobloc de trame

      mvLXN[1] = mvLXN[1] / 2 (8-214)

      refIdxLXN = refIdxLXN * 2 (8-215)

    – Sinon, si le macrobloc en cours est un macrobloc de trame et que le macrobloc mbAddrN soit un macrobloc de sous-trame

      mvLXN[1] = mvLXN[1] * 2 (8-216)

      refIdxLXN = refIdxLXN / 2 (8-217)

    – Sinon, la composante verticale de vecteur cinétique mvLXN[1] et l'indice de référence refIdxLXN restent inchangés.

8.4.1.4 Processus de déduction pour des vecteurs cinétiques chroma

Ce processus n'est appelé que lorsque chroma_format_idc est différent de 0 (monochrome).

Les entrées dans ce processus sont un vecteur cinétique luma mvLX et un indice de référence refIdxLX.

Le résultat de ce processus est un vecteur cinétique chroma mvCLX.

Un vecteur cinétique chroma est calculé à partir du vecteur cinétique luma correspondant.

La précision des composantes du vecteur cinétique chroma est $1 \div (4 * \text{SubWidthC})$ horizontalement et $1 \div (4 * \text{SubHeightC})$ verticalement.

NOTE – Par exemple, lorsque l'on utilise le format chromatique 4:2:0, étant donné que les unités des vecteurs cinétiques luma sont un quart des unités d'échantillon luma et que les échantillons chroma ont une résolution horizontale et verticale de moitié par rapport aux échantillons luma, les unités de vecteurs cinétiques chroma sont un huitième des unités d'échantillons chroma, c'est-à-dire qu'une valeur de 1 pour le vecteur cinétique chroma se rapporte à un déplacement d'un huitième d'échantillon chroma. Par exemple, lorsque le vecteur luma s'applique à 8x16 échantillons luma, le vecteur chroma correspondant en format chromatique 4:2:0 s'applique à 4x8 échantillons chroma et lorsque le vecteur luma s'applique à 4x4 échantillons luma, le vecteur chroma correspondant en format chromatique 4:2:0 s'applique à 2x2 échantillons chroma.

Pour la déduction du vecteur cinétique mvCLX, on applique ce qui suit.

– Si chroma_format_idc n'est pas égal à 1 ou que le macrobloc actuel soit un macrobloc de trames, les composantes horizontale et verticale du vecteur cinétique chroma mvCLX sont des variables calculées comme suit:

mvCLX[0] = mvLX[0] (8-218)
mvCLX[1] = mvLX[1] (8-219)

– Sinon (si , lorsque chroma_format_idc est égal à 1 et que le macrobloc en cours soit un macrobloc de sous-trame), seule la composante horizontale du vecteur cinétique chroma mvCLX[0] est déduite au moyen de l'équation 8-218. La composante verticale du vecteur cinétique chroma mvCLX[1] dépend de la parité de la sous-trame en cours ou du macrobloc en cours et l'image de référence, qui est désignée par l'indice de référence refIdxLX. mvCLX[1] est déduite de mvLX[1] conformément au Tableau 8-10.

Tableau 8-10 – Déduction de la composante verticale du vecteur chroma dans le mode de codage de sous-trame

Conditions de parité		**mvCLX[1]**
Image de référence (refIdxLX)	Sous-trame en cours (image/macrobloc)	
Sous-trame supérieure	Sous-trame inférieure	mvLX[1] + 2
Sous-trame inférieure	Sous-trame supérieure	mvLX[1] – 2
Sinon		mvLX[1]

8.4.2 Processus de décodage pour échantillons d'interprédiction

Les entrées dans ce processus sont

– une subdivision de macrobloc mbPartIdx,

– une subdivision de sous macrobloc subMbPartIdx

– les variables spécifiant partition la largeur et la hauteur des échantillons luma et chroma (si disponibles), partWidth, partHeight, partWidthC (si disponibles) et partHeightC (si disponible)

– les vecteurs cinétiques luma mvL0 et mvL1 et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), les vecteurs cinétiques chroma mvCL0 et mvCL1

– des indices de référence refIdxL0 et refIdxL1

– des fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1

Les résultats de ce processus sont

– les échantillons d'interprédiction predPart, qui sont une matrice (partWidth)x(partHeight) $predPart_L$ d'échantillons luma de prédiction et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), deux matrices (partWidthC)x(partHeightC) $predPart_{Cb}$, $predPart_{Cr}$ d'échantillons chroma de prédiction, une pour chacune des composantes chroma Cb et Cr.

Soient $predPartL0_L$ et $predPartL1_L$ des matrices (partWidth)x(partHeight) pour les valeurs d'échantillon luma prédits et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), $predPartL0_{Cb}$, $predPartL1_{Cb}$, $predPartL0_{Cr}$, et $predPartL1_{Cr}$ les matrices (partWidthC)x(partHeightC) des valeurs d'échantillon chroma prédits.

Pour LX remplacé par L0 ou L1 dans les variables predFlagLX, RefPicListX, refIdxLX, refPicLX, predPartLX, on spécifie ce qui suit.

Lorsque predFlagLX est égal à 1, on applique ce qui suit.

– L'image de référence consistant en une matrice ordonnée à deux dimensions $refPicLX_L$ d'échantillons luma et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), deux matrices ordonnées à deux dimensions $refPicLX_{Cb}$ et $refPicLX_{Cr}$ d'échantillons chroma est déduite en invoquant le processus spécifié au § 8.4.2.1 avec refIdxLX et RefPicListX donnés en entrée.

– La matrice $predPartLX_L$ et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), les matrices $predPartLX_{Cb}$ et $predPartLX_{Cr}$ sont déduites en invoquant le processus spécifié au § 8.4.2.2 avec la subdivision en cours spécifiée par mbPartIdx\subMbPartIdx, les vecteurs cinétiques mvLX, mvCLX, et les matrices de référence avec $refPicLX_L$, $refPicLX_{Cb}$ (si disponible), et $refPicLX_{Cr}$ (si disponible) données en entrée.

Pour C remplacé par L, Cb (si disponible), ou Cr (si disponible), la matrice $predPart_C$ des échantillons de prédiction de la composante C est déduite en invoquant le processus spécifié au § 8.4.2.3 avec la subdivision en cours spécifiée par mbPartIdx et subMbPartIdx et les matrices $predPartL0_C$ et $predPartL1_C$ ainsi que predFlagL0 et predFlagL1 données en entrée.

8.4.2.1 Processus de sélection de l'image de référence

L'entrée dans ce processus est un indice de référence refIdxLX.

Le résultat de ce processus est une image de référence consistant en une matrice à deux dimensions d'échantillons luma $refPicLX_L$ et deux matrices à deux dimensions d'échantillons chroma $refPicLX_{Cb}$ et $refPicLX_{Cr}$.

Selon field_pic_flag, la liste d'images de référence RefPicListX (qui a été déduite comme spécifié au § 8.2.4) se compose de ce qui suit.

– Si field_pic_flag est égal à 1, chaque entrée de RefPicListX est une sous-trame de référence ou une sous-trame d'une trame de référence.

– Sinon (si field_pic_flag est égal à 0), chaque entrée de RefPicListX est une trame de référence ou une paire de sous-trames de référence complémentaires.

Pour la déduction de l'image de référence, on applique ce qui suit.

– Si field_pic_flag est égal à 1, la sous-trame de référence ou la sous-trame d'une trame de référence RefPicListX[refIdxLX] est la sortie. La sous-trame de référence ou sous-trame d'une trame de référence en sortie se compose d'une matrice ($PicWidthInSamples_L$)x($PicHeightInSamples_L$) d'échantillons luma $refPicLX_L$ et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), de deux matrices ($PicWidthInSamples_C$)x($PicHeightInSamples_C$) d'échantillons chroma $refPicLX_{Cb}$ et $refPicLX_{Cr}$.

– Sinon (si field_pic_flag est égal à 0), ce qui suit est applicable.

  – Si le macrobloc actuel est un macrobloc de trames, la trame de référence ou paire de sous-trames de référence complémentaires RefPicListX[refIdxLX] est la sortie. La trame de référence ou paire de sous-trames de référence complémentaires en sortie se compose d'une matrice ($PicWidthInSamples_L$)x($PicHeightInSamples_L$) d'échantillons luma $refPicLX_L$ et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), de deux matrices ($PicWidthInSamples_C$)x($PicHeightInSamples_C$) d'échantillons chroma $refPicLX_{Cb}$ et $refPicLX_{Cr}$.

  – Sinon (si le macrobloc actuel est un sous macrobloc de trames), ce qui suit est applicable:

    – Soit refFrame la trame de référence ou paire de sous-trames de référence complémentaires RefPicListX[refIdxLX / 2].

    – La sous-trame de refFrame est sélectionnée comme suit:

      – Si refIdxLX % 2 est égal à 0, la sous-trame de refFrame qui possède la même parité que le macrobloc actuel est la sortie.

      – Sinon (si refIdxLX % 2 est égal à 1), la sous-trame de refFrame qui possède la parité opposée à celle du macrobloc actuel est la sortie.

    – La sortie de sous-trame de référence ou de sous-trame d'une trame de référence se compose d'une matrice ($PicWidthInSamples_L$)x($PicHeightInSamples_L$ / 2) d'échantillons luma $refPicLX_L$ et, lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), de deux matrices ($PicWidthInSamples_C$)x($PicHeightInSamples_C$ / 2) d'échantillons chroma $refPicLX_{Cb}$ et $refPicLX_{Cr}$.

Les matrices d'échantillons des images de référence $refPicLX_L$, $refPicLX_{Cb}$ (si disponibles), et $refPicLX_{Cr}$ (si disponible) correspondent aux matrices d'échantillons décodés S_L, S_{Cb} (si disponibles), S_{Cr} (si disponible) calculés dans le § 8.7 pour une sous-trame de référence ou trame de référence ou paire de sous-trames de référence complémentaires ou sous-trame d'une trame de référence déjà décodée.

8.4.2.2 Processus d'interpolation d'échantillon fractionnaire

Les entrées dans ce processus sont

– la subdivision en cours donnée par son indice de subdivision mbPartIdx et son indice de subdivision de sous macrobloc subMbPartIdx;

– la largeur partWidth, et la hauteur partHeight de cette subdivision en unités d'échantillon luma;

– un vecteur cinétique luma mvLX donné en unités de quart d'échantillon luma;

– un vecteur cinétique chroma mvCLX donné en unités de huitième d'échantillon chroma;

– les matrices d'échantillons d'image de référence choisies refPicLXL, refPicLXcb, et refPicLXcb.

Les résultats de ce processus sont

– une matrice (partWidth)x(partHeight) $predPartLX_L$ de valeurs d'échantillons luma de prédiction; et

– lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), deux matrices (partWidthC)x(partHeightC) predPartLXcb, et predPartLXcb de valeurs d'échantillons chroma de prédiction.

Soit (xA_L, yA_L) la localisation donnée en unités d'échantillons complets de l'échantillon luma supérieur gauche de la subdivision en cours donnée par mbPartIdx\subMbPartIdx relative à la localisation de l'échantillon luma supérieur gauche de la matrice bidimensionnelle donnée d'échantillons luma.

Soit ($xInt_L$, $yInt_L$) la localisation luma donnée en unités d'échantillons complets et ($xFrac_L$, $yFrac_L$) un décalage donné en unités de quart d'échantillon. Ces variables ne sont utilisées qu'au sein du présent paragraphe afin de spécifier les localisations générales d'échantillon fractionnaire au sein des matrices d'échantillons de référence $refPicLX_L$, $refPicLX_{Cb}$ (si disponibles), et $refPicLX_{Cr}$ (si disponible).

Pour chaque localisation d'échantillon luma ($0 <= x_L < partWidth$, $0 <= y_L < partHeight$) au sein de la matrice de prédiction d'échantillons luma $predPartLX_L$, la valeur correspondante d'échantillon luma de prédiction $predPartLX_L[x_L, y_L]$ est calculée comme suit:

Les variables $xInt_L$, $yInt_L$, $xFrac_L$, et $yFrac_L$ sont calculées par:

$$xInt_L = xA_L + (mvLX[0] >> 2) + x_L \quad (8\text{-}220)$$
$$yInt_L = yA_L + (mvLX[1] >> 2) + y_L \quad (8\text{-}221)$$

$$xFrac_L = mvLX[0] \& 3 \quad (8\text{-}222)$$
$$yFrac_L = mvLX[1] \& 3 \quad (8\text{-}223)$$

– La valeur d'échantillon luma de prédiction $predPartLX_L[x_L, y_L]$ est déduite en invoquant le processus spécifié au § 8.4.2.2.1 avec ($xInt_L$, $yInt_L$), ($xFrac_L$, $yFrac_L$) et $refPicLX_L$ donnés en entrée.

Lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), ce qui suit est applicable.

Soit ($xInt_C$, $yInt_C$) une localisation chroma donnée en unités d'échantillons complets et ($xFrac_C$, $yFrac_C$) un décalage donné en unités d'un huitième d'échantillon. Ces variables ne sont utilisées qu'au sein du présent paragraphe afin de spécifier les localisations générales d'échantillon fractionnaire au sein des matrices d'échantillons de référence $refPicLX_{Cb}$, et $refPicLX_{Cr}$.

Pour chaque localisation d'échantillon chroma ($0 <= x_C < partWidthC$, $0 <= y_C < partHeightC$) au sein des matrices d'échantillons chroma de prédiction $predPartLX_{Cb}$ et $predPartLX_{Cr}$, les valeurs correspondantes d'échantillons chroma de prédiction $predPartLX_{Cb}[x_C, y_C]$ et $predPartLX_{Cr}[x_C, y_C]$ sont calculées comme suit:

– Selon la valeur de chroma_format_idc, les variables $xInt_C$, $yInt_C$, $xFrac_C$, et $yFrac_C$ sont calculées comme suit:

  – Si chroma_format_idc est égal à 1,

$$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 3) + x_C \quad (8\text{-}224)$$
$$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 3) + y_C \quad (8\text{-}225)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8\text{-}226)$$
$$yFrac_C = mvCLX[1] \& 7 \quad (8\text{-}227)$$

  – Sinon, si chroma_format_idc est égal à 2,

$$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 3) + x_C \quad (8\text{-}228)$$
$$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 2) + y_C \quad (8\text{-}229)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8\text{-}230)$$
$$yFrac_C = (mvCLX[1] \& 3) << 1 \quad (8\text{-}231)$$

  – Sinon (si chroma_format_idc est égal à 3),

$$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 2) + x_C \quad (8\text{-}232)$$
$$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 2) + y_C \quad (8\text{-}233)$$

$$xFrac_C = (mvCLX[0] \& 3) << 1 \quad (8\text{-}234)$$
$$yFrac_C = (mvCLX[1] \& 3) << 1 \quad (8\text{-}235)$$

– La valeur d'échantillon de prédiction $predPartLX_{Cb}[x_C, y_C]$ est déduite en invoquant le processus spécifié au § 8.4.2.2.2 avec ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) et $refPicLX_{Cb}$ donnés en entrée.

– La valeur d'échantillon de prédiction $predPartLX_{Cr}[x_C, y_C]$ est calculée en invoquant le processus spécifié au § 8.4.2.2.2 avec ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) et $refPicLX_{Cr}$ donnés en entrée.

8.4.2.2.1 Processus d'interpolation d'échantillon luma

Les entrées dans ce processus sont

– une localisation luma en unités d'échantillon complet ($xInt_L$, $yInt_L$),

– un décalage de localisation luma en unités d'échantillons fractionnaires ($xFrac_L$, $yFrac_L$),

– la matrice d'échantillons luma de l'image de référence choisie $refPicLX_L$,

Le résultat de ce processus est une valeur prédite d'échantillon luma $predPartLX_L[x_L, y_L]$.

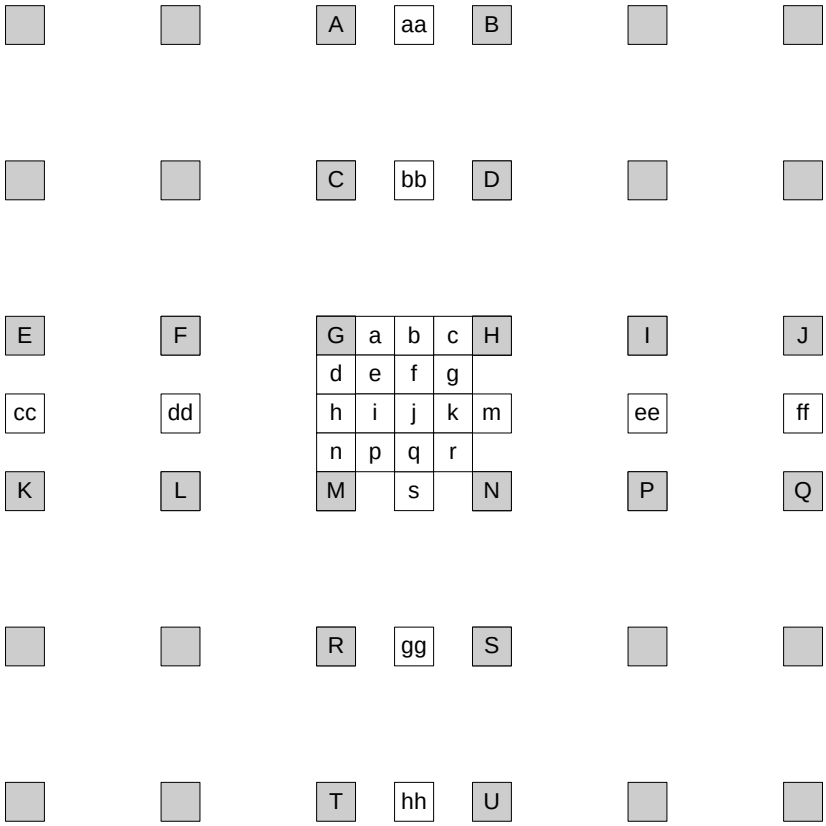


Figure 8-4 – Echantillons entiers (blocs ombrés avec lettres majuscules) et positions d'échantillons fractionnaires (blocs non ombrés avec lettres minuscules) pour interpolation de quarts d'échantillon luma

La variable $refPicHeightEffective_L$, qui est la hauteur de la matrice luma d'image de référence effective, est calculée comme suit.

– Si MbaffFrameFlag est égal à 0 ou mb_field_decoding_flag est égal à 0, $refPicHeightEffective_L$ est mis égal à $PicHeightInSamples_L$.

– Sinon (si MbaffFrameFlag est égal à 1 et si mb_field_decoding_flag est égal à 1), $refPicHeightEffective_L$ est mis égal à $PicHeightInSamples_L$ / 2.

A la Figure 8-4, les positions désignées avec des lettres minuscules au sein de blocs ombrés représentent des échantillons luma à des localisations d'échantillons entiers à l'intérieur de la matrice bidimensionnelle donnée $refPicLX_L$ d'échantillons luma. Ces échantillons peuvent être utilisés afin de produire la valeur d'échantillon luma prédite $predPartLX_L[x_L, y_L]$. Les localisations (xZ_L, yZ_L) pour chacun des échantillons luma correspondants Z, où Z peut être A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, ou U, au sein de la matrice donnée $refPicLX_L$ d'échantillons luma sont déduites comme suit:

$$xZ_L = \text{Clip3}(0, \text{PicWidthInSamples}_L - 1, xInt_L + xDZ_L)$$
$$yZ_L = \text{Clip3}(0, \text{refPicHeightEffective}_L - 1, yInt_L + yDZ_L) \quad (8\text{-}236)$$

Le Tableau 8-11 spécifie (xDZ_L, yDZ_L) pour différents remplacements de Z.

Tableau 8-11 – Localisation différentielle d'échantillons luma entiers

Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	Q	R	S	T	U
xDZ_L	0	1	0	1	-2	-1	0	1	2	3	-2	-1	0	1	2	3	0	1	0	1
yDZ_L	-2	-2	-1	-1	0	0	0	0	0	0	1	1	1	1	1	1	2	2	3	3

Etant donné les échantillons luma 'A' à 'U' à des localisations d'échantillons entiers (xA_L, yA_L) à (xU_L, yU_L), les échantillons luma 'a' à 's' à des positions d'échantillons fractionnaires sont déduits par les règles suivantes. Les valeurs de prédiction luma à des positions de demi échantillon sont déduites par application d'un filtre à 6 branches avec des valeurs de branche (1, –5, 20, 20, –5, 1). Les valeurs de prédiction luma à des positions de quart d'échantillon sont déduites en intégrant les échantillons à des positions de demi échantillon et d'échantillon entier. Le processus pour chaque position fractionnaire est décrit ci-dessous.

– Les échantillons à des positions de demi échantillon désignées par b sont déduits en calculant d'abord les valeurs intermédiaires notées b_1 par application du filtre à six branches aux échantillons de position entière la plus proche dans la direction horizontale. Les échantillons à des positions de demi échantillon désignées h sont déduits en calculant d'abord les valeurs intermédiaires notées h_1 par application du filtre à six branches aux échantillons de position entière la plus proche dans la direction verticale:

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \quad (8\text{-}237)$$
$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \quad (8\text{-}238)$$

Les valeurs de prédiction finales b et h sont déduites au moyen de:

$$b = \text{Clip1}_Y((b_1 + 16) >> 5) \quad (8\text{-}239)$$
$$h = \text{Clip1}_Y((h_1 + 16) >> 5) \quad (8\text{-}240)$$

– Les échantillons à une position de demi échantillon désignées j sont déduits en calculant d'abord la valeur intermédiaire notée j_1 par application du filtre à six branches aux valeurs intermédiaires des positions de demi échantillon les plus proches dans la direction verticale ou horizontale parce que cela donne un résultat équivalent.

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff, \text{ or} \quad (8\text{-}241)$$
$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh \quad (8\text{-}242)$$

où les valeurs intermédiaires notées aa, bb, gg, s_1 et hh sont déduites en appliquant horizontalement le filtre à six branches de la même façon que pour la déduction de b_1 et les valeurs intermédiaires notées cc, dd, ee, m_1 et ff sont déduites en appliquant verticalement le filtre à six branches de la même façon que pour la déduction de h_1. La valeur de prédiction finale j doit être calculée au moyen de:

$$j = \text{Clip1}_Y((j_1 + 512) >> 10) \quad (8\text{-}243)$$

– Les valeurs de prédiction finales s et m sont déduites de s_1 et m_1 de la même façon que pour la déduction de b et h, telles que donnée par:

$$s = \text{Clip1}_Y((s_1 + 16) >> 5) \quad (8\text{-}244)$$
$$m = \text{Clip1}_Y((m_1 + 16) >> 5) \quad (8\text{-}245)$$

– Les échantillons à des positions de quart d'échantillon désignées par a, c, d, n, f, i, k, et q sont déduits en intégrant avec arrondi supérieur aux deux échantillons les plus proches à des positions d'échantillons entiers et de moitié au moyen de:

$$a = (G + b + 1) >> 1 \quad (8\text{-}246)$$
$$c = (H + b + 1) >> 1 \quad (8\text{-}247)$$

$d = (G + h + 1) >> 1$ (8-248)
$n = (M + h + 1) >> 1$ (8-249)
$f = (b + j + 1) >> 1$ (8-250)
$i = (h + j + 1) >> 1$ (8-251)
$k = (j + m + 1) >> 1$ (8-252)
$q = (j + s + 1) >> 1.$ (8-253)

– Les échantillons à des positions de quart d'échantillon désignées par e, g, p, et r sont déduits en intégrant avec arrondi supérieur aux deux échantillons les plus proches à des positions de demi échantillon dans la direction de la diagonale au moyen de

$e = (b + h + 1) >> 1$ (8-254)
$g = (b + m + 1) >> 1$ (8-255)
$p = (h + s + 1) >> 1$ (8-256)
$r = (m + s + 1) >> 1.$ (8-257)

Le décalage de localisation luma en unités d'échantillons fractionnaires ($xFrac_L$, $yFrac_L$) spécifie lequel des échantillons luma produits à des localisations d'échantillon entier et d'échantillon fractionnaire est alloué à la valeur d'échantillon luma prédite $predPartLX_L[x_L, y_L]$. Cette allocation est faite conformément au Tableau 8-12. La valeur de $predPartLX_L[x_L, y_L]$ est le résultat.

Tableau 8-12 – Allocation de l'échantillon de prédiction luma $predPartLX_L[x_L, y_L]$

$xFrac_L$	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
$yFrac_L$	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$predPartLX_L[x_L, y_L]$	G	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r

8.4.2.2.2 Processus d'interpolation d'échantillon chroma

Ce processus ne doit être invoqué que lorsque chroma_format_idc n'est pas égal à 0 (format monochrome).

Les entrées dans ce processus sont

– une localisation chroma en unités d'échantillon entier ($xInt_C$, $yInt_C$),

– un décalage de localisation chroma en unités d'échantillon fractionnaire ($xFrac_C$, $yFrac_C$),

– des échantillons de composante chromatique provenant de l'image de référence choisie $refPicLX_C$.

Le résultat de ce processus est une valeur prédite d'échantillon chroma $predPartLX_C[x_C, y_C]$.

Dans la Figure 8-5, les positions désignées par A, B, C, et D représentent des échantillons chroma à des localisations d'échantillon entier dans le tableau bidimensionnel donné $refPicLX_C$ d'échantillons chroma.

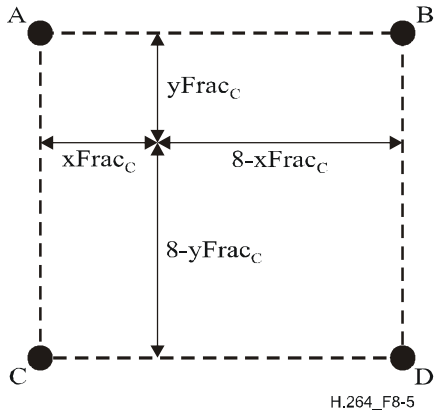


Figure 8-5 – Variables selon la position d'échantillon fractionnaire en interpolation chroma avec les échantillons A, B, C, et D de position entière qui les entourent

La variable $refPicHeightEffective_C$, qui est la hauteur de la matrice chroma d'image de référence effective, est calculée comme suit.

– Si MbaffFrameFlag est égal à 0 ou mb_field_decoding_flag est égal à 0, $refPicHeightEffective_C$ est mis égal à $PicHeightInSamples_C$.

– Sinon (si MbaffFrameFlag est égal à 1 et si mb_field_decoding_flag est égal à 1), $refPicHeightEffective_C$ est mis égal à $PicHeightInSamples_C$ / 2.

Les coordonnées des échantillons spécifiées dans les équations 8-258 à 8-265 sont utilisées afin de produire la valeur d'échantillon chroma prédite $predPartLX_C[x_C, y_C]$.

$$xA_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C) \quad (8\text{-}258)$$
$$xB_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C + 1) \quad (8\text{-}259)$$
$$xC_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C) \quad (8\text{-}260)$$
$$xD_C = Clip3(0, PicWidthInSamples_C - 1, xInt_C + 1) \quad (8\text{-}261)$$

$$yA_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C) \quad (8\text{-}262)$$
$$yB_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C) \quad (8\text{-}263)$$
$$yC_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C + 1) \quad (8\text{-}264)$$
$$yD_C = Clip3(0, refPicHeightEffective_C - 1, yInt_C + 1) \quad (8\text{-}265)$$

Etant donné les échantillons chroma A, B, C, et D aux localisations d'échantillon entier spécifiées dans les équations 8-258 à 8-265, la valeur prédite d'échantillon chroma $predPartLX_C[x_C, y_C]$ est calculée comme suit:

$$predPartLX_C[x_C, y_C] = ((8 - xFrac_C) * (8 - yFrac_C) * A + xFrac_C * (8 - yFrac_C) * B + (8 - xFrac_C) * yFrac_C * C + xFrac_C * yFrac_C * D + 32) >> 6 \quad (8\text{-}266)$$

8.4.2.3 Processus de prédiction d'échantillon pondéré

Les entrées dans ce processus sont

– mbPartIdx: la subdivision en cours donnée par l'indice de subdivision

– subMbPartIdx: l'indice de subdivision de sous-macrobloc

– predFlagL0 et predFlagL1: des fanions d'utilisation de liste de prédiction

– $predPartLX_L$: une matrice (partWidth)x(partHeight) d'échantillons de prédiction luma (avec LX remplacé par L0 ou L1 en fonction de predFlagL0 et predFlagL1)

– lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), $predPartLX_{Cb}$ et $predPartLX_{Cr}$: des matrices (partWidthC)x(partHeightC) d'échantillons de prédiction chroma, une pour chacun des composantes chroma Cb et Cr (avec LX remplacé par L0 ou L1 en fonction de predFlagL0 et predFlagL1).

Les résultats de ce processus sont

– $predPart_L$: une matrice (partWidth)x(partHeight) d'échantillons de prédiction luma et

– lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), $predPart_{Cb}$, et $predPart_{Cr}$: des matrices (partWidthC)x(partHeightC) d'échantillons de prédiction chroma, une pour chacun des composantes chroma Cb et Cr.

Pour les macroblocs ou les subdivisions avec predFlagL0 égal à 1 dans les tranches P et SP, on applique ce qui suit:

– Si weighted_pred_flag est égal à 0, le processus de prédiction d'échantillon pondéré par défaut comme décrit au § 8.4.2.3.1 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

– Sinon (si weighted_pred_flag est égal à 1), le processus explicite de prédiction pondéré comme décrit au § 8.4.2.3.2 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

Pour les macroblocs ou partitions avec predFlagL0 ou predFlagL1 égal à 1 dans les tranches B, on applique ce qui suit:

– Si weighted_bipred_idc est égal à 0, le processus de prédiction d'échantillon pondéré par défaut comme décrit au § 8.4.2.3.1 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

– Sinon, si weighted_bipred_idc est égal à 1, le processus explicite de prédiction d'échantillon pondéré comme décrit au § 8.4.2.3.2, pour les macroblocs ou partitions avec predFlagL0 ou predFlagL1 égal à 1 avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

– Sinon (si weighted_bipred_idc est égal à 2), on applique ce qui suit:

  – Si predFlagL0 est égal à 1 et predFlagL1 est égal à 1, la prédiction implicite d'échantillon pondéré comme décrit au § 8.4.2.3.2 est invoquée avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

  – Sinon (si predFlagL0 ou predFlagL1 sont égaux à 1 mais pas tous les deux), le processus de prédiction d'échantillon pondéré par défaut comme décrit au § 8.4.2.3.1 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

8.4.2.3.1 Processus de prédiction d'échantillon pondéré par défaut

Les entrées dans ce processus sont les mêmes que spécifié au § 8.4.2.3.

Les résultats de ce processus sont les mêmes que spécifié au § 8.4.2.3.

En fonction de la composante disponible pour laquelle le bloc de prédiction est déduit, on applique ce qui suit.

– Si les valeurs de prédiction d'échantillon luma $predPart_L[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à L, x mis égal à 0 .. partWidth – 1, et y mis égal à 0 .. partHeight – 1.

– Sinon, si les valeurs de prédiction d'échantillon de composante chromatique Cb $predPart_{Cb}[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à Cb, x mis égal à 0 .. partWidthC – 1, et y mis égal à 0 .. partHeightC – 1.

– Sinon (si les valeurs de prédiction d'échantillon de composante chromatique Cr $predPart_{Cr}[x, y]$ sont calculées), on applique ce qui suit avec C mis égal à Cr, x mis égal à 0 .. partWidthC – 1, et y mis égal à 0 .. partHeightC – 1.

Les valeurs d'échantillon de prédiction sont calculées comme suit.

– Si predFlagL0 est égal à 1 et predFlagL1 est égal à 0 pour la subdivision en cours,

$$predPart_C[x, y] = predPartL0_C[x, y] \quad (8\text{-}267)$$

– Sinon, si predFlagL0 est égal à 0 et predFlagL1 est égal à 1 pour la subdivision en cours,

$$predPart_C[x, y] = predPartL1_C[x, y] \quad (8\text{-}268)$$

– Sinon (si predFlagL0 et predFlagL1 sont égaux à 1 pour la subdivision en cours),

$$predPart_C[x, y] = (predPartL0_C[x, y] + predPartL1_C[x, y] + 1) >> 1. \quad (8\text{-}269)$$

8.4.2.3.2 Processus de prédiction d'échantillon pondéré

Les entrées dans ce processus sont les mêmes que spécifié au § 8.4.2.3.

Les résultats de ce processus sont les mêmes que spécifié au § 8.4.2.3.

En fonction de la composante disponible pour laquelle est déduit le bloc de prédiction, on applique ce qui suit:

– Si les valeurs de prédiction d'échantillon luma $predPart_L[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à L, x mis égal à 0 .. partWidth – 1, et y mis égal à 0 .. partHeight – 1.

– Sinon, si les valeurs de prédiction d'échantillon de composante chromatique Cb $predPart_{Cb}[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à Cb, x mis égal à 0 .. partWidthC – 1, et y mis égal à 0 .. partHeightC – 1.

– Sinon (si les valeurs de prédiction d'échantillon de composante chromatique Cr $predPart_{Cr}[x, y]$ sont calculées), on applique ce qui suit avec C mis égal à Cr, x mis égal à 0 .. partWidthC – 1, et y mis égal à 0 .. partHeightC – 1.

Les valeurs d'échantillon de prédiction sont calculées comme suit.

– Si la subdivision mbPartIdx\subMbPartIdx a predFlagL0 égal à 1 et predFlagL1 égal à 0, les valeurs prédites d'échantillon finales $predPart_C[x, y]$ sont calculées par

$$\begin{aligned} &\text{si(logWD >= 1)} \\ &\quad predPart_C[x, y] = Clip1_C(((predPartL0_C[x, y] * w_0 + 2^{logWD - 1}) >> logWD) + o_0) \\ &\text{ou} \\ &\quad predPart_C[x, y] = Clip1_C(predPartL0_C[x, y] * w_0 + o_0) \end{aligned} \tag{8-270}$$

– Sinon, si la subdivision mbPartIdx\subMbPartIdx a predFlagL0 égal à 0 et predFlagL1 égal à 1, les valeurs prédites d'échantillon finales $predPart_C[x, y]$ sont calculées par

$$\begin{aligned} &\text{si(logWD >= 1)} \\ &\quad predPart_C[x, y] = Clip1_C(((predPartL1_C[x, y] * w_1 + 2^{logWD - 1}) >> logWD) + o_1) \\ &\text{ou} \\ &\quad predPart_C[x, y] = Clip1_C(predPartL1_C[x, y] * w_1 + o_1) \end{aligned} \tag{8-271}$$

– Sinon (si la subdivision mbPartIdx\subMbPartIdx a les deux predFlagL0 et predFlagL1 égaux à 1), les valeurs prédites d'échantillon finales $predPart_C[x, y]$ sont calculées par

$$\begin{aligned} predPart_C[x, y] = Clip1_C(((predPartL0_C[x, y] * w_0 + predPartL1_C[x, y] * w_1 + 2^{logWD}) >> \\ (logWD + 1)) + ((o_0 + o_1 + 1) >> 1)) \end{aligned} \tag{8-272}$$

Les variables des déductions ci-dessus pour les échantillons de prédiction sont calculées comme suit.

– Si weighted_bipred_idc est égal à 2 et que le slice_type est égal à B, la prédiction pondérée en mode implicite est utilisée comme suit.

$$logWD = 5 \tag{8-273}$$

$$o_0 = 0 \tag{8-274}$$

$$o_1 = 0 \tag{8-275}$$

et w_0 et w_1 sont déduits comme suit.

– Les variables currPicOrField, pic0 et pic1 sont calculées comme suit:
  – si field_pic_flag est égal à 0 et si le macrobloc en cours est un macrobloc de sous-trame, on applique ce qui suit.
    – currPicOrField est la sous-trame de l'image en cours CurrPic qui a la même parité que le macrobloc en cours.
    – La variable pic0 est calculée comme suit.
      – Si refIdxL0 % 2 est égal à 0, pic0 est la sous-trame de RefPicList0[refIdxL0 / 2] qui possède la même parité que le macrobloc actuel.
      – Sinon (si refIdxL0 % 2 n'est pas égal à 0), pic0 est la sous-trame de RefPicList0[refIdxL0 / 2] qui possède la parité opposée du macrobloc actuel.
    – La variable pic1 est calculée comme suit.
      – Si refIdxL1 % 2 est égal à 0, pic1 est la sous-trame de RefPicList1[refIdxL1 / 2] qui possède la même parité que le macrobloc actuel.
      – Sinon (si refIdxL1 % 2 n'est pas égal à 0), pic1 est la sous-trame de RefPicList1[refIdxL1 / 2] qui possède la parité opposée du macrobloc actuel.
  – Sinon (si field_pic_flag est égal à 1 ou si le macrobloc actuel est un macrobloc de trames), currPicOrField est l'image actuelle CurrPic, pic1 est RefPicList1[refIdxL1], et pic0 est RefPicList0[refIdxL0].
– Les variables tb, td, tx, et DistScaleFactor sont calculées à partir des valeurs de currPicOrField, pic0, pic1 au moyen des équations 8-198, 8-199, 8-194, et 8-195, respectivement.

– Si DiffPicOrderCnt(pic1, pic0) est égal à 0 ou si une des deux variables (ou les deux) pic1 et pic0 est (sont) marquée(s) "utilisée comme référence à long terme" ou si (DistScaleFactor >> 2) < -64 ou (DistScaleFactor >> 2) > 128, w_0 et w_1 sont des variables calculées comme suit:

$$w_0 = 32 \quad (8\text{-}276)$$

$$w_1 = 32 \quad (8\text{-}277)$$

– Sinon,

$$w_0 = 64 - (\text{DistScaleFactor} >> 2) \quad (8\text{-}278)$$

$$w_1 = \text{DistScaleFactor} >> 2 \quad (8\text{-}279)$$

– Sinon (si weighted_pred_flag est égal à 1 dans les tranches P ou SP ou si weighted_bipred_idc est égal à 1 dans les tranches B), la prédiction pondérée de mode explicite est utilisée comme suit.

  – Les variables refIdxL0WP et refIdxL1WP sont déduites comme suit.

    – Si MbaffFrameFlag est égal à 1 et si le macrobloc en cours est un macrobloc de sous-trame

$$\text{refIdxL0WP} = \text{refIdxL0} >> 1 \quad (8\text{-}280)$$

$$\text{refIdxL1WP} = \text{refIdxL1} >> 1 \quad (8\text{-}281)$$

    – Sinon (si MbaffFrameFlag est égal à 0 ou si le macrobloc en cours est un macrobloc de trame),

$$\text{refIdxL0WP} = \text{refIdxL0} \quad (8\text{-}282)$$

$$\text{refIdxL1WP} = \text{refIdxL1} \quad (8\text{-}283)$$

  – Les variables logWD, w_0, w_1, o_0, et o_1 sont déduites comme suit.

    – Si C dans $\text{predPart}_C[x, y]$ est remplacé par L pour les échantillons luma

$$\text{logWD} = \text{luma\_log2\_weight\_denom} \quad (8\text{-}284)$$

$$w_0 = \text{luma\_weight\_l0[refIdxL0WP]} \quad (8\text{-}285)$$

$$w_1 = \text{luma\_weight\_l1[refIdxL1WP]} \quad (8\text{-}286)$$

$$o_0 = \text{luma\_offset\_l0[refIdxL0WP]} * (1 << (\text{BitDepth}_Y - 8)) \quad (8\text{-}287)$$

$$o_1 = \text{luma\_offset\_l1[refIdxL1WP]} * (1 << (\text{BitDepth}_Y - 8)) \quad (8\text{-}288)$$

    – Sinon (si C dans $\text{predPart}_C[x, y]$ est remplacé par Cb ou Cr pour les échantillons chroma, avec iCbCr = 0 pour Cb, iCbCr = 1 pour Cr),

$$\text{logWD} = \text{chroma\_log2\_weight\_denom} \quad (8\text{-}289)$$

$$w_0 = \text{chroma\_weight\_l0[refIdxL0WP][iCbCr]} \quad (8\text{-}290)$$

$$w_1 = \text{chroma\_weight\_l1[refIdxL1WP][iCbCr]} \quad (8\text{-}291)$$

$$o_0 = \text{chroma\_offset\_l0[refIdxL0WP][iCbCr]} * (1 << (\text{BitDepth}_C - 8)) \quad (8\text{-}292)$$

$$o_1 = \text{chroma\_offset\_l1[refIdxL1WP][iCbCr]} * (1 << (\text{BitDepth}_C - 8)) \quad (8\text{-}293)$$

Lorsque la prédiction pondérée en mode explicite est utilisée et que la subdivision mbPartIdx\subMbPartIdx a à la fois predFlagL0 et predFlagL1 égaux à 1, on doit respecter les contraintes suivantes:

$$-128 <= w_0 + w_1 <= ((logWD == 7) ? 127 : 128) \quad (8\text{-}294)$$

NOTE – Pour la prédiction pondérée en mode implicite, les pondérations w_0 et w_1 sont chacune obligatoirement dans la gamme de –64...128 et la contrainte exprimée dans l'équation 8-294, bien que non explicitement imposée, doit toujours être respectée. Pour la prédiction pondérée en mode explicite avec logWD égal à 7, lorqu'une des deux pondérations w_0 ou w_1 est supposée être égale à 128 (du fait que luma_weight_l0_flag, luma_weight_l1_flag, chroma_weight_l0_flag, ou chroma_weight_l1_flag est égal à 0), l'autre pondération (w_1 ou w_0) doit avoir une valeur négative afin que la contrainte exprimée dans l'équation (8-294) se vérifie (et par conséquent l'autre fanion luma_weight_l0_flag, luma_weight_l1_flag, chroma_weight_l0_flag, ou chroma_weight_l1_flag doit être égal à 1).

8.5 Processus de décodage du coefficient de transformée et processus de construction d'image antérieurement au processus de filtrage de démontage de blocs

Les entrées dans ce processus sont Intra16x16DCLevel (si disponible), Intra16x16ACLevel (si disponible), LumaLevel (si disponible), LumaLevel8x8 (si disponible), ChromaDCLevel (si disponible), ChromaACLevel (si disponible), et les matrices d'échantillons d'inter ou d'intraprédiction disponibles pour le macrobloc en cours pour la composante applicable $pred_L$, $pred_{Cb}$, ou $pred_{Cr}$.

NOTE 1 – Lors du décodage d'un macrobloc en mode de prédiction Intra_4x4 (ou Intra_8x8), la composante luma du tableau de prédiction de macrobloc peut n'être pas complet, dans la mesure où pour chaque bloc luma 4x4 (ou 8x8), le processus de prédiction Intra_4x4 (ou Intra_8x8) pour les échantillons luma, comme spécifié au § 8.3.1 ou (§ 8.3.2), et le processus spécifié au présent paragraphe, font l'objet d'une itération.

Les résultats de ce processus sont les matrices d'échantillons construites antérieurement au processus de filtrage de démontage de blocs pour les composantes applicables S'_L, S'_{Cb}, ou S'_{Cr}.

NOTE 2 – Lors du décodage d'un macrobloc en mode de prédiction Intra_4x4 (ou Intra_8x8), la composante luma des matrices d'échantillons construites antérieurement au processus de filtrage de démontage de blocs peut n'être pas complet, dans la mesure où pour chaque bloc luma 4x4 (ou 8x8), le processus de prédiction Intra_4x4 (ou Intra_8x8) pour les échantillons luma, comme spécifié au § 8.3.1 (ou § 8.3.2), et le processus spécifié au présent paragraphe, font l'objet d'une itération.

Le présent paragraphe spécifie le décodage du coefficient de transformée et la construction d'image antérieurement au processus de filtrage de démontage de blocs.

Lorsque le macrobloc en cours est codé P_Skip ou B_Skip, toutes les valeurs de LumaLevel, LumaLevel8x8, ChromaDCLevel, ChromaACLevel sont mises égales à 0 pour le macrobloc en cours.

Lorsque le fanion residual_colour_transform_flag est égal à 1, le processus de transformation chromatique résiduelle spécifié dans le § 8.5.13 est invoqué.

8.5.1 Spécification du processus de décodage de transformée pour les blocs luma 4x4 résiduels

La présente spécification s'applique lorsque le fanion transform_size_8x8_flag est égal à 0.

Lorsque le mode de prédiction du macrobloc en cours n'est pas égal à Intra_16x16, la variable LumaLevel contient les niveaux pour les coefficients de transformée luma. Pour un bloc luma 4x4 indexé par luma4x4BlkIdx = 0..15 sont spécifiées les étapes ordonnées suivantes.

1. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.5, est invoqué avec LumaLevel[luma4x4BlkIdx] comme entrée et le tableau bidimensionnel c comme résultat.

2. Le processus de normalisation et de transformation pour les blocs de 4x4 échantillons résiduels, comme spécifié au § 8.5.10, est invoqué avec c comme entrée et r comme résultat.

3. Lorsque le fanion residual_colour_transform_flag est égal à 1, la variable $R_{Y,ij}$ est mise à r_{ij} avec i, j = 0..3 et ce processus est suspendu jusqu'à l'achèvement complet du processus de transformation chromatique résiduelle spécifié dans le § 8.5.13; achèvement après lequel la variable r_{ij} est mise à $R_{G,ij}$ avec i, j = 0..3. Puis ce processus est poursuivi.

4. La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc est déduite en invoquant le processus inverse de balayage de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO).

5. La matrice 4x4 u avec les éléments u_{ij} pour i, j = 0..3 est déduite avec

$$u_{ij} = Clip1_Y(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}295)$$

Lorsque le fanion qpprime_y_zero_transform_bypass_flag est égal à 1 et que QP'_Y est égal à 0, le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de u_{ij} qui, calculée par l'équation 8-295, ne serait pas égale à $pred_L[xO + j, yO + i] + r_{ij}$.

6. Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.12 est invoqué avec luma4x4BlkIdx et u comme entrées.

8.5.2 Spécification du processus de décodage de transformée pour échantillons luma du mode de prédiction de macrobloc Intra_16x16

Lorsque le mode de prédiction du macrobloc en cours est égal à Intra_16x16, les variables Intra16x16DCLevel et Intra16x16ACLevel contiennent les niveaux pour les coefficients de transformée luma. Le décodage du coefficient de transformée s'effectue selon les étapes ordonnées suivantes:

1. Les coefficients de transformée 4x4 luma DC de tous les blocs luma 4x4 du macrobloc sont décodés.
   a. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.5, est invoqué avec Intra16x16DCLevel comme entrée et la matrice bidimensionnelle c comme résultat.
   b. Le processus de normalisation et de transformation pour les coefficients de transformée luma DC pour le type de macrobloc Intra_16x16, comme spécifié au § 8.5.8, est invoqué avec c comme entrée et dcY comme résultat.
2. Pour un bloc luma 4x4 indexé par luma4x4BlkIdx = 0..15 sont spécifiées les étapes ordonnées suivantes.
   a. La variable lumaList, qui est une liste de 16 entrées, est déduite. La première entrée de lumaList est la valeur correspondante provenant de la matrice dcY. La Figure 8-6 montre l'allocation des indices de la matrice dcY aux luma4x4BlkIdx. Les deux chiffres dans les petits carrés se rapportent aux indices i et j de dcY_{ij}, et les chiffres dans les grands carrés se rapportent à luma4x4BlkIdx.

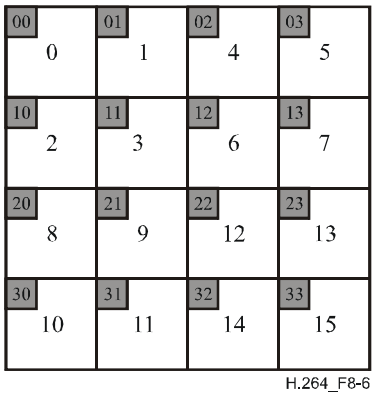


Figure 8-6 – Allocation des indices de dcY à luma4x4BlkIdx

Les éléments de lumaList avec les indices k = 1..15 sont spécifiés comme:

$$lumaList[k] = Intra16x16ACLevel[luma4x4BlkIdx][k - 1] \quad (8\text{-}296)$$

   b. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.5, est invoqué avec lumaList comme entrée et la matrice bidimensionnelle c comme résultat.
   c. Le processus de normalisation et de transformation pour les blocs de 4x4 échantillons résiduels, comme spécifié au § 8.5.10, est invoqué avec c comme entrée et r comme résultat.
   d. Lorsque le fanion residual_colour_transform_flag est égal à 1, la variable $R_{Y,ij}$ est mise à r_{ij} avec i, j = 0..3 et ce processus est suspendu jusqu'à l'achèvement complet du processus de transformation chromatique résiduelle spécifié dans le § 8.5.13, achèvement après lequel la variable r_{ij} est mise à $R_{G,ij}$ avec i, j = 0..3. Puis ce processus est poursuivi.
   e. La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc est déduite en invoquant le processus inverse de balayage de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO).
   f. La matrice 4x4 u avec les éléments u_{ij} pour i, j = 0..3, est déduite comme suit:

$$u_{ij} = Clip1_Y(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}297)$$

Lorsque le fanion qpprime_y_zero_transform_bypass_flag est égal à 1 et que QP'_Y est égal à 0, le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de u_{ij} qui, calculée par l'équation 8-297, ne serait pas égale à $pred_L[xO + j, yO + i] + r_{ij}$.

g. Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.12 est invoqué avec luma4x4BlkIdx, u comme entrées.

8.5.3 Spécification du processus de décodage de transformée pour les blocs de 8x8 échantillons luma résiduels

La présente spécification s'applique lorsque le fanion transform_size_8x8_flag est égal à 1.

La variable LumaLevel8x8[luma8x8BlkIdx] avec luma8x8BlkIdx = 0..3 contient les niveaux pour les coefficients de transformée d'échantillons luma du bloc de 8x8 échantillons luma ayant l'indice luma8x8BlkIdx.

Pour un bloc de 8x8 échantillons luma indexés par luma8x8BlkIdx = 0..3, la séquence des étapes suivantes est spécifiée.

1. Le processus de balayage inverse pour coefficients de transformée de 8x8 échantillons luma comme décrit dans le § 8.5.6 est invoqué avec LumaLevel8x8[luma8x8BlkIdx] comme entrée et la matrice à deux dimensions c comme sortie.
2. Le processus de normalisation et de transformation pour blocs de 8x8 échantillons résiduels spécifié dans le § 8.5.11 est invoqué avec c comme entrée et r comme sortie.
3. Lorsque le fanion residual_colour_transform_flag est égal à 1, la variable $R_{Y,ij}$ est mise à r_{ij} avec i, j = 0..7 et ce processus est suspendu jusqu'à l'achèvement complet du processus de transformation chromatique résiduelle spécifié dans le § 8.5.13, achèvement après lequel la variable r_{ij} est mise à $R_{G,ij}$ avec i, j = 0..7. Puis ce processus est poursuivi.
4. La position de l'échantillon supérieur gauche d'un bloc de 8x8 échantillons luma ayant l'indice luma8x8BlkIdx à l'intérieur du macrobloc est calculé par invocation du processus de balayage inverse de blocs de 8x8 échantillons luma dans le § 6.4.4 avec luma8x8BlkIdx comme entrée et la sortie étant attribuée à (xO, yO).
5. La matrice de 8x8 échantillons u avec éléments u_{ij} pour i, j = 0..7 est calculée comme suit:

$$u_{ij} = Clip1_Y(pred_L[xO + j, yO + i] + r_{ij}) \quad (8\text{-}298)$$

   Lorsque le fanion qpprime_y_zero_transform_bypass_flag est égal à 1 et que QP'_Y est égal à 0, le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de u_{ij} qui, calculée par l'équation 8-298, ne serait pas égale à $pred_L[xO + j, yO + i] + r_{ij}$.

6. Le processus de construction d'image avant processus de filtre de démontage de bloc indiqué dans le § 8.5.12 est invoqué avec luma8x8BlkIdx et u comme entrées.

8.5.4 Spécification du processus de décodage de transformée pour échantillons chroma

Ce processus est invoqué pour chaque composante chromatique Cb et Cr séparément.

Pour chaque composante chromatique, les variables ChromaDCLevel[iCbCr] et ChromaACLevel[iCbCr], avec iCbCr mis égal à 0 pour Cb et iCbCr mis égal à 1 pour Cr, contiennent les niveaux pour les deux composantes des coefficients de transformée chroma.

Soit la variable numChroma4x4Blks mise égale à (MbWidthC / 4) * (MbHeightC / 4).

Pour chaque composante chromatique, le décodage de transformée traite séparément les étapes ordonnées suivantes:

1. On décode les coefficients numChroma4x4Blks de transformée chroma DC des blocs chroma 4x4 de la composante du macrobloc indexé par iCbCr.

   a. Selon la variable chroma_format_idc, ce qui suit est applicable.

   – Si chroma_format_idc est égal à 1, la matrice c 2x2 est déduite au moyen du processus de balayage matriciel inverse appliqué à ChromaDCLevel comme suit:

$$c = \begin{bmatrix} ChromaDCLevel[iCbCr][0] & ChromaDCLevel[iCbCr][1] \\ ChromaDCLevel[iCbCr][2] & ChromaDCLevel[iCbCr][3] \end{bmatrix} \quad (8\text{-}299)$$

– Sinon, si chroma_format_idc est égal à 2, la matrice c 2x4 est déduite au moyen du processus de balayage matriciel inverse appliqué comme suit à ChromaDCLevel:

$$c = \begin{bmatrix} ChromaDCLevel[\,iCbCr\,][\,0\,] & ChromaDCLevel[\,iCbCr\,][\,2\,] \\ ChromaDCLevel[\,iCbCr\,][\,1\,] & ChromaDCLevel[\,iCbCr\,][\,5\,] \\ ChromaDCLevel[\,iCbCr\,][\,3\,] & ChromaDCLevel[\,iCbCr\,][\,6\,] \\ ChromaDCLevel[\,iCbCr\,][\,4\,] & ChromaDCLevel[\,iCbCr\,][\,7\,] \end{bmatrix} \quad (8\text{-}300)$$

– Sinon (si chroma_format_idc est égal à 3), le processus de balayage inverse pour coefficients de transformée spécifié dans le § 8.5.5 est invoqué avec ChromaDCLevel[iCbCr] comme entrée et la matrice 4x4 à deux dimensions c comme sortie.

b. Le processus de normalisation et de transformation pour les coefficients de transformée chroma DC comme spécifié au § 8.5.9 est invoqué avec c comme entrée et dcC comme résultat.

2. Pour chaque bloc chroma 4x4 indexé par chroma4x4BlkIdx = 0..numChroma4x4Blks – 1 de la composante indexée par iCbCr, on spécifie les étapes ordonnées suivantes.

a. On déduit la variable chromaList, qui est une liste de 16 entrées. La première entrée de chromaList est la valeur correspondante provenant de la matrice dcC. La Figure 8-7 montre l'allocation des indices de la matrice dcC à chroma4x4BlkIdx. Les deux chiffres dans les petits carrés se rapportent aux indices i et j dans dcC_{ij}, et les chiffres dans les grands carrés se rapportent à chroma4x4BlkIdx.

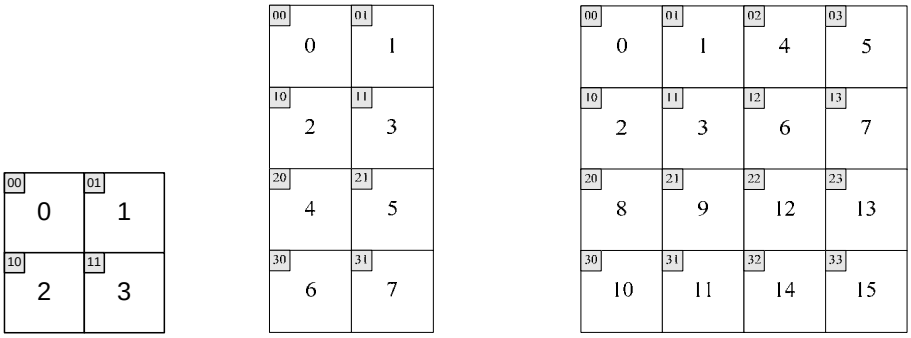


Figure 8-7 – Allocation des indices de dcC à chroma4x4BlkIdx: (a) chroma_format_idc égal à 1, (b) chroma_format_idc égal à 2, (c) chroma_format_idc égal à 3

Les éléments contenus dans la liste chromaList avec l'indice k = 1..15 sont spécifiés comme suit:

$$chromaList[\,k\,] = ChromaACLevel[\,chroma4x4BlkIdx\,][\,k-1\,] \quad (8\text{-}301)$$

b. Le processus de balayage inverse des coefficients de transformée, spécifié au § 8.5.9, est invoqué avec chromaList comme entrée et la matrice bidimensionnelle c comme résultat.

c. Le processus de normalisation et de transformation pour des blocs résiduels 4x4, spécifié au § 8.5.10, est invoqué avec c comme entrée et r comme résultat.

d. En fonction de la valeur de la variable chroma_format_idc, la position de l'échantillon supérieur gauche d'un bloc chroma 4x4 avec l'indice chroma4x4BlkIdx au sein du macrobloc est déduite comme suit

– Si chroma_format_idc est égal à 1 ou 2, ce qui suit est applicable.

$$xO = InverseRasterScan(\,chroma4x4BlkIdx, 4, 4, 8, 0\,) \quad (8\text{-}302)$$

$$yO = InverseRasterScan(\,chroma4x4BlkIdx, 4, 4, 8, 1\,) \quad (8\text{-}303)$$

– Sinon (si chroma_format_idc est égal à 3), ce qui suit est applicable:

$$xO = InverseRasterScan(\,chroma4x4BlkIdx / 4, 8, 8, 16, 0\,) + InverseRasterScan(\,chroma4x4BlkIdx \,\%\, 4, 4, 4, 8, 0\,) \quad (8\text{-}304)$$

yO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) +
InverseRasterScan(chroma4x4BlkIdx % 4, 4, 4, 8, 1) (8-305)

e. Lorsque le fanion residual_colour_transform_flag est égal à 1, la variable xO' est mise à xO % (4 << transform_size_8x8_flag), la variable yO' est mise à yO % (4 << transform_size_8x8_flag), et ce qui suit est applicable.

– Si ce processus est invoqué pour la composante chromatique Cb, la variable $R_{Cb,mn}$ est mise à r_{ij} avec i, j = 0..3, m = xO' + i, n = yO' + j, et ce processus est suspendu jusqu'à l'achèvement complet du processus de transformation chromatique résiduelle spécifié dans le § 8.5.13, après quoi la variable r_{ij} est mise à $R_{B,mn}$ avec i, j = 0..3, m = xO' + i, n = yO' + j et ce processus est poursuivi.

– Sinon (si ce processus est invoqué pour la composante chromatique Cr), la variable $R_{Cr,mn}$ est mise à r_{ij} avec i, j = 0..3, m = xO' + i, n = yO' + j et ce processus est suspendu jusqu'à l'achèvement complet du processus de transformation chromatique résiduelle spécifié dans le § 8.5.13, après quoi la variable r_{ij} est mise à $R_{R,mn}$ avec i, j = 0..3, m = xO' + i, n = yO' + j et ce processus est poursuivi.

f. La matrice 4x4 u avec éléments u_{ij} pour i, j = 0..3 est calculée comme suit:

$$u_{ij} = Clip1_C(pred_C[xO + j, yO + i] + r_{ij}) \quad (8\text{-}306)$$

Lorsque le fanion qpprime_y_zero_transform_bypass_flag est égal à 1 et que QP'_Y est égal à 0, le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de u_{ij} qui, calculée par l'équation 8-306, ne serait pas égale à $pred_C[xO + j, yO + i] + r_{ij}$.

g. Le processus de construction d'image avant processus de filtre de démontage de bloc décrit dans le § 8.5.12 est invoqué avec chroma4x4BlkIdx et u comme entrées.

8.5.5 Processus de balayage inverse pour les coefficients de transformée

L'entrée dans ce processus est une liste de 16 valeurs.

La sortie de ce processus est une variable c contenant une matrice bidimensionnelle de 4x4 valeurs. Dans le cas de coefficients de transformée, ces 4x4 valeurs représentent les niveaux attribués à des emplacements dans le bloc de transformée. Dans le cas de l'application du processus de balayage inverse à une liste de normalisation, la variable de sortie c contient une matrice bidimensionnelle représentant une matrice de normalisation 4x4.

Le processus de balayage inverse pour les coefficients de transformée mappe la séquence des niveaux des coefficients de transformée en les appliquant aux positions des niveaux de coefficient de transformée. Le Tableau 8-13 spécifie les deux mappages: balayage inverse en zigzag et balayage inverse de sous-trame. Le balayage inverse en zigzag est utilisé pour les coefficients de transformée dans les macroblocs de trame et le balayage inverse de sous-trame est utilisé pour les coefficients de transformée dans les macroblocs de sous-trame.

Le processus de balayage inverse pour listes de normalisation mappe la séquence des entrées dans la liste de normalisation en les appliquant aux positions dans la matrice de normalisation correspondante. Pour ce mappage, le balayage inverse en zigzag est utilisé.

La Figure 8-8 décrit ces balayages.

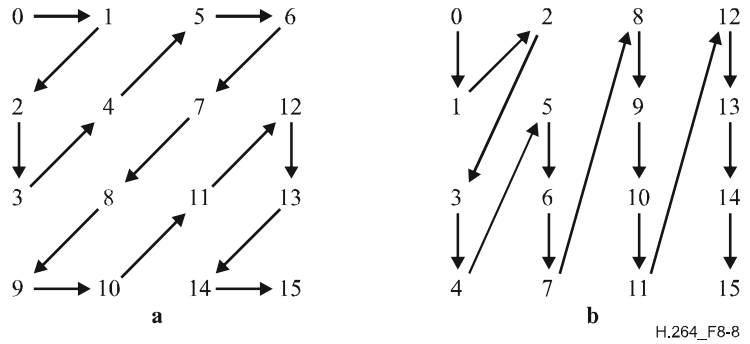


Figure 8-8 – Balayage de bloc 4x4: (a) Balayage en zigzag. (b) Balayage de sous-trame (pour information)

Le Tableau 8-13 donne le mappage entre l'indice idx de la liste d'entrées de 16 éléments et les indices i et j de la matrice bidimensionnelle c.

Tableau 8-13 – Spécification du mappage des indices à c_{ij} pour balayage en zigzag et de sous-trame

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{31}	c_{22}	c_{13}	c_{23}	c_{32}	c_{33}
sous-trame	c_{00}	c_{10}	c_{01}	c_{20}	c_{30}	c_{11}	c_{21}	c_{31}	c_{02}	c_{12}	c_{22}	c_{32}	c_{03}	c_{13}	c_{23}	c_{33}

8.5.6 Processus de balayage inverse pour les coefficients de transformée de 8x8 échantillons luma

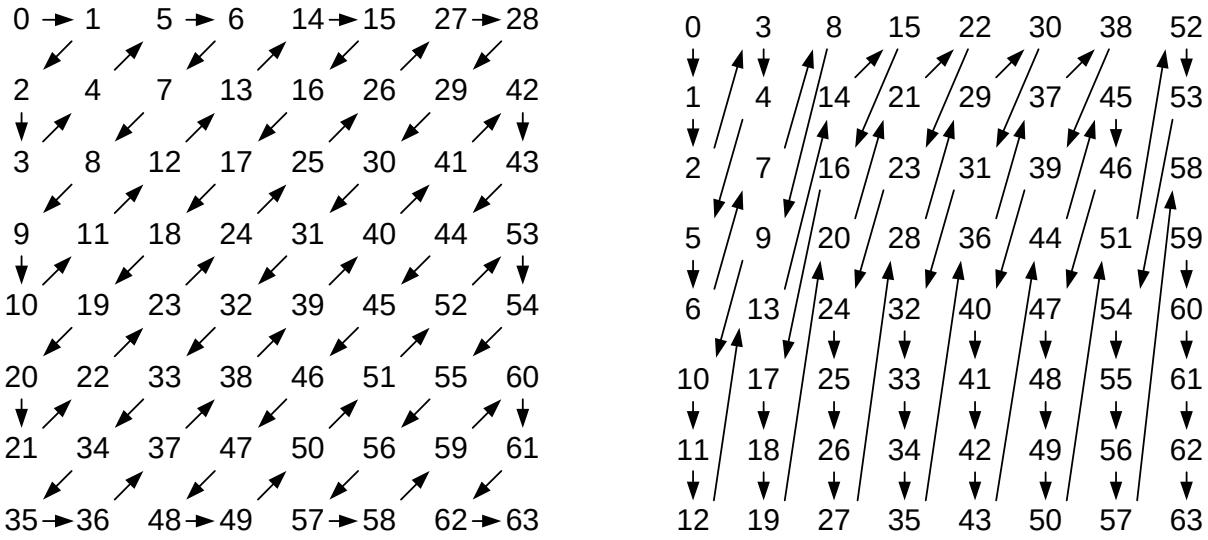
L'entrée dans ce processus est une liste de 64 valeurs.

La sortie de ce processus est une variable c contenant une matrice bidimensionnelle de 8x8 valeurs. Dans le cas des coefficients de transformée, ces 8x8 valeurs représentent les niveaux attribués à des emplacements dans le bloc de transformée. Dans le cas de l'application du processus de balayage inverse à une liste de normalisation, la variable de sortie c contient une matrice bidimensionnelle représentant une matrice de normalisation 8x8.

Le processus de balayage inverse pour les coefficients de transformée mappe la séquence des niveaux des coefficients de transformée en les appliquant aux positions des niveaux de coefficient de transformée. Le Tableau 8-14 spécifie les deux mappages: balayage inverse en zigzag 8x8 et balayage inverse de sous-trame 8x8. Le balayage inverse en zigzag 8x8 est utilisé pour les coefficients de transformée contenus dans les macroblocs de trame et le balayage inverse de sous-trame 8x8 est utilisé pour les coefficients de transformée contenus dans les macroblocs de sous-trame.

Le processus de balayage inverse pour listes de normalisation mappe la séquence des entrées dans la liste de normalisation en les appliquant aux positions dans la matrice de normalisation correspondante. Pour ce mappage, le balayage inverse en zigzag est utilisé.

La Figure 8-9 décrit ces balayages.



a b

Figure 8-9 – Balayage de bloc 8x8: (a) Balayage 8x8 en zigzag.
(b) Balayage 8x8 de sous-trame (pour information)

Le Tableau 8-14 fournit le mappage de l'indice idx de la liste d'entrée des 64 éléments à indices i et j de la matrice bidimensionnelle c.

Tableau 8-14 – Spécification du mappage des indices idx à c_{ij} pour balayage 8x8 en zigzag et de sous-trame

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{40}	c_{31}	c_{22}	c_{13}	c_{04}	c_{05}
sous-trame	c_{00}	c_{10}	c_{20}	c_{01}	c_{11}	c_{30}	c_{40}	c_{21}	c_{02}	c_{31}	c_{50}	c_{60}	c_{70}	c_{41}	c_{12}	c_{03}

Tableau 8-14 (suite) – Spécification du mappage des indices idx à cij pour balayage 8x8 en zigzag et de sous-trame

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
zig-zag	c_{14}	c_{23}	c_{32}	c_{41}	c_{50}	c_{60}	c_{51}	c_{42}	c_{33}	c_{24}	c_{15}	c_{06}	c_{07}	c_{16}	c_{25}	c_{34}
sous-trame	c_{22}	c_{51}	c_{61}	c_{71}	c_{32}	c_{13}	c_{04}	c_{23}	c_{42}	c_{52}	c_{62}	c_{72}	c_{33}	c_{14}	c_{05}	c_{24}

Tableau 8-14 (suite) – Spécification du mappage des indices idx à cij pour balayage 8x8 en zigzag et de sous-trame

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
zig-zag	c_{43}	c_{52}	c_{61}	c_{70}	c_{71}	c_{62}	c_{53}	c_{44}	c_{35}	c_{26}	c_{17}	c_{27}	c_{36}	c_{45}	c_{54}	c_{63}
sous-trame	c_{43}	c_{53}	c_{63}	c_{73}	c_{34}	c_{15}	c_{06}	c_{25}	c_{44}	c_{54}	c_{64}	c_{74}	c_{35}	c_{16}	c_{26}	c_{45}

Tableau 8-14 (fin) – Spécification du mappage des indices idx à cij pour balayage 8x8 en zigzag et de sous-trame

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
zig-zag	c_{72}	c_{73}	c_{64}	c_{55}	c_{46}	c_{37}	c_{47}	c_{56}	c_{65}	c_{74}	c_{75}	c_{66}	c_{57}	c_{67}	c_{76}	c_{77}
sous-trame	c_{55}	c_{65}	c_{75}	c_{36}	c_{07}	c_{17}	c_{46}	c_{56}	c_{66}	c_{76}	c_{27}	c_{37}	c_{47}	c_{57}	c_{67}	c_{77}

8.5.7 Processus de déduction pour les paramètres de quantification chroma et la fonction de normalisation

Les résultats de ce processus sont:

– QP_C: le paramètre de quantification chromatique pour chaque composante chromatique Cb et Cr

– QS_C: le paramètre de quantification chromatique additionnel pour chaque composante chromatique Cb et Cr requise pour le décodage des tranches SP et SI (si applicable)

NOTE 1 – Les valeurs QP_Y et QS_Y du paramètre de quantification QP sont toujours dans l'étendue de $-QpBdOffset_Y$ à 51, inclus. Les valeurs QP_C et QS_C du paramètre de quantification QP sont toujours dans l'étendue de $-QpBdOffset_C$ à 51, inclus.

La valeur de QP_C pour une composante chromatique est déterminée à partir de la valeur actuelle de QP_Y et à partir de la valeur de chroma_qp_indice_offset (pour la composante Cb) ou de second_chroma_qp_indice_offset (pour la composante Cr).

NOTE 2 – Les équations de normalisation sont spécifiées de façon que le facteur de normalisation du niveau équivalent des coefficients de transformée double pour chaque augmentation de 6 dans la valeur de QP_Y. Ainsi, il y a une augmentation d'environ 12 % du facteur utilisé pour la normalisation pour chaque augmentation de 1 dans la valeur de QP_Y.

La valeur de QP_C pour chaque composante chromatique est déterminée comme spécifié dans le Tableau 8-15 sur la base de l'indice désigné par qP_I.

La variable qP_{Offset} pour chaque composante chromatique est calculée comme suit:

– Si la composante chromatique est la composante Cb, qP_{Offset} est spécifié par

$$qP_{Offset} = \text{chroma\_qp\_index\_offset} \quad (8\text{-}307)$$

– Sinon (si la composante chromatique est la composante Cr), qP_{Offset} est spécifié par

$$qP_{Offset} = \text{second\_chroma\_qp\_index\_offset} \quad (8\text{-}308)$$

La valeur de qP_I pour chaque composante chromatique est calculée comme suit:

$$qP_I = \text{Clip3}(-\text{QpBdOffset}_C, 51, QP_Y + qP_{Offset}) \quad (8\text{-}309)$$

La valeur de QP'_C pour les composantes chromatiques est calculée comme suit:

$$QP'_C = QP_C + \text{QpBdOffset}_C \quad (8\text{-}310)$$

La valeur de $\text{BitDepth}'_C$ pour les composantes chromatiques est calculée comme suit:

$$\text{BitDepth}'_C = \text{BitDepth}_C + \text{residual\_colour\_transform\_flag} \quad (8\text{-}311)$$

Tableau 8-15 – Spécification de QP_C en fonction de qP_I

qP_I	<30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
QP_C	$=qP_I$	29	30	31	32	32	33	34	34	35	35	36	36	37	37	37	38	38	38	39	39	39	39

Lorsque la tranche en cours est une tranche SP ou SI, QS_C est déduit au moyen du processus ci-dessus, en substituant QP_Y à QS_Y et QP_C à QS_C.

La fonction LevelScale(m, i, j) est spécifiée comme suit.

– La fonction 4x4 matrix weightScale(i, j) est spécifiée comme suit:
  – La variable mbIsInterFlag est calculée comme suit:
    – Si le macrobloc actuel est codé au moyen de modes d'interprédiction de macrobloc, le fanion mbIsInterFlag est mis à 1.
    – Sinon (si le macrobloc actuel est codé au moyen de modes d'intraprédiction de macrobloc), mbIsInterFlag est mis à 0.
  – La variable iYCbCr est calculée comme suit:
    – Si la matrice de données d'entrée c se rapporte à un bloc d'échantillons luma résiduels, iYCbCr est mis à 0.
    – Sinon, si la matrice de données d'entrée c se rapporte à un bloc d'échantillons chroma résiduels et si la composante chromatique est égale à Cb, iYCbCr est mis à 1.
    – Sinon (si la matrice de données d'entrée c se rapporte à un bloc d'échantillons chroma résiduels et si la composante chromatique est égale à Cr), iYCbCr est mis à 2.
  – Le processus de balayage inverse pour les coefficients de transformée, spécifié dans le § 8.5.5, est invoqué avec ScalingList4x4[iYCbCr + ((mbIsInterFlag == 1) ? 3 : 0)] comme entrée et la sortie est attribuée à la fonction 4x4 matrix weightScale.

$$\text{LevelScale}(m, i, j) = \text{weightScale}(i, j) * \text{normAdjust}(m, i, j) \quad (8\text{-}312)$$

où:

$$\text{normAdjust}(m,i,j) = \begin{cases} V_{m0} & \text{pour (i\% 2, j\% égal à (00),} \\ V_{m1} & \text{pour (i,\% 2, j\% égal à (1,1),} \\ V_{m2} & \text{sinon} \end{cases} \quad (8\text{-}313)$$

où le premier et le second indice de v sont respectivement les indices de rangée et de colonne, de la matrice spécifiée comme suit:

$$v = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix} \tag{8-314}$$

La fonction LevelScale8x8(m, i, j) est spécifiée comme suit:

– La matrice 8x8 weightScale8x8(i, j) est spécifiée comme suit:

  – La variable mbIsInterFlag est calculée comme suit:

    – Si le macrobloc actuel est codé au moyen de modes d'interprédiction de macrobloc, mbIsInterFlag est mis à 1.

    – Sinon (si le macrobloc actuel est codé au moyen de modes d'intraprédiction de macrobloc), mbIsInterFlag est mis à 0.

  – Le processus de balayage inverse pour les coefficients de transformée de 8x8 échantillons luma spécifié dans le § 8.5.6 est invoqué avec ScalingList8x8[mbIsInterFlag] comme entrée et la sortie est attribuée à la matrice 8x8 weightScale8x8.

$$\text{LevelScale8x8}(m, i, j) = \text{weightScale8x8}(i, j) * \text{normAdjust8x8}(m, i, j) \tag{8-315}$$

où:

$$\text{normAdjust8x8}(m,i,j) = \begin{cases} V_{m0} & \text{pour (i\% 4, j\% 4) égal à (0,0),} \\ V_{m1} & \text{pour (i\% 2, j\% 2) égal à (1,1),} \\ V_{m2} & \text{pour (i\% 4, j\% 4) égal à (2,2),} \\ V_{m3} & \text{pour (i\% 4, j\% 2) égal à (0,1) ou (i\% 2, j\% 4) égal à (1,0),} \\ V_{m4} & \text{pour (i\% 4, j\% 4) égal à (0,2) ou (i\% 4, j\% 4) égal à (2,0),} \\ V_{m5} & \text{sinon;} \end{cases} \tag{8-316}$$

où le premier et le second indices inférieurs de v sont respectivement les indices de rangée et de colonne, de la matrice spécifiée par:

$$v = \begin{bmatrix} 20 & 18 & 32 & 19 & 25 & 24 \\ 22 & 19 & 35 & 21 & 28 & 26 \\ 26 & 23 & 42 & 24 & 33 & 31 \\ 28 & 25 & 45 & 26 & 35 & 33 \\ 32 & 28 & 51 & 30 & 40 & 38 \\ 36 & 32 & 58 & 34 & 46 & 43 \end{bmatrix} \tag{8-317}$$

8.5.8 Processus de normalisation et de transformation pour coefficients de transformée luma DC pour type de macrobloc Intra_16x16

Les entrées dans ce processus sont les valeurs de niveau de coefficient de transformée pour les coefficients de transformée luma DC de macroblocs Intra_16x16 comme une matrice c 4x4 avec des éléments c_{ij}, où i et j forment un indice de fréquence à deux dimensions.

Les résultats de ce processus sont 16 valeurs DC normalisées pour des blocs luma 4x4 de macroblocs Intra_16x16 comme une matrice 4x4 dcY avec des éléments dcY_{ij}.

Selon les valeurs du fanion qpprime_y_zero_transform_bypass_flag et QP'_Y, ce qui suit est applicable:

– Si qpprime_y_zero_transform_bypass_flag est égal à 1 et QP'_Y égal à 0, la sortie dcY est calculée comme suit:

$$dcY_{ij} = c_{ij} \quad \text{avec} \quad i, j = 0..3 \tag{8-318}$$

– Sinon (si qpprime_y_zero_transform_bypass_flag est égal à 0 ou si QP'_Y n'est pas égal à 0), le texte ci-après de ce processus spécifie la sortie.

La transformation inverse pour les coefficients apériodiques de transformée de 4x4 échantillons luma est spécifiée par:

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{8-319}$$

Le flux binaire ne doit pas contenir de données qui feraient qu'un élément f_{ij} de f avec i,j = 0 ..3 excède la gamme de valeurs entières de $-2^{(7 + \text{BitDepth}_Y)}$ à $2^{(7 + \text{BitDepth}_Y)}-1$, inclus.

Après la transformation inverse, la normalisation est effectuée comme suit.

– Si QP'_Y est supérieur ou égal à 36, le résultat normalisé est calculé comme

$$dcY_{ij} = (f_{ij} * \text{LevelScale}(QP'_Y \% 6, 0, 0)) << (QP'_Y / 6 - 6), \quad \text{avec} \quad i, j = 0..3 \tag{8-320}$$

– Sinon (si QP'_Y est inférieur à 36), le résultat normalisé est calculé comme

$$dcY_{ij} = (f_{ij} * \text{LevelScale}(QP_Y \% 6, 0, 0) + 2^{5 - QP_Y/6}) >> (2 - QP'_Y / 6), \quad \text{avec} \quad i, j = 0..3 \tag{8-321}$$

Le flux binaire ne doit pas contenir de données qui feraient qu'un élément dcY_{ij} de dcY avec i,j = 0..3 excède la gamme de valeurs entières de $-2^{(7 + \text{BitDepth}_Y)}$ à $2^{(7 + \text{BitDepth}_Y)}-1$, inclus.

NOTE 1 – Lorsque entropy_coding_mode_flag est égal à 0 et que QP'_y est inférieur à 10 et que profile_idc est égal à 66, 77 ou 88, la gamme des valeurs qui peuvent être représentées pour les éléments c_{ij} de c est insuffisante afin de représenter la gamme complète des valeurs des éléments dcY_{ij} de dcY qui pourrait être nécessaire afin de former une approximation fidèle du contenu de toute image source possible par l'utilisation du type de macroblocs Intra_16x16.

NOTE 2 – Etant donné que la limitation de la gamme des valeurs imposée aux éléments dcY_{ij} de dcY est imposée après le décalage à droite dans l'équation 8-321, une plus large gamme de valeurs doit être admise dans le décodeur avant le décalage à droite.

8.5.9 Processus de normalisation et de transformation pour coefficients de transformée chroma DC

Les entrées dans ce processus sont des valeurs de niveau de coefficient de transformée pour des coefficients de transformée chroma DC d'une composante chromatique du macrobloc en une matrice c (MbWidthC / 4)x(MbHeightC / 4) avec des éléments c_{ij}, où i et j forment un indice de fréquence à deux dimensions.

Les résultats de ce processus sont les valeurs apériodiques normalisées en une matrice dcC (MbWidthC / 4)x(MbHeightC / 4) avec les éléments dcC_{ij}.

Selon les valeurs de qpprime_y_zero_transform_bypass_flag et QP'_Y, ce qui suit est applicable:

– Si le fanion qpprime_y_zero_transform_bypass_flag est égal à 1 et QP'_Y est égal à 0, la sortie dcC est calculée comme suit:

$$dcC_{ij} = c_{ij} \quad \text{avec} \quad i = 0..(MbWidthC / 4) - 1 \text{ et } j = 0..(MbHeightC / 4) - 1. \tag{8-322}$$

– Sinon (si qpprime_y_zero_transform_bypass_flag est égal à 0 ou si QP'_Y n'est pas égal à 0), le texte ci-après de ce processus spécifie la sortie.

Selon la variable chroma_format_idc, la transformation inverse est spécifiée comme suit:

– Si chroma_format_idc est égal à 1, la transformation inverse pour les coefficients apériodiques de transformée de 2x2 échantillons chroma est spécifiée par

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-323}$$

– Sinon, si chroma_format_idc est égal à 2, la transformation inverse pour les coefficients apériodiques de transformée de 2x2 échantillons chroma est spécifiée par

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ c_{20} & c_{21} \\ c_{30} & c_{31} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-324}$$

– Sinon (si chroma_format_idc est égal à 3), la transformation inverse pour les coefficients apériodiques de transformée de 4x4 échantillons chroma est spécifiée comme suit:

  – Si le fanion residual_colour_transform_flag est égal à 1 et si le mode de prédiction du macrobloc actuel MbPartPredMode(mb_type, 0) est Intra_4x4 ou Intra_8x8, la transformation inverse pour les coefficients apériodiques de transformée de 4x4 échantillons chroma est spécifiée par

$$f_{ij} = c_{ij} << 2 \quad \text{avec} \quad i, j = 0..3 \tag{8-325}$$

  – Sinon, la transformation inverse pour les coefficients apériodiques de transformée de 4x4 échantillons chroma est spécifiée par

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{8-326}$$

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément f_{ij} de f avec i, j = 0..3 hors de l'étendue de valeurs entières de $-2^{(7 + \text{BitDepth}'_C)}$ à $2^{(7 + \text{BitDepth}'_C)}-1$, inclus.

Après la transformation inverse, la normalisation est effectuée en fonction de la variable chroma_format_idc comme suit:

– Si chroma_format_idc est égal à 1, le résultat normalisé est calculé comme suit:

$$dcC_{ij} = ((f_{ij} * \text{LevelScale}(QP'_C \,\%\, 6, 0, 0)) << (QP'_C / 6)) >> 5, \quad \text{avec} \quad i, j = 0, 1 \tag{8-327}$$

– si chroma_format_idc est égal à 2, ce qui suit est applicable:

  – La variable $QP'_{C,DC}$ est calculée comme suit:

$$QP'_{C,DC} = QP'_C + 3 \tag{8-328}$$

  – Selon la valeur de $QP'_{C,DC}$, ce qui suit est applicable:

    – Si $QP'_{C,DC}$ est plus grand ou égal à 36, le résultat normalisé est calculé comme suit:

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP'_{C,DC} \,\%6, 0, 0)) << (QP'_{C,DC} / 6 - 6), \quad \text{avec } i = 0..3, j = 0, 1 \tag{8-329}$$

    – Sinon (si $QP'_{C,DC}$ est inférieur à 36), le résultat normalisé est calculé comme suit:

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP'_{C,DC} \,\%6,0,0) + 2^{5 - QP'_{C,DC}/6}) >> (6 - QP'_{C,DC} / 6), \text{ avec } i = 0..3, j = 0,1 \tag{8-330}$$

– Sinon (si chroma_format_idc est égal à 3), ce qui suit est applicable:

  – Si QP'_C est plus grand ou égal à 36, le résultat normalisé est calculé comme suit:

$$dcC_{ij} = (f_{ij} * \text{LevelScale} (QP'_C \,\%6, 0, 0)) << (QP'_C / 6 - 6), \quad \text{avec} \quad i, j = 0..3. \tag{8-331}$$

– Sinon (si QP'_C est inférieur à 36), le résultat normalisé est calculé comme suit:

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_C \% 6, 0, 0) + 2^{5-QP'_C/6}) >> (6 - QP'_C / 6), \text{ avec } i, j, = 0..3 \quad (8\text{-}332)$$

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément dcC_{ij} de dcC avec i, j = 0..3 hors de l'étendue de valeurs entières de $-2^{(7 + BitDepth'_C)}$ à $2^{(7 + BitDepth'_C)}-1$, inclus.

NOTE 1 – Lorsque entropy_coding_mode_flag est égal à 0 et que QP'_C est inférieur à 4 et lorsque profil_idc est égal à 66, 77 ou 88, la gamme des valeurs qui peuvent être représentées pour les éléments C_{ij} de C peut être insuffisante afin de représenter la gamme complète des valeurs des éléments dcC_{ij} de dcC qui pourrait être nécessaire afin de former une approximation fidèle du contenu de toute image source possible.

NOTE 2 – Etant donné que la limitation de la gamme des valeurs imposée aux éléments dcC_{ij} de dcC est imposée après le décalage à droite dans l'équation 8-327, 8-330 ou 8-332, une plus large gamme de valeurs doit être admise dans le décodeur avant le décalage à droite.

8.5.10 Processus de normalisation et de transformation pour blocs de 4x4 échantillons résiduels

L'entrée dans ce processus est une matrice 4x4 c avec des éléments c_{ij} qui sont soit une matrice se rapportant au bloc résiduel de la composante luma soit une matrice se rapportant au bloc résiduel d'une composante chromatique.

Les résultats de ce processus sont des valeurs résiduelles d'échantillon en une matrice 4x4 r avec des éléments r_{ij}.

Selon les valeurs de qpprime_y_zero_transform_bypass_flag et de QP'_Y, ce qui suit est applicable:

– Si qpprime_y_zero_transform_bypass_flag est égal à 1 et QP'_Y est égal à 0, la sortie r est calculée comme suit:

$$r_{ij} = c_{ij} \text{ avec } i, j = 0..3 \quad (8\text{-}333)$$

– Sinon (si qpprime_y_zero_transform_bypass_flag est égal à 0 ou si QP'_Y n'est pas égal à 0), le texte ci-après de ce processus spécifie la sortie.

La variable bitDepth est calculée comme suit:

– Si la matrice de données d'entrée c se rapporte à un bloc d'échantillons luma résiduels, bitDepth est mis à $BitDepth_Y$.

– Sinon (si la matrice de données d'entrée c se rapporte à un bloc d'échantillons chroma résiduels), bitDepth est mis à $BitDepth'_C$.

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément c_{ij} de c avec i, j = 0..3 hors de l'étendue de valeurs entières de $-2^{(7 + bitDepth)}$ à $2^{(7 + bitDepth)}-1$, inclus.

La variable sMbFlag est déduite comme suit.

– Si mb_type est égal à SI ou si le mode de prédiction de macrobloc est égal à Inter dans une tranche SP, sMbFlag est mis égal à 1.

– Sinon (si mb_type n'est pas égal à SI et si le mode de prédiction de macrobloc n'est pas égal à Inter dans une tranche SP), sMbFlag est mis égale à 0.

La variable qP est déduite comme suit.

– Si la matrice de résultat c se rapporte à un bloc luma résiduel et si sMbFlag est égal à 0

$$qP = QP'_Y \quad (8\text{-}334)$$

– Sinon, si la matrice de résultat c se rapporte à un bloc luma résiduel et si sMbFlag est égal à 1

$$qP = QS_Y \quad (8\text{-}335)$$

– Sinon, si la matrice de résultat c se rapporte à un bloc chroma résiduel et si sMbFlag est égal à 0

$$qP = QP'_C \quad (8\text{-}336)$$

– Sinon (si la matrice de résultat c se rapporte à un bloc chroma résiduel et sMbFlag est égal à 1)

$$qP = QS_C \quad (8\text{-}337)$$

La normalisation des niveaux de coefficient de transformée de bloc 4x4 c_{ij} s'effectue comme suit.

– Si toutes les conditions suivantes sont vraies:
  – i est égal à 0
  – j est égal à 0
  – c se rapporte à un bloc luma résiduel codé au moyen du mode de prédiction Intra_16x16 ou c se rapporte à un bloc résiduel chroma

  la variable d_{00} est déduite comme:

$$d_{00} = c_{00} \tag{8-338}$$

– Sinon, ce qui suit s'applique.
  – Si qP est plus grand ou égal à 24, le résultat normalisé est calculé comme suit:

$$d_{ij} = (c_{ij} * \text{LevelScale}(qP \% 6, i, j)) << (qP / 6 - 4), \text{ avec } i,j = 0..3 \text{ excepté comme noté ci-dessus} \tag{8-339}$$

  – Sinon (si qP est inférieur à 24), le résultat normalisé est calculé comme suit:

$$d_{ij} = (c_{ij} * \text{LevelScale}(qP \% 6, i, j) + 2^{3-qP/6}) >> (4 - qP / 6), \text{ avec } i, j = 0..3 \text{ excepté comme noté ci-dessus} \tag{8-340}$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément d_{ij} de d avec i, j = 0..3 excède la gamme de valeurs entières allant de $-2^{(7 + \text{bitDepth})}$ à $2^{(7 + \text{bitDepth})} - 1$ inclus.

Le processus de transformée doit convertir le bloc de coefficients de transformée normalisés en un bloc d'échantillons de résultat d'une manière mathématiquement équivalente à ce qui suit.

D'abord, chaque rangée (horizontale) de coefficients de transformée normalisés est transformée au moyen d'une transformation inverse unidimensionnelle comme suit.

Un ensemble de valeurs intermédiaires est calculé comme suit.

$$e_{i0} = d_{i0} + d_{i2}, \quad \text{avec} \quad i = 0..3 \tag{8-341}$$

$$e_{i1} = d_{i0} - d_{i2}, \quad \text{avec} \quad i = 0..3 \tag{8-342}$$

$$e_{i2} = (d_{i1} >> 1) - d_{i3}, \quad \text{avec} \quad i = 0..3 \tag{8-343}$$

$$e_{i3} = d_{i1} + (d_{i3} >> 1), \quad \text{avec} \quad i = 0..3 \tag{8-344}$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément e_{ij} de e avec i, j = 0..3 excède la gamme des valeurs entières allant de $-2^{(7 + \text{bitDepth})}$ à $2^{(7 + \text{bitDepth})} - 1$ inclus.

Alors, le résultat de la transformée est calculé à partir de ces valeurs intermédiaires comme suit.

$$f_{i0} = e_{i0} + e_{i3}, \quad \text{avec} \quad i = 0..3 \tag{8-345}$$

$$f_{i1} = e_{i1} + e_{i2}, \quad \text{avec} \quad i = 0..3 \tag{8-346}$$

$$f_{i2} = e_{i1} - e_{i2}, \quad \text{avec} \quad i = 0..3 \tag{8-347}$$

$$f_{i3} = e_{i0} - e_{i3}, \quad \text{avec} \quad i = 0..3 \tag{8-348}$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément f_{ij} de f avec i, j = 0..3 excède la gamme de valeurs entières allant de $-2^{(7 + \text{bitDepth})}$ à $2^{(7 + \text{bitDepth})} - 1$ inclus.

Puis, chaque colonne (verticale) de la matrice résultante est transformée au moyen de la même transformation inverse unidimensionnelle comme suit.

On calcule comme suit un ensemble de valeurs intermédiaires.

$$g_{0j} = f_{0j} + f_{2j}, \quad \text{avec} \quad j = 0..3 \tag{8-349}$$

$$g_{1j} = f_{0j} - f_{2j}, \quad \text{avec} \quad j = 0..3 \tag{8-350}$$

$$g_{2j} = (f_{1j} >> 1) - f_{3j}, \quad \text{avec} \quad j = 0..3 \tag{8-351}$$

$$g_{3j} = f_{1j} + (f_{3j} >> 1), \quad \text{avec} \quad j = 0..3 \tag{8-352}$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément g_{ij} de g avec i, j = 0..3 excède la gamme de valeurs entières allant de $-2^{(7 + \text{bitDepth})}$ à $2^{(7 + \text{bitDepth})} - 1$ inclus.

Puis, le résultat transformé est calculé comme suit à partir de ces valeurs intermédiaires.

$$h_{0j} = g_{0j} + g_{3j}, \quad \text{avec} \quad j = 0..3 \tag{8-353}$$

$$h_{1j} = g_{1j} + g_{2j}, \quad \text{avec} \quad j = 0..3 \tag{8-354}$$

$$h_{2j} = g_{1j} - g_{2j}, \quad \text{avec} \quad j = 0..3 \tag{8-355}$$

$$h_{3j} = g_{0j} - g_{3j}, \quad \text{avec} \quad j = 0..3 \tag{8-356}$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément h_{ij} of h avec i, j = 0..3 excède la gamme de valeurs entières allant de $-2^{(7 + \text{bitDepth})}$ à $2^{(7 + \text{bitDepth})} - 33$ inclus.

Après avoir effectué les deux transformées inverses unidimensionnelles, l'horizontale et la verticale, afin de produire une matrice d'échantillons de transformée, les valeurs finales d'échantillon résiduel construit sont calculées comme:

$$r_{ij} = (h_{ij} + 2^5) >> 6 \quad \text{avec} \quad i, j = 0..3 \tag{8-357}$$

8.5.11 Processus de normalisation et de transformation pour blocs de 8x8 échantillons luma résiduels

L'entrée dans ce processus est une matrice c de 8x8 éléments c_{ij} qui se rapporte en un bloc résiduel 8x8 de la composante luma.

Les résultats de ce processus sont les suivants: valeurs d'échantillons résiduels formant la matrice r de 8x8 éléments r_{ij}.

Selon les valeurs du fanion qpprime_y_zero_transform_bypass_flag et de QP'_Y, ce qui suit est applicable:

– Si qpprime_y_zero_transform_bypass_flag est égal à 1 et si QP'_Y est égal à 0, la sortie r est calculée comme suit:

$$r_{ij} = c_{ij} \quad \text{avec} \quad i, j = 0..7 \tag{8-358}$$

– Sinon (si qpprime_y_zero_transform_bypass_flag est égal à 0 ou si QP'_Y n'est pas égal à 0), le texte ci-après de ce processus spécifie la sortie.

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément c_{ij} de c avec i, j = 0..7 hors de l'étendue des valeurs entières de $-2^{(7 + \text{BitDepth}_Y)}$ à $2^{(7 + \text{BitDepth}_Y)} - 1$, inclus.

Le processus de normalisation pour les niveaux des coefficients de transformée d'un bloc 8x8 c_{ij} procède comme suit:

– Si QP'_Y est plus grand ou égal à 36, le résultat normalisé est calculé comme suit:

$$d_{ij} = (c_{ij} * \text{LevelScale8x8}(QP'_Y \% 6, i, j)) << (QP'_Y / 6 - 6), \text{ avec } i,j = 0..7 \tag{8-359}$$

– Sinon (si QP'_Y est inférieur à 36), le résultat normalisé est calculé comme suit:

$$d_{ij} = (c_{ij} * \text{LevelScale8x8}(QP'_Y \% 6, i, j)) + 2^{5-QP'_Y/6}) >> (6 - QP'_Y /6), \text{ avec } i,j = 0..7 \quad (8\text{-}360)$$

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément d_{ij} de d avec i, j = 0..7 hors de l'étendue de valeurs entières de $-2^{(7 + \text{BitDepth}_Y)}$ à $2^{(7 + \text{BitDepth}_Y)} - 1$, inclus.

Le processus de transformation doit convertir le bloc de coefficients normalisés de transformée en un bloc de échantillons de sortie d'une manière mathématiquement équivalente à ce qui suit.

Tout d'abord, chaque rangée (horizontale) de coefficients normalisés de transformée est transformée au moyen d'une transformation inverse unidimensionnelle comme suit:

– Un ensemble de valeurs intermédiaires e_{ij} est calculé par:

$$e_{i0} = d_{i0} + d_{i4}, \text{ avec } i = 0..7 \quad (8\text{-}361)$$

$$e_{i1} = - d_{i3} + d_{i5} - d_{i7} - (d_{i7} >> 1), \text{ avec } i = 0..7 \quad (8\text{-}362)$$

$$e_{i2} = d_{i0} - d_{i4}, \text{ avec } i = 0..7 \quad (8\text{-}363)$$

$$e_{i3} = d_{i1} + d_{i7} - d_{i3} - (d_{i3} >> 1), \text{ avec } i = 0..7 \quad (8\text{-}364)$$

$$e_{i4} = (d_{i2} >> 1) - d_{i6}, \text{ avec } i = 0..7 \quad (8\text{-}365)$$

$$e_{i5} = - d_{i1} + d_{i7} + d_{i5} + (d_{i5} >> 1), \text{ avec } i = 0..7 \quad (8\text{-}366)$$

$$e_{i6} = d_{i2} + (d_{i6} >> 1), \text{ avec } i = 0..7 \quad (8\text{-}367)$$

$$e_{i7} = d_{i3} + d_{i5} + d_{i1} + (d_{i1} >> 1), \text{ avec } i = 0..7 \quad (8\text{-}368)$$

– Un second ensemble de résultats intermédiaires f_{ij} est calculé à partir des valeurs intermédiaires e_{ij} comme suit:

$$f_{i0} = e_{i0} + e_{i6}, \text{ avec } i = 0..7 \quad (8\text{-}369)$$

$$f_{i1} = e_{i1} + (e_{i7} >> 2), \text{ avec } i = 0..7 \quad (8\text{-}370)$$

$$f_{i2} = e_{i2} + e_{i4}, \text{ avec } i = 0..7 \quad (8\text{-}371)$$

$$f_{i3} = e_{i3} + (e_{i5} >> 2), \text{ avec } i = 0..7 \quad (8\text{-}372)$$

$$f_{i4} = e_{i2} - e_{i4}, \text{ avec } i = 0..7 \quad (8\text{-}373)$$

$$f_{i5} = (e_{i3} >> 2) - e_{i5}, \text{ avec } i = 0..7 \quad (8\text{-}374)$$

$$f_{i6} = e_{i0} - e_{i6}, \text{ avec } i = 0..7 \quad (8\text{-}375)$$

$$f_{i7} = e_{i7} - (e_{i1} >> 2), \text{ avec } i = 0..7 \quad (8\text{-}376)$$

– Ensuite, le résultat de la transformée g_{ij} est calculé à partir de ces valeurs intermédiaires f_{ij} comme suit:

$$g_{i0} = f_{i0} + f_{i7}, \text{ avec } i = 0..7 \quad (8\text{-}377)$$

$$g_{i1} = f_{i2} + f_{i5},\ \text{avec}\ i = 0..7 \tag{8-378}$$

$$g_{i2} = f_{i4} + f_{i3},\ \text{avec}\ i = 0..7 \tag{8-379}$$

$$g_{i3} = f_{i6} + f_{i1},\ \text{avec}\ i = 0..7 \tag{8-380}$$

$$g_{i4} = f_{i6} - f_{i1},\ \text{avec}\ i = 0..7 \tag{8-381}$$

$$g_{i5} = f_{i4} - f_{i3},\ \text{avec}\ i = 0..7 \tag{8-382}$$

$$g_{i6} = f_{i2} - f_{i5},\ \text{avec}\ i = 0..7 \tag{8-383}$$

$$g_{i7} = f_{i0} - f_{i7},\ \text{avec}\ i = 0..7 \tag{8-384}$$

Ensuite, chaque colonne (verticale) de la matrice résultante est transformée au moyen de la même transformation inverse unidimensionnelle comme suit:

– Un ensemble de valeurs intermédiaires h_{ij} est calculé à partir de la valeur transformée horizontalement g_{ij} comme suit:

$$h_{0j} = g_{0j} + g_{4j},\ \text{avec}\ j = 0..7 \tag{8-385}$$

$$h_{1j} = -g_{3j} + g_{5j} - g_{7j} - (g_{7j} >> 1),\ \text{avec}\ j = 0..7 \tag{8-386}$$

$$h_{2j} = g_{0j} - g_{4j},\ \text{avec}\ j = 0..7 \tag{8-387}$$

$$h_{3j} = g_{1j} + g_{7j} - g_{3j} - (g_{3j} >> 1),\ \text{avec}\ j = 0..7 \tag{8-388}$$

$$h_{4j} = (\ g_{2j} >> 1\ ) - g_{6j},\ \text{avec}\ j = 0..7 \tag{8-389}$$

$$h_{5j} = -g_{1j} + g_{7j} + g_{5j} + (g_{5j} >> 1),\ \text{avec}\ j = 0..7 \tag{8-390}$$

$$h_{6j} = g_{2j} + (\ g_{6j} >> 1\ ),\ \text{avec}\ j = 0..7 \tag{8-391}$$

$$h_{7j} = g_{3j} + g_{5j} + g_{1j} + (g_{1j} >> 1),\ \text{avec}\ j = 0..7 \tag{8-392}$$

– Un second ensemble de résultats intermédiaires k_{ij} est calculé à partir des valeurs intermédiaires h_{ij} comme suit:

$$k_{0j} = h_{0j} + h_{6j},\ \text{avec}\ j = 0..7 \tag{8-393}$$

$$k_{1j} = h_{1j} + (h_{7j} >> 2),\ \text{avec}\ j = 0..7 \tag{8-394}$$

$$k_{2j} = h_{2j} + h_{4j},\ \text{avec}\ j = 0..7 \tag{8-395}$$

$$k_{3j} = h_{3j} + (h_{5j} >> 2),\ \text{avec}\ j = 0..7 \tag{8-396}$$

$$k_{4j} = h_{2j} - h_{4j},\ \text{avec}\ j = 0..7 \tag{8-397}$$

$$k_{5j} = (h_{3j} >> 2) - h_{5j}, \text{ avec } j = 0..7 \quad (8\text{-}398)$$

$$k_{6j} = h_{0j} - h_{6j}, \text{ avec } j = 0..7 \quad (8\text{-}399)$$

$$k_{7j} = h_{7j} - (h_{1j} >> 2), \text{ avec } j = 0..7 \quad (8\text{-}400)$$

– Ensuite, le résultat de la transformée m_{ij} est calculé à partir de ces valeurs intermédiaires k_{ij} comme suit:

$$m_{0j} = k_{0j} + k_{7j}, \text{ avec } j = 0..7 \quad (8\text{-}401)$$

$$m_{1j} = k_{2j} + k_{5j}, \text{ avec } j = 0..7 \quad (8\text{-}402)$$

$$m_{2j} = k_{4j} + k_{3j}, \text{ avec } j = 0..7 \quad (8\text{-}403)$$

$$m_{3j} = k_{6j} + k_{1j}, \text{ avec } j = 0..7 \quad (8\text{-}404)$$

$$m_{4j} = k_{6j} - k_{1j}, \text{ avec } j = 0..7 \quad (8\text{-}405)$$

$$m_{5j} = k_{4j} - k_{3j}, \text{ avec } j = 0..7 \quad (8\text{-}406)$$

$$m_{6j} = k_{2j} - k_{5j}, \text{ avec } j = 0..7 \quad (8\text{-}407)$$

$$m_{7j} = k_{0j} - k_{7j}, \text{ avec } j = 0..7 \quad (8\text{-}408)$$

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément e_{ij}, f_{ij}, g_{ij}, h_{ij}, ou k_{ij} pour i et j dans l'étendue de 0..7, inclus, hors de l'étendue des valeurs entières de $-2^{(7 + \text{BitDepth}_Y)}$ à $2^{(7 + \text{BitDepth}_Y)} - 1$, inclus.

Le flux binaire ne doit pas contenir de données qui se traduiraient par l'apparition d'un quelconque élément m_{ij} pour i et j dans l'étendue de 0..7, inclus, hors de l'étendue des valeurs entières de $-2^{(7 + \text{BitDepth}_Y)}$ à $2^{(7 + \text{BitDepth}_Y)} - 33$, inclus.

Après exécution des deux transformations inverses unidimensionnelles (horizontale et verticale) afin de produire une matrice d'échantillons de transformée, les valeurs des échantillons résiduels finalement construits sont des variables calculées comme suit:

$$r_{ij} = (m_{ij} + 2^5) >> 6 \text{ avec } i, j = 0..7 \quad (8\text{-}409)$$

8.5.12 Processus de construction d'image antérieurement au processus de filtrage de démontage de blocs

Les entrées dans ce processus sont:

– luma4x4BlkIdx ou chroma4x4BlkIdx ou luma8x8BlkIdx

– une matrice d'échantillons u avec des éléments u_{ij} qui est soit un bloc 4x4 luma ou un bloc 4x4 chroma ou un bloc 8x8 luma.

La position de l'échantillon luma supérieur gauche du macrobloc en cours est déduite par invocation du processus de balayage inverse de macrobloc du § 6.4.1 avec CurrMbAddr en entrée et le résultat étant alloué à (xP, yP).

Lorsque u est un bloc luma, pour chaque échantillon u_{ij} du bloc luma, on applique ce qui suit.

– En fonction de la taille du bloc u, on applique ce qui suit.

  – Si u est un bloc 4x4 luma, la position de l'échantillon supérieur gauche du bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc est déduite en invoquant le processus de balayage inverse de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO) et la variable nE est mise égale à 4.

– Sinon (si u est un bloc 8x8 luma), la position de l'échantillon supérieur gauche du bloc luma 8x8 avec l'indice luma8x8BlkIdx au sein du macrobloc est déduite par invocation du processus de balayage inverse de bloc 8x8 luma du § 6.4.4 avec luma8x8BlkIdx comme entrée et le résultat étant alloué à (xO, yO) et la variable nE est mise égale à 8.

– En fonction de la variable MbaffFrameFlag et du macrobloc actuel, on applique ce qui suit.

  – Si MbaffFrameFlag est égal à 1 et que le macrobloc en cours est un macrobloc de sous-trame

$$S'_L[\ xP + xO + j, yP + 2 * (yO + i)\ ] = u_{ij} \quad \text{avec } i, j = 0..nE - 1 \tag{8-410}$$

  – Sinon (si MbaffFrameFlag est égal à 0 ou le macrobloc en cours est un macrobloc de trame)

$$S'_L[\ xP + xO + j, yP + yO + i\ ] = u_{ij} \quad \text{avec } i, j = 0..nE - 1 \tag{8-411}$$

Lorsque u est un bloc chroma, pour chaque échantillon u_{ij} du bloc chroma 4x4, on applique ce qui suit.

– L'indice inférieur C dans la variable S'_C est remplacé par Cb pour la composante chroma Cb et par Cr pour la composante chroma Cr.

– Selon la variable chroma_format_idc, la position de l'échantillon supérieur gauche d'un bloc chroma 4x4 avec l'indice chroma4x4BlkIdx au sein du macrobloc est déduite comme suit:

  – si chroma_format_idc est égal à 1 ou à 2, on applique ce qui suit:

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \tag{8-412}$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \tag{8-413}$$

  – Sinon (si chroma_format_idc est égal à 3), ce qui suit est applicable:

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 0) \tag{8-414}$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 1) \tag{8-415}$$

– Selon la variable MbaffFrameFlag et le macrobloc actuel, ce qui suit est applicable:

  – Si MbaffFrameFlag est égal à 1 et si le macrobloc actuel est un sous-macrobloc de trames

$$S'_C[\ (xP / \text{subWidthC}) + xO + j, ((yP + \text{SubHeightC} - 1) / \text{SubHeightC}) + 2 * (yO + i)\ ] = u_{ij} \quad \text{avec } i, j = 0..3 \tag{8-416}$$

  – Sinon (si MbaffFrameFlag est égal à 0 ou si le macrobloc actuel est un macrobloc de trames),

$$S'_C[\ (xP/ \text{subWidthC}) + xO + j, (yP / \text{SubHeightC}) + yO + i\ ] = u_{ij} \text{ avec } i, j = 0..3 \tag{8-417}$$

8.5.13 Processus de transformation d'échantillons chromatiques résiduels

Ce processus est invoqué lorsque le fanion residual_colour_transform_flag est égal à 1.

Après invocation, ce processus est suspendu jusqu'à ce que la déduction de $R_{Y,ij}$, $R_{Cb,ij}$, et $R_{Cr,ij}$ ait été effectuée pour i, j = 0..ijMax, où ijMax est spécifié comme suit:

– Si transform_size_8x8_flag est égal à 0, la variable ijMax est mise à 3.

– Sinon (si transform_size_8x8_flag est égal à 1), la variable ijMax est mise à 7.

A la reprise de ce processus, toutes les valeurs $R_{Y,ij}$, $R_{Cb,ij}$, et $R_{Cr,ij}$ avec i, j = 0..ijMax doivent être disponibles avant les invocations des processus correspondants, spécifiés dans les § 8.5.1, 8.5.2, 8.5.3, ou 8.5.4

Pour chaque i, j = 0..ijMax, la transformée chromatique résiduelle est calculée comme suit:

$$t = R_{Y,ij} - (R_{Cb,ij} >> 1) \quad (8\text{-}418)$$

$$R_{G,ij} = t + R_{Cb,ij} \quad (8\text{-}419)$$

$$R_{B,ij} = t - (R_{Cr,ij} >> 1) \quad (8\text{-}420)$$

$$R_{R,ij} = R_{B,ij} + R_{Cr,ij} \quad (8\text{-}421)$$

NOTE – La transformée chromatique résiduelle est similaire à la transformation YCgCo spécifiée dans les équations E-30 à E-33. Cependant, la transformée chromatique résiduelle agit sur les données différentielles, résiduelles et décodées qui sont contenues dans le processus de décodage plutôt que d'agir comme une étape de post-traitement qui est hors du processus de décodage spécifié dans la présente Recommandation | Norme internationale.

8.6 Processus de décodage pour macroblocs P dans des tranches SP ou des macroblocs SI

Ce processus est invoqué lors du décodage de types de macrobloc P dans une tranche de type SP ou un macrobloc de type SI dans des tranches SI.

Les entrées dans ce processus sont les niveaux de coefficient de transformée résiduelle de prédiction et les échantillons prédits pour le macrobloc en cours.

Les résultats de ce processus sont les échantillons du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

Le présent paragraphe spécifie le processus de décodage du coefficient de transformée et le processus de construction d'image pour les macroblocs de type P dans les tranches SP et les macroblocs de type SI dans les tranches SI.

NOTE – Les tranches SP utilisent le codage interprédictif afin d'exploiter les redondances temporelles dans la séquence, d'une manière similaire au codage de tranche P. A la différence du codage de tranche P, le codage de tranche SP permet cependant une reconstruction à l'identique d'une tranche même lorsque des images de référence différentes sont utilisées. Les tranches SI utilisent la prédiction spatiale, d'une façon similaire aux tranches I. Le codage de tranche SI permet une reconstruction à l'identique dans une tranche SP correspondante. Les propriétés des tranches SP et SI aident à fournir les fonctionnalités afin de commuter les flux binaires, faire des raccords, donner un accès aléatoire, un accès rapide vers l'avant ou vers l'arrière, et pour la résistance aux erreurs ou leur correction.

Une tranche SP se compose de macroblocs codés comme macroblocs de type I ou macroblocs de type P.

Une tranche SI se compose de macroblocs codés comme macroblocs de type I ou macroblocs de type SI.

Le processus de décodage du coefficient de transformée et le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs pour les macroblocs de type I dans les tranches SI sont invoqués comme spécifié au § 8.5. Les macroblocs de type SI sont décodés comme décrit ci-dessous.

Lorsque le macrobloc en cours est codé P_Skip, toutes les valeurs de LumaLevel, ChromaDCLevel, ChromaACLevel sont mises égales à 0 pour le macrobloc en cours.

8.6.1 Processus de décodage SP pour images non commutatives

Ce processus est invoqué, lors du décodage de macroblocs de type P dans des tranches SP dans lesquelles sp_for_switch_flag est égal à 0.

Les entrées dans ce processus sont des échantillons d'interprédiction pour le macrobloc en cours, selon le § 8.4, et les niveaux de coefficient de transformée résiduelle de prédiction.

Les résultats de ce processus sont les échantillons du macrobloc en cours décodés avant le processus de filtrage de démontage de blocs.

Le présent paragraphe s'applique à tous les macroblocs dans les tranches SP dans lesquelles sp_for_switch_flag est égal à 0, excepté celles avec le mode de prédiction de macrobloc égal à Intra_4x4 ou Intra_16x16. Il ne s'applique pas aux tranches SI.

8.6.1.1 Processus de décodage de coefficient de transformée luma

Les entrées dans ce processus sont des échantillons luma d'interprédiction pour le macrobloc en cours $pred_L$ d'après le § 8.4, les niveaux de coefficient de transformée résiduelle de prédiction, LumaLevel, et l'indice du bloc luma 4x4 luma4x4BlkIdx.

La position de l'échantillon supérieur gauche du bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc en cours est déduite en invoquant le processus de balayage inverse de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (x, y).

Soit la variable p une matrice 4x4 d'échantillons de prédiction avec les éléments p_{ij} qu'on calcule comme suit.

$$p_{ij} = pred_L[\ x + j, y + i\ ] \quad \text{avec } i, j = 0..3 \tag{8-422}$$

La variable p est transformée, ce qui produit des coefficients de transformée c^p conformément à:

$$c^p = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \tag{8-423}$$

Le processus inverse de balayage de coefficient de transformée comme décrit au § 8.5.5 est invoqué avec LumaLevel[luma4x4BlkIdx] comme entrée et la matrice bidimensionnelle c^r avec les éléments c_{ij}^r comme résultat.

Les coefficients de transformée de prédiction résiduels c^r sont normalisés au moyen du paramètre de quantification QP_Y, et ajoutés au coefficient de transformées du bloc de prédiction c^p avec i, j = 0..3 comme suit.

$$c_{ij}^s = c_{ij}^p + (\ (\ (\ c_{ij}^r * LevelScale(\ QP_Y \% 6, i, j\ ) * A_{ij}\ ) << (\ QP_Y / 6\ )\ ) >> 10\ ) \tag{8-424}$$

où LevelScale(m, i, j) est spécifié dans l'équation 8-312, et où A_{ij} est spécifié comme:

$$A_{ij} = \begin{cases} 16 & \text{pour } (i,j) \in \{(0,0),(0,2),(2,0),(2,2)\}, \\ 25 & \text{pour } (i,j) \in \{(1,1),(1,3),(3,1),(3,3)\}, \\ 20 & \text{sinon;} \end{cases} \tag{8-425}$$

La fonction LevelScale2(m, i, j), utilisée dans les formules ci-dessous, est spécifiée comme:

$$LevelScale2(m,i,j) = \begin{cases} w_{m0} & \text{pour } (i,j) \in \{(0,0),(0,2),(2,0),(2,2)\}, \\ w_{m1} & \text{pour } (i,j) \in \{(1,1),(1,3),(3,1),(3,3)\}, \\ w_{m2} & \text{sinon;} \end{cases} \tag{8-426}$$

où le premier et le second indice de w sont les indices respectivement de rangée et de colonne de la matrice spécifiée comme:

$$w = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix} \tag{8-427}$$

La somme résultante, c^s, est quantifiée avec un paramètre de quantification QS_Y et avec i, j = 0..3 comme suit.

$$c_{ij} = Sign(\ c_{ij}^s\ ) * (\ (\ Abs(\ c_{ij}^s\ ) * LevelScale2(\ QS_Y \% 6, i, j\ ) + (\ 1 << (\ 14 + QS_Y / 6\ )\ )\ ) >> (\ 15 + QS_Y / 6\ )\ ) \tag{8-428}$$

Le processus de normalisation et de transformation pour les blocs de 4x4 échantillons résiduels, comme spécifié au § 8.5.10, est invoqué avec c comme entrée et r comme résultat.

La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit.

$$u_{ij} = Clip1_Y(r_{ij}) \text{ avec } i, j = 0..3 \quad (8\text{-}429)$$

Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.12 est invoqué avec luma4x4BlkIdx et u comme entrées.

8.6.1.2 Processus de décodage du coefficient de transformée chroma

Les entrées dans ce processus sont les échantillons chroma d'interprédiction pour le macrobloc en cours du § 8.4 et les niveaux de coefficient de transformée de prédiction résiduel, ChromaDCLevel et ChromaACLevel.

Ce processus est invoqué deux fois: une fois pour la composante Cb et une fois pour la composante Cr. La composante est mentionné en remplaçant C par Cb pour la composante Cb et C par Cr pour la composante Cr. Soit iCbCr qui désigne la composante chroma en cours.

Pour chaque bloc 4x4 de la composante chroma en cours indexée au moyen de chroma4x4BlkIdx avec chroma4x4BlkIdx égal à 0..3, on applique ce qui suit:

– La position de l'échantillon supérieur gauche d'un bloc chroma 4x4 avec l'indice chroma4x4BlkIdx au sein du macrobloc est déduite comme suit.

$$x = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) \quad (8\text{-}430)$$

$$y = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) \quad (8\text{-}431)$$

– Soit p une matrice 4x4 d'échantillons de prédiction avec des éléments p_{ij} qu'on calcule comme suit.

$$p_{ij} = pred_C[x + j, y + i] \quad \text{avec } i, j = 0..3 \quad (8\text{-}432)$$

– La matrice 4x4 p est transformée afin de produire les coefficients de transformée c^p(chroma4x4BlkIdx) au moyen de l'équation 8-423.

– La variable chromaList, qui est une liste de 16 entrées, est déduite. chromaList[0] est mis égal à 0. Les chromaList[k] avec les indices k = 1..15 sont spécifiées comme suit.

$$chromaList[k] = ChromaACLevel[iCbCr][chroma4x4BlkIdx][k - 1] \quad (8\text{-}433)$$

– Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.5, est invoqué avec chromaList comme entrée et la matrice 4x4 c^r comme résultat.

– Les coefficients de transformée de prédiction résiduels c^r sont normalisés au moyen du paramètre de quantification QP_C, et ajoutés aux coefficients de transformée du bloc de prédiction c^p avec i, j = 0..3, sauf pour la combinaison i = 0, j = 0, comme suit.

$$c_{ij}^s = c_{ij}^p(chroma4x4BlkIdx) + (((c_{ij}^r * LevelScale(QP_C \% 6, i, j) * A_{ij}) << (QP_C / 6)) >> 10) \quad (8\text{-}434)$$

– La somme résultante, c^s, est quantifiée avec un paramètre de quantification QS_C et avec i, j = 0..3, sauf pour la combinaison i = 0, j = 0, comme suit. La déduction de c_{00}(chroma4x4BlkIdx) est décrite plus loin dans le présent paragraphe.

$$c_{ij}(chroma4x4BlkIdx) = (Sign(c_{ij}^s) * (Abs(c_{ij}^s) * LevelScale2(QS_C \% 6, i, j) + (1 << (14 + QS_C / 6)))) >> (15 + QS_C / 6) \quad (8\text{-}435)$$

– Le processus de normalisation et de transformation pour les blocs de 4x4 échantillons résiduels, comme spécifié au § 8.5.10, est invoqué avec c(chroma4x4BlkIdx) comme entrée et r comme résultat.

– La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit.

$$u_{ij} = Clip1_C(r_{ij}) \text{ avec } i, j = 0..3 \quad (8\text{-}436)$$

– Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.12 est invoqué avec chroma4x4BlkIdx et u comme entrées.

Le calcul du niveau de coefficient de transformée DC c_{00}(chroma4x4BlkIdx) est spécifié comme suit. Les coefficients de transformée DC des quatre blocs 4x4 chroma de prédiction de la composante en cours du macrobloc sont assemblés en une matrice 2x2 avec des éléments c_{00}^{p}(chroma4x4BlkIdx) et une transformée 2x2 est appliquée aux coefficients de transformée DC comme suit.

$$dc^{p} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00}^{p}(0) & c_{00}^{p}(1) \\ c_{00}^{p}(2) & c_{00}^{p}(3) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-437}$$

Les niveaux de coefficient de transformée résiduel de prédiction DC chroma, ChromaDCLevel[iCbCr][k] avec k = 0..3 sont normalisés au moyen du paramètre de quantification QP, et ajoutés aux coefficients de transformée DC de prédiction comme suit.

$$dc_{ij}^{s} = dc_{ij}^{p} + (((\text{ChromaDCLevel}[iCbCr][j * 2 + i] * \text{LevelScale}(QP_C \% 6, 0, 0) * A_{00}) << (QP_C / 6)) >> 9) \quad \text{avec } i, j = 0, 1 \tag{8-438}$$

La matrice 2x2 dc^{s}, est quantifiée au moyen du paramètre de quantification QS_C comme suit:

$$dc_{ij}^{r} = (\text{Sign}(dc_{ij}^{s}) * (\text{Abs}(dc_{ij}^{s}) * \text{LevelScale2}(QS_C \% 6, 0, 0) + (1 << (15 + QS_C / 6)))) >> (16 + QS_C / 6) \quad \text{avec } i, j = 0, 1 \tag{8-439}$$

La matrice 2x2 f avec les éléments f_{ij} et i, j = 0..1 est déduite comme suit.

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} dc_{00}^{r} & dc_{01}^{r} \\ dc_{10}^{r} & dc_{11}^{r} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-440}$$

La normalisation des éléments f_{ij} de f est effectuée comme suit:

$$c_{00}(j * 2 + i) = ((f_{ij} * \text{LevelScale}(QS_C \% 6, 0, 0)) << (QS_C / 6)) >> 5 \quad \text{avec } i, j = 0, 1 \tag{8-441}$$

8.6.2 Processus de décodage de tranche SP et SI pour images commutatives

Ce processus est invoqué, lors du décodage de macroblocs de type P dans des tranches SP dans lesquelles sp_for_switch_flag est égal à 1 et lors du décodage de macroblocs de type SI dans des tranches SI.

Les entrées dans ce processus sont les niveaux de coefficient de transformée résiduel de prédiction et les matrices d'échantillons de prédiction $pred_L$, $pred_{Cb}$, $pred_{Cr}$ pour le macrobloc en cours.

8.6.2.1 Processus de décodage de coefficient de transformée luma

Les entrées dans ce processus sont des échantillons luma de prédiction $pred_L$ et les niveaux de coefficient de transformée résiduel de prédiction, LumaLevel.

La matrice 4x4 p avec des éléments p_{ij} avec i, j = 0..3, est calculée comme au § 8.6.1.1. Elle est transformée conformément à l'équation 8-423 afin de produire les coefficients de transformée c^{p}. Ces coefficients de transformée sont alors quantifiés avec le paramètre de quantification QS_Y, comme suit:

$$c_{ij}^{s} = \text{Sign}(c_{ij}^{p}) * ((\text{Abs}(c_{ij}^{p}) * \text{LevelScale2}(QS_Y \% 6, i, j) + (1 << (14 + QS_Y / 6))) >> (15 + QS_Y / 6)) \quad \text{avec } i, j = 0..3 \tag{8-442}$$

Le processus inverse de balayage de coefficient de transformée décrit au § 8.5.5 est invoqué avec LumaLevel[luma4x4BlkIdx] comme entrée et la matrice bidimensionnelle c^{r} avec les éléments c_{ij}^{r} comme résultat.

La matrice 4x4 c avec les éléments c_{ij} avec i, j = 0..3 est déduite comme suit.

$$c_{ij} = c_{ij}^{r} + c_{ij}^{s} \quad \text{avec } i, j = 0..3 \tag{8-443}$$

Le processus de normalisation et de transformation pour blocs de 4x4 échantillons résiduels, comme spécifié au § 8.5.10, est invoqué avec c comme entrée et r comme résultat.

La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit.

$$u_{ij} = Clip1_Y(r_{ij}) \text{ avec } i, j = 0..3 \quad (8\text{-}444)$$

Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.12 est invoqué avec luma4x4BlkIdx et u comme entrées.

8.6.2.2 Processus de décodage du coefficient de transformée chroma

Les entrées dans ce processus sont des échantillons chroma prédits pour le macrobloc en cours selon le § 8.4 et les niveaux de coefficient de transformée résiduel de prédiction, ChromaDCLevel et ChromaACLevel.

Ce processus est invoqué deux fois: une fois pour la composante Cb et une fois pour la composante Cr. La composante est mentionnée en remplaçant C par Cb pour la composante Cb et C par Cr pour la composante Cr. Soit iCbCr qui choisit la composante chroma en cours.

Pour chaque bloc 4x4 de la composante chroma en cours indexé au moyen de chroma4x4BlkIdx avec chroma4x4BlkIdx égal à 0..3, on applique ce qui suit.

1. La matrice 4x4 p avec les éléments p_{ij} avec i, j = 0..3 est déduite comme au § 8.6.1.2. Elle est transformée conformément à l'équation 8-423 afin de produire des coefficients de transformée c^p(chroma4x4BlkIdx). Ces coefficients de transformée sont alors quantifiés avec le paramètre de quantification QS_C, avec i, j = 0..3, sauf pour la combinaison i = 0, j = 0 comme suit. Le traitement de c_{00}^p(chroma4x4BlkIdx) est décrit ci-après dans le présent paragraphe.

$$c_{ij}^s = (Sign(c_{ij}^p(chroma4x4BlkIdx)) * (Abs(c_{ij}^p(chroma4x4BlkIdx)) * LevelScale2(QS_C \% 6, i, j) + (1 << (14 + QS_C / 6)))) >> (15 + QS_C / 6) \quad (8\text{-}445)$$

– On calcule la variable chromaList, qui est une liste de 16 entrées. chromaList[0] est mis égal à 0. Les chromaList[k] avec l'indice k = 1..15 sont spécifiées comme suit:

$$chromaList[k] = ChromaACLevel[iCbCr][chroma4x4BlkIdx][k - 1] \quad (8\text{-}446)$$

– Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.5, est invoqué avec chromaList comme entrée et la matrice bidimensionnelle c^r(chroma4x4BlkIdx) avec les éléments c_{ij}^r(chroma4x4BlkIdx) comme résultat.

– La matrice 4x4 c(chroma4x4BlkIdx) avec les éléments c_{ij}(chroma4x4BlkIdx) avec i, j = 0..3, sauf pour la combinaison i = 0, j = 0 est déduite comme suit. La déduction de c_{00}(chroma4x4BlkIdx) est décrite ci-dessous.

$$c_{ij}(chroma4x4BlkIdx) = c_{ij}^r(chroma4x4BlkIdx) + c_{ij}^s \quad (8\text{-}447)$$

– Le processus de normalisation et de transformation pour les blocs de 4x4 échantillons résiduels, comme spécifié au § 8.5.10, est invoqué avec c(chroma4x4BlkIdx) comme entrée et r comme résultat.

– La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit:

$$u_{ij} = Clip1_C(r_{ij}) \text{ avec } i, j = 0..3 \quad (8\text{-}448)$$

– Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.12 est invoqué avec chroma4x4BlkIdx et u comme entrées.

Le calcul du niveau DC de coefficient de transformée c_{00}(chroma4x4BlkIdx) est spécifié comme suit. Les coefficients de transformée DC des 4 blocs 4x4 de prédiction chroma de la composante en cours du macrobloc, c_{00}^p(chroma4x4BlkIdx), sont assemblés en une matrice 2x2, et une transformée 2x2 est appliquée aux coefficients de transformée DC de ces blocs conformément à l'équation (8-437) résultant en coefficients de transformée DC dc_{ij}^p.

Ces coefficients de transformée DC sont alors quantifiés avec le paramètre de quantification QS_C, tel que donné par:

$$dc_{ij}^s = (Sign(dc_{ij}^p) * (Abs(dc_{ij}^p) * LevelScale2(QS_C \% 6, 0, 0) + (1 << (15 + QS_C / 6)))) >> (16 + QS_C / 6) \text{ avec } i, j = 0, 1 \quad (8\text{-}449)$$

Les coefficients de transformée résiduels de prédiction DC chroma analysés syntaxiquement, ChromaDCLevel[iCbCr][k] avec k = 0..3, sont ajoutés à ces coefficients de transformée DC quantifiés du bloc de prédiction, comme donné par:

$$dc_{ij}^{r} = dc_{ij}^{s} + ChromaDCLevel[\ iCbCr\ ][\ j * 2 + i\ ] \quad \text{avec } i, j = 0, 1 \tag{8-450}$$

La matrice 2x2 f avec les éléments f_{ij} et i, j = 0..1 est calculée au moyen de l'équation 8-440.

La matrice 2x2 f avec les éléments f_{ij} et i, j = 0..1 est copiée comme suit.

$$c_{00}(\ j * 2 + i\ ) = f_{ij} \quad \text{avec } i, j = 0, 1 \tag{8-451}$$

8.7 Processus de filtrage de démontage de blocs

Un processus de filtrage conditionnel est appliqué à toutes les bordures de bloc NxN (où N = 4 ou N = 8 pour luma et N = 4 pour chroma) d'une image, sauf les bordures qui sont à la frontière de l'image et toute bordure pour laquelle le processus de filtrage de démontage de blocs est désactivé par disable_deblocking_filter_idc, comme spécifié ci-dessous. Ce processus de filtrage est effectué sur la base du macrobloc une fois terminé le processus de construction d'image précédant le processus de filtrage de démontage de blocs (spécifié aux § 8.5 et 8.6) pour l'image décodée entière, avec tous les macroblocs d'une image traités dans l'ordre croissant des adresses de macroblocs.

NOTE 1 – Avant d'effectuer le processus de filtrage de démontage de blocs pour chaque macrobloc, les échantillons sortis des blocs du macrobloc ou de la paire de macroblocs de dessus (le cas échéant) et du macrobloc ou de la paire de macroblocs à gauche (le cas échéant) du macrobloc en cours, sont toujours disponibles du fait que le processus de filtrage de démontage de blocs est effectué une fois terminé le processus de construction d'image précédant le processus de filtrage de démontage de blocs pour l'image décodée entière. Cependant, aux fins de la détermination des bordures qui doivent être filtrées lorsque disable_deblocking_filter_idc est égal à 2, les macroblocs contenus dans différentes tranches sont considérés comme indisponibles pendant les étapes spécifiées du fonctionnement du processus de filtre de démontage de bloc.

Le processus de filtre de démontage de bloc est invoqué séparément pour les composantes luma et chroma. Pour chaque macrobloc et chaque composante, les bordures verticales sont filtrées tout d'abord, à partir de la bordure située à gauche des macroblocs en allant vers les bordures situées à droite du macrobloc, dans leur ordre géométrique. Puis les bordures horizontales sont filtrées à partir de la bordure située au sommet du macrobloc en allant vers les bordures situées en bas du macrobloc, dans leur ordre géométrique. La Figure 8-10 montre les bordures d'un macrobloc qui peuvent être interprétées comme des bordures luma ou chroma.

Lorsque l'on interprète les bordures de la Figure 8-10 comme des bordures luma, selon le fanion transform_size_8x8_flag, ce qui suit est applicable:

– Si transform_size_8x8_flag est égal à 0, les deux types de bordure luma: en traits gras continus et en traits gras discontinus, sont filtrés.

– Sinon (si transform_size_8x8_flag est égal à 1), seules les bordures luma en trait gras continu sont filtrées.

Lorsque l'on interprète les bordures de la Figure 8-10 en tant que bordures chroma, selon chroma_format_idc, ce qui suit est applicable:

– Si chroma_format_idc est égal à 1 (format 4:2:0), seules les bordures chroma en trait gras continu sont filtrées.

– Sinon, si chroma_format_idc est égal à 2 (format 4:2:2), les bordures chroma verticales en trait gras continu sont filtrées et les deux types de bordure chroma, en trait gras continu et en trait gras discontinu, sont filtrés.

– Sinon, si chroma_format_idc est égal à 3 (format 4:4:4), les deux types de bordure chroma, en trait gras continu et en trait gras discontinu, sont filtrés.

– Sinon (si chroma_format_idc est égal à 0 (format monochrome)), aucune bordure chroma n'est filtrée.

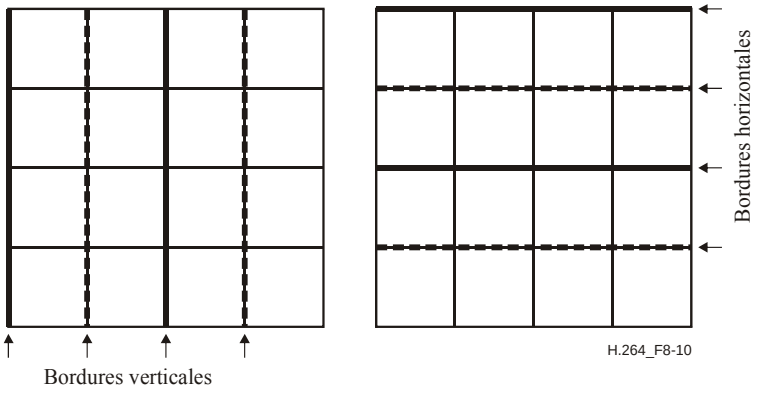


Figure 8-10 – Frontières dans un macrobloc à filtrer

Pour l'adresse du macrobloc actuel CurrMbAddr en allant vers les valeurs 0..PicSizeInMbs – 1, ce qui suit est applicable:

1. Le processus de calcul pour les macroblocs voisins, spécifié dans le § 6.4.8.1, est invoqué et la sortie est attribuée à mbAddrA et mbAddrB.
2. Les variables fieldModeMbFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag et filterTopMbEdgeFlag sont déduites comme suit.
   - La variable fieldModeMbFlag est déduite comme suit.
     - Si l'une des conditions suivantes est vraie, fieldModeMbFlag est mis égal à 1.
       - field_pic_flag est égal à 1.
       - MbaffFrameFlag est égal à 1 et le macrobloc CurrMbAddr est un macrobloc de sous-trame.
     - Sinon, fieldModeMbFlag est mis égal à 0.
   - La variable filterInternalEdgesFlag est déduite comme suit.
     - si disable_deblocking_filter_idc pour la tranche qui contient le macrobloc CurrMbAddr est égal à 1, la variable filterInternalEdgesFlag est mise égale à 0;
     - sinon (si disable_deblocking_filter_idc pour la tranche qui contient le macrobloc CurrMbAddr n'est pas égal à 1), la variable filterInternalEdgesFlag est mise égale à 1.
   - La variable filterLeftMbEdgeFlag est déduite comme suit.
     - Si l'une des conditions suivantes est vraie, la variable filterLeftMbEdgeFlag est mise égale à 0.
       - MbaffFrameFlag est égal à 0 et CurrMbAddr % PicWidthInMbs est égal à 0.
       - MbaffFrameFlag est égal à 1 et (CurrMbAddr >> 1) % PicWidthInMbs est égal à 0.
       - disable_deblocking_filter_idc pour la tranche qui contient le macrobloc CurrMbAddr est égal à 1.
       - disable_deblocking_filter_idc pour la tranche qui contient le macrobloc CurrMbAddr est égal à 2 et le macrobloc mbAddrA est indisponible.
     - Sinon, la variable filterLeftMbEdgeFlag est mise égale à 1.
   - La variable filterTopMbEdgeFlag est déduite comme suit.
     - Si l'une des conditions suivantes est vraie, la variable filterTopMbEdgeFlag est mise égale à 0.
       - MbaffFrameFlag est égal à 0 et CurrMbAddr est inférieur à PicWidthInMbs.
       - MbaffFrameFlag est égal à 1, (CurrMbAddr >> 1) est inférieur à PicWidthInMbs, et le macrobloc CurrMbAddr est un sous-macrobloc de trames.
       - MbaffFrameFlag est égal à 1, (CurrMbAddr >> 1) est inférieur à PicWidthInMbs, le macrobloc CurrMbAddr est un macrobloc de trames, et CurrMbAddr % 2 est égal à 0.
       - disable_deblocking_filter_idc pour la tranche qui contient le macrobloc CurrMbAddr est égal à 1.
       - disable_deblocking_filter_idc pour la tranche qui contient le macrobloc CurrMbAddr est égal à 2 et le macrobloc mbAddrB est indisponible.
     - Sinon, la variable filterTopMbEdgeFlag est mise égale à 1.
3. Etant donné les variables fieldModeMbFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag et filterTopMbEdgeFlag, le filtrage de démontage de blocs est commandé comme suit.
   - Lorsque filterLeftMbEdgeFlag est égal à 1, le filtrage de la bordure luma verticale gauche est spécifié comme suit:
     - Le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 1, sous-trameModeFilteringFlag = sous-trameModeMbFlag, et $(xE_k, yE_k) = (0, k)$ avec k = 0..15 en entrée et S'_L comme résultat.
   - Lorsque filterInternalEdgesFlag est égal à 1, le filtrage des bordures luma verticales internes est spécifié comme suit:
     - Lorsque transform_size_8x8_flag est égal à 0, le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (4, k)$ avec k = 0..15 en entrée et S'_L comme résultat;
     - le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (8, k)$ avec k = 0..15 en entrée et S'_L comme résultat;

- lorsque transform_size_8x8_flag est égal à 0, le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (12, k)$ avec k = 0..15 en entrée et S'_L comme résultat.

- Lorsque filterTopMbEdgeFlag est égal à 1, le filtrage de la bordure luma horizontale supérieure est spécifié comme suit.
  - Si MbaffFrameFlag est égal à 1, (CurrMbAddr % 2) est égal à 0, CurrMbAddr est supérieur ou égal à 2 * PicWidthInMbs, le macrobloc CurrMbAddr est un macrobloc de trame, et le macrobloc (CurrMbAddr – 2 * PicWidthInMbs + 1) est un macrobloc de sous-trame, on applique ce qui suit:
    - le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFiltrageFlag = 1, et $(xE_k, yE_k) = (k, 0)$ avec k = 0..15 en entrée et S'_L comme résultat;
    - le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFiltrageFlag = 1, et $(xE_k, yE_k) = (k, 1)$ avec k = 0..15 en entrée et S'_L comme résultat.
  - Sinon, le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 0)$ avec k = 0..15 en entrée et S'_L comme résultat.

- Lorsque filterInternalEdgesFlag est égal à 1, le filtrage des bordures luma horizontales internes est spécifié comme suit.
  - lorsque transform_size_8x8_flag est égal à 0, le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 4)$ avec k = 0..15 en entrée et S'_L comme résultat;
  - le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 8)$ avec k = 0..15 en entrée et S'_L comme résultat;
  - lorsque transform_size_8x8_flag est égal à 0, le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 0, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 12)$ avec k = 0..15 en entrée et S'_L comme résultat.

- Pour le filtrage des deux composantes chroma avec iCbCr = 0 pour Cb et iCbCr = 1 pour Cr, on applique ce qui suit.
  - Lorsque filterLeftMbEdgeFlag est égal à 1, le filtrage de la bordure verticale gauche est spécifié comme suit.
    - le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (0, k)$ avec k = 0..MbHeightC – 1 en entrée, et S'_C avec C remplacé par Cb pour iCbCr = 0 et C remplacé par Cr pour iCbCr = 1 comme résultat.
  - Lorsque filterInternalEdgesFlag est égal à 1, le filtrage de la bordure chroma verticale interne est spécifié comme suit.
    - le processus spécifié au § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (4, k)$ avec k = 0..MbHeightC – 1 en entrée, et S'_C avec C remplacé par Cb pour iCbCr = 0 et C remplacé par Cr pour iCbCr = 1 comme résultat.
    - Lorsque chroma_format_idc est égal à 3, le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFiltrageFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (8, k)$ avec k = 0..MbHeightC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.
    - Lorsque chroma_format_idc est égal à 3, le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFiltrageFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (12, k)$ avec k = 0..MbHeightC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.
  - Lorsque filterTopMbEdgeFlag est égal à 1, le filtrage du bordure chroma horizontale du sommet est spécifiée comme suit:
    - Si MbaffFrameFlag est égal à 1, (CurrMbAddr % 2) est égal à 0, CurrMbAddr est plus grand ou égal à 2 * PicWidthInMbs, le macrobloc CurrMbAddr est un macrobloc de trames, et le macrobloc (CurrMbAddr – 2 * PicWidthInMbs + 1) est un sous-macrobloc de trames, ce qui suit est applicable:

– Le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFiltrageFlag = 1, et $(xE_k, yE_k) = (k, 0)$ avec k = 0..MbWidthC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.

– Le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFiltrageFlag = 1, et $(xE_k, yE_k) = (k, 1)$ avec k = 0..MbWidthC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.

– Sinon, le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFiltrageFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 0)$ avec k = 0..MbWidthC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.

– Lorsque filterInternalEdgesFlag est égal à 1, le filtrage de la bordure chromatique horizontale interne est spécifié comme suit:

– Le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFiltrageFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 4)$ avec k = 0..MbWidthC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.

– Lorsque chroma_format_idc n'est pas égal à 1, le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFiltrageFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 8)$ avec k = 0..MbWidthC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 en sortie.

– Lorsque chroma_format_idc n'est pas égal à 1, le processus spécifié dans le § 8.7.1 est invoqué avec chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFiltrageFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 12)$ avec k = 0..MbWidthC – 1 en entrée et S'_C avec C étant remplacé par Cb pour iCbCr = 0 et C étant remplacé par Cr pour iCbCr = 1 comme résultat.

NOTE 2 – Lorsque le filtrage de mode de sous-trame (fieldModeFilteringFlag est égal à 1) est appliqué sur les bordures horizontales supérieures d'un macrobloc de trame, ce filtrage vertical sur la frontière supérieure ou inférieure du macrobloc peut impliquer certains échantillons qui s'étendent au-delà de la bordure interne d'un bloc qui est aussi filtré en interne en mode de trame.

NOTE 3 – Par exemple, en format chromatique 4:2:0, lorsque le fanion transform_size_8x8_flag est égal à 0, les bordures suivantes sont applicables: trois bordures luma horizontales, une bordure chroma horizontale pour Cb, et une bordure chroma horizontale pour Cr, qui sont internes à un macrobloc. Lorsque le filtrage de mode de sous-trame (fieldModeFilteringFlag est égal à 1) est appliqué aux bordures supérieures d'un macrobloc de trame, on filtre deux bordures luma horizontales, deux bordures chroma horizontales pour Cb, et deux bordures chroma horizontales pour Cr entre le macrobloc de trame et la paire de macroblocs ci-dessus, au moyen du filtrage de mode de sous-trame pour au total jusqu'à cinq bordures luma horizontales, trois bordures chroma horizontales pour Cb, et trois bordures chroma horizontales pour Cr filtrées qui sont considérées comme contrôlées par le macrobloc de trame. Dans tous les autres cas, au plus quatre bordures luma horizontales, deux chroma horizontales pour Cb, et deux chroma horizontales pour Cr sont filtrées et considérées comme contrôlées par un macrobloc particulier.

Finalement, les matrices S'_L, S'_{Cb}, S'_{Cr} sont respectivement allouées aux matrices S_L, S_{Cb}, S_{Cr} (qui représentent l'image décodée).

8.7.1 Processus de filtrage pour bordures de bloc

Les entrées dans ce processus sont chromaEdgeFlag, l'indice de composante chromatique iCbCr (lorsque chromaEdgeFlag est égal à 1), verticalEdgeFlag, fieldModeFilteringFlag, et un ensemble de nE localisations d'échantillon (xE_k, yE_k), avec k = 0..nE – 1, exprimé par rapport au coin supérieur gauche du macrobloc CurrMbAddr. L'ensemble de localisations d'échantillon (xE_k, yE_k) représente les localisations d'échantillon immédiatement à la droite d'une bordure verticale (lorsque verticalEdgeFlag est égal à 1) ou immédiatement au-dessous d'une bordure horizontale (lorsque verticalEdgeFlag est égal à 0).

La variable nE est déduite comme suit:

– si chromaEdgeFlag est égal à 0, nE est mis égal à 16;

– Sinon (si chromaEdgeFlag est égal à 1), nE est mis égale à (verticalEdgeFlag == 1) ? MbHeightC : MbWidthC.

Soit s' une variable spécifiant une matrice d'échantillons luma ou chroma, calculée comme suit:

– si chromaEdgeFlag est égal à 0, s' représente la matrice d'échantillons luma S'_L de l'image en cours;

– sinon, si chromaEdgeFlag est égal à 1 et iCbCr est égal à 0, s' représente la matrice d'échantillons chroma S'_{Cb} de la composante chroma Cb de l'image en cours;

– Sinon (si chromaEdgeFlag est égal à 1 et iCbCr est égal à 1), s' représente la matrice d'échantillons chroma S'_{Cr} de la composante chroma Cr de l'image en cours.

La variable dy est déduite comme suit:

– si fieldModeFilteringFlag est égal à 1 et MbaffFrameFlag est égal à 1, dy est mis égal à 2;

– Sinon (si fieldModeFilteringFlag est égal à 0 ou MbaffFrameFlag est égal à 0), dy est mis égal à 1.

La position de l'échantillon luma supérieur gauche du macrobloc CurrMbAddr est déduite en invoquant le processus de balayage inverse de macrobloc du § 6.4.1 avec CurrMbAddr en entrée et le résultat étant alloué à (xI, yI).

Les variables xP et yP sont calculées comme suit:

– Si chromaEdgeFlag est égal à 0, xP est mis à xI et yP est mis à yI.

– Sinon (si chromaEdgeFlag est égal à 1), xP est mis à xI / SubWidthC et yP est mis à (yI + SubHeightC – 1) / SubHeightC.

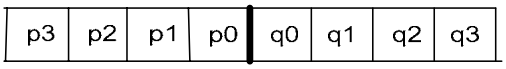


Figure 8-11 – Convention visant à décrire les échantillons sur une frontière de bloc 4x4 horizontale ou verticale

Pour chaque localisation d'échantillon (xE_k, yE_k), k = 0 .. nE – 1, on applique ce qui suit.

– Le processus de filtrage s'applique à un ensemble de huit échantillons sur une bordure horizontale ou verticale de bloc 4x4 noté p_i et q_i avec i = 0..3 comme indiqué à la Figure 8-11 avec la bordure se tenant entre p_0 et q_0. p_i et q_i avec i = 0..3 sont spécifiés comme suit:

  – si verticalEdgeFlag est égal à 1,

$$q_i = s'[\ xP + xE_k + i,\ yP + dy * yE_k\ ] \quad (8\text{-}452)$$

$$p_i = s'[\ xP + xE_k – i – 1,\ yP + dy * yE_k\ ] \quad (8\text{-}453)$$

  – sinon (si verticalEdgeFlag est égal à 0),

$$q_i = s'[\ xP + xE_k,\ yP + dy * (\ yE_k + i\ ) – (yE_k \% 2\ )\ ] \quad (8\text{-}454)$$

$$p_i = s'[\ xP + xE_k,\ yP + dy * (\ yE_k – i – 1\ ) – (yE_k \% 2\ )\ ] \quad (8\text{-}455)$$

– Le processus spécifié au § 8.7.2 est invoqué avec les valeurs d'échantillon p_i et q_i (i = 0..3), chromaEdgeFlag, verticalEdgeFlag, et fieldModeFilteringFlag en entrée, et le résultat est alloué aux valeurs d'échantillon résultantes filtrées p'_i et q'_i avec i = 0..2.

– Les valeurs d'échantillon d'entrée p_i et q_i avec i = 0..2 sont remplacées par les valeurs d'échantillon résultantes filtrées correspondantes p'_i et q'_i avec i = 0..2 au sein de la matrice d'échantillon s' comme suit:

  – si verticalEdgeFlag est égal à 1,

$$s'[\ xP + xE_k + i,\ yP + dy * yE_k\ ] = q'_i \quad (8\text{-}456)$$

$$s'[\ xP + xE_k – i – 1,\ yP + dy * yE_k\ ] = p'_i \quad (8\text{-}457)$$

  – sinon (si verticalEdgeFlag est égal à 0),

$$s'[\ xP + xE_k,\ yP + dy * (\ yE_k + i\ ) – (\ yE_k \% 2\ )\ ] = q'_i \quad (8\text{-}458)$$

$$s'[\ xP + xE_k,\ yP + dy * (\ yE_k – i – 1\ ) – (\ yE_k \% 2\ )\ ] = p'_i \quad (8\text{-}459)$$

8.7.2 Processus de filtrage pour un ensemble d'échantillons sur une bordure horizontale ou verticale de bloc

Les entrées dans ce processus sont les valeurs d'échantillon d'entrée p_i et q_i avec i dans la gamme de 0..3 d'un ensemble unique d'échantillons sur une bordure à filtrer, chromaEdgeFlag, verticalEdgeFlag, et fieldModeFilteringFlag.

Les résultats de ce processus sont les valeurs d'échantillon résultantes filtrées p'_i et q'_i avec i dans la gamme de 0..2.

La variable dépendante du contenu de force du filtrage de frontière bS est déduite comme suit.

– si chromaEdgeFlag est égal à 0, le processus de déduction pour la force du filtrage de frontière dépendant du contenu spécifiée au § 8.7.2.1 est invoqué avec p_0, q_0, et verticalEdgeFlag en entrée, et le résultat est alloué à bS;

– Sinon (si chromaEdgeFlag est égal à 1), la variable bS utilisée pour le filtrage d'un ensemble d'échantillons d'une bordure chroma horizontale ou verticale est mise égale à la valeur de bS pour le filtrage de l'ensemble d'échantillons d'une bordure luma, horizontale ou verticale selon le cas, qui contient l'échantillon luma à l'emplacement (SubWidthC * x, SubHeightC * y) à l'intérieur de la matrice d'échantillons luma de la même sous-trame, où (x, y) est la localisation de l'échantillon chroma q_0 au sein de la matrice chroma pour cette sous-trame.

Le processus spécifié au § 8.7.2.2 est invoqué avec p_0, q_0, p_1, q_1, chromaEdgeFlag, et bS en entrée, et le résultat est alloué à filterSamplesFlag, indexA, α, et β.

En fonction de la variable filterSamplesFlag, on applique ce qui suit.

– si filterSamplesFlag est égal à 1, on applique ce qui suit.
  – si bS est inférieur à 4, le processus spécifié au § 8.7.2.3 est invoqué avec p_i et q_i (i = 0..2), chromaEdgeFlag, bS, β, et indexA donnés en entrée, et le résultat est alloué à p'_i et q'_i (i = 0..2);
  – sinon (si bS est égal à 4), le processus spécifié au § 8.7.2.4 est invoqué avec p_i et q_i (i = 0..3), chromaEdgeFlag, α, et β donnés en entrée, et le résultat est alloué à p'_i et q'_i (i = 0..2).
– Sinon (si filterSamplesFlag est égal à 0), les échantillons résultants filtrés p'_i et q'_i (i = 0..2) sont remplacés par les échantillons d'entrée correspondants p_i et q_i:

$$\text{pour } i = 0..2, \qquad p'_i = p_i \tag{8-460}$$

$$\text{pour } i = 0..2, \qquad q'_i = q_i \tag{8-461}$$

8.7.2.1 Processus de déduction pour force de filtrage de frontière luma dépendant du contenu

Les entrées dans ce processus sont les valeurs d'échantillon d'entrée p_0 et q_0 d'un ensemble unique d'échantillons sur une bordure à filtrer et verticalEdgeFlag.

Le résultat de ce processus est la variable bS.

Soit la variable mixedModeEdgeFlag à calculer comme suit:

– si MbaffFrameFlag est égal à 1 et que les échantillons p_0 et q_0 sont dans différentes paires de macroblocs, dont l'une est une paire de macroblocs de sous-trame et l'autre une paire de macroblocs de trame, mixedModeEdgeFlag est mis égal à 1;

– sinon, mixedModeEdgeFlag est mis égal à 0.

La variable bS est déduite comme suit.

– Si la bordure de bloc est aussi une bordure de macrobloc et si une des conditions suivantes est vraie, une valeur de bS égale à 4 est le résultat:
  – les échantillons p_0 et q_0 sont tous deux dans des macroblocs de trame et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc codé au moyen d'un mode d'intraprédiction de macrobloc;
  – les échantillons p_0 et q_0 sont tous deux dans des macroblocs de trame et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc qui est dans une tranche avec slice_type égal à SP ou SI;
  – MbaffFrameFlag est égal à 1 ou field_pic_flag est égal à 1, et verticalEdgeFlag est égal à 1, et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc codé au moyen d'un mode d'intraprédiction de macrobloc.
  – MbaffFrameFlag est égal à 1 ou field_pic_flag est égal à 1, et verticalEdgeFlag est égal à 1, et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc qui est dans une tranche avec slice_type égal à SP ou SI;

– Sinon, si une des conditions suivantes est vraie, une valeur de bS égale à 3 est le résultat:

  – mixedModeEdgeFlag est égal à 0 et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc codé au moyen d'un mode d'intraprédiction de macrobloc;

  – mixedModeEdgeFlag est égal à 0 et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc qui est dans une tranche avec slice_type égal à SP ou SI.

  – mixedModeEdgeFlag est égal à 1, verticalEdgeFlag est égal à 0, et l'un ou l'autre des échantillons p_0 ou q_0 ou les deux, sont dans un macrobloc codé au moyen d'un mode d'intraprédiction de macrobloc.

  – mixedModeEdgeFlag est égal à 1, verticalEdgeFlag est égal à 0, et l'un ou l'autre des échantillons p_0 ou q_0, ou les deux, sont dans un macrobloc qui est dans une tranche avec slice_type égal à SP ou SI.

– Sinon, si la condition suivante est vraie, une valeur de bS égale à 2 est le résultat:

  – le bloc luma contenant l'échantillon p_0 ou le bloc luma contenant l'échantillon q_0 contient des niveaux de coefficient de transformée différents de zéro.

– Sinon, si une des conditions suivantes est vraie, une valeur de bS égale à 1 est le résultat:

  – mixedModeEdgeFlag est égal à 1;

  – mixedModeEdgeFlag est égal à 0 et on utilise pour la prédiction de la subdivision de macrobloc/sous-macrobloc contenant l'échantillon p_0 des images de référence différentes ou des numéros de vecteurs cinétiques différents de ceux de la prédiction de la subdivision de macrobloc/sous-macrobloc contenant l'échantillon q_0,

    NOTE 1 – La question de savoir si les images de référence utilisées pour les deux subdivisions de macrobloc/sous-macrobloc sont identiques ou différentes est uniquement fonction des images qui sont mentionnées, indépendamment de savoir, d'une part, si une prédiction est établie au moyen d'un indice de la liste 0 d'images de référence ou d'un indice de la liste 1 d'images de référence et, d'autre part, si la position d'indice dans une liste d'images de référence est différente ou non.

  – mixedModeEdgeFlag est égal à 0 et un vecteur cinétique est utilisé afin de prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 et un vecteur cinétique est utilisé afin de prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon q_0 et la différence absolue entre la composante horizontale ou verticale du vecteur cinétique utilisé est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma;

  – mixedModeEdgeFlag est égal à 0 et deux vecteurs cinétiques et deux images de référence différentes sont utilisés afin de prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 et deux vecteurs cinétiques pour les deux mêmes images de référence sont utilisés afin de prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon q_0 et la différence absolue entre la composante horizontale ou verticale d'un vecteur cinétique utilisé dans la prédiction des deux macroblocs/subdivision de sous-macroblocs pour la même image de référence est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma;

  – mixedModeEdgeFlag est égal à 0 et deux vecteurs cinétiques pour la même image de référence sont utilisés afin de prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 et deux vecteurs cinétiques pour la même image de référence sont utilisés afin de prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon q_0 et les deux conditions suivantes sont vraies:

    – la différence absolue entre la composante horizontale ou verticale des vecteurs cinétiques de liste 0 utilisés dans la prédiction des deux macroblocs/subdivisions de sous-macroblocs est supérieure ou égale à 4 en quarts d'échantillon de trame luma ou la différence absolue entre la composante horizontale ou verticale des vecteurs cinétiques de liste 1 utilisés dans la prédiction des deux macroblocs/subdivisions de sous-macroblocs est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma;

    – la différence absolue entre la composante horizontale ou verticale du vecteur cinétique de liste 0 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon p_0 et le vecteur cinétique de liste 1 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon q_0 est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma ou la différence absolue entre la composante horizontale ou verticale du vecteur cinétique de liste 1 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon p_0 et le vecteur cinétique de liste 0 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon q_0 est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma.

      NOTE 2 – Une différence verticale de 4 en unités de quart d'échantillon de trame luma est une différence de 2 en unités de quart d'échantillon de sous-trame luma.

– Sinon, une valeur de bS égale à 0 est le résultat.

8.7.2.2 Processus de déduction pour les seuils de chaque bordure de bloc

Les entrées dans ce processus sont les valeurs d'échantillon d'entrée p_0, q_0, p_1 et q_1 d'un ensemble d'échantillons sur une bordure à filtrer, chromaEdgeFlag, et bS, pour l'ensemble d'échantillons d'entrée, comme spécifié au § 8.7.2.

Les résultats de ce processus sont la variable filterSamplesFlag, qui indique si les échantillons d'entrée sont filtrés, la valeur de indexA, et les valeurs des variables de seuil α et β.

Soit qP_p et qP_q les variables spécifiant les valeurs des paramètres de quantification pour les macroblocs contenant respectivement les échantillons p_0 et q_0. Les variables qP_z (avec z remplacé par p ou q) sont déduites comme suit:

– Si chromaEdgeFlag est égal à 0, on applique ce qui suit.

  – Si le macrobloc contenant l'échantillon z_0 est un macrobloc I_PCM, qP_z est mis à 0;

  – Sinon (si le macrobloc contenant l'échantillon z_0 n'est pas un macrobloc I_PCM), qP_z est mis à la valeur de QP_Y du macrobloc contenant l'échantillon z_0.

– Sinon (si chromaEdgeFlag est égal à 1), on applique ce qui suit.

  – Si le macrobloc contenant l'échantillon z_0 est un macrobloc I_PCM, qP_z est mis à la valeur de QP_C qui correspond à une valeur de 0 pour QP_Y comme spécifié au § 8.5.7;

  – Sinon (si le macrobloc contenant l'échantillon z_0 n'est pas un macrobloc I_PCM), qP_z est mis à la valeur de QP_C qui correspond à la valeur QP_Y du macrobloc contenant l'échantillon z_0 comme spécifié au § 8.5.7.

Soit qPav une variable spécifiant un paramètre de quantification moyen. Elle est déduite comme suit.

$$qP_{av} = (qP_p + qP_q + 1) >> 1 \quad (8\text{-}462)$$

NOTE – Dans les tranches SP et SI, qP_{av} est calculée de la même façon que dans les autres types de tranche. QS_Y de l'équation 7-28 n'est pas utilisé dans le filtre de démontage de blocs.

Soit indexA une variable qui est utilisée afin d'accéder au tableau des décalages α (Tableau 8-16) aussi bien qu'au tableau t_{C0} (Tableau 8-17), qui est utilisé afin de filtrer les bordures avec bS inférieur à 4 comme spécifié au § 8.7.2.3, et soit indexB une variable qui est utilisée afin d'accéder au tableau des décalages β (Tableau 8-16). Les variables indexA et indexB sont déduites comme suit, lorsque les valeurs de FilterOffsetA et FilterOffsetB sont les valeurs de ces variables spécifiées au § 7.4.3 pour la tranche qui contient le macrobloc contenant l'échantillon q_0.

$$indexA = Clip3(0, 51, qP_{av} + FilterOffsetA) \quad (8\text{-}463)$$

$$indexB = Clip3(0, 51, qP_{av} + FilterOffsetB) \quad (8\text{-}464)$$

Les variables α' et β' sont spécifiées dans le Tableau 8-16 en fonction des valeurs de indexA et de indexB. Selon chromaEdgeFlag, les variables de seuil α et β correspondantes sont calculées comme suit:

– Si chromaEdgeFlag est égal à 0,

$$\alpha = \alpha' * (1 << (BitDepth_Y – 8)) \quad (8\text{-}465)$$

$$\beta = \beta' * (1 << (BitDepth_Y – 8)) \quad (8\text{-}466)$$

– Sinon (si chromaEdgeFlag est égal à 1),

$$\alpha = \alpha' * (1 << (BitDepth_C – 8)) \quad (8\text{-}467)$$

$$\beta = \beta' * (1 << (BitDepth_C – 8)) \quad (8\text{-}468)$$

La variable filterSamplesFlag est calculée par

$$filterSamplesFlag = (bS != 0 \ \&\& \ Abs(p_0 – q_0) < \alpha \ \&\& \ Abs(p_1 – p_0) < \beta \ \&\& \ Abs(q_1 – q_0) < \beta) \quad (8\text{-}469)$$

Tableau 8-16 – Calcul des variables de seuil dépendantes du décalage α' et β' à partir de indexA et de indexB

	indexA (pour α) ou indexB (pour β)																									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
α'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	5	6	7	8	9	10	12	13
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	3	3	3	3	4	4	4

Tableau 8-16 (fin) – Calcul des variables de seuil dépendantes du décalage α' et β' à partir de indexA et de indexB

	indexA (pour α) ou indexB (pour β)																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
α'	15	17	20	22	25	28	32	36	40	45	50	56	63	71	80	90	101	113	127	144	162	182	203	226	255	255
β'	6	6	7	7	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18

8.7.2.3 Processus de filtrage pour bordures avec bS inférieur à 4

Les entrées dans ce processus sont les valeurs d'échantillon d'entrée p_i et q_i (i = 0..2) d'un ensemble d'échantillons sur une bordure à filtrer, chromaEdgeFlag, bS, β, et indexA, pour l'ensemble d'échantillons d'entrée, comme spécifié au 8.7.2.

Les résultats de ce processus sont les valeurs d'échantillon filtré résultant p'_i et q'_i (i = 0..2) pour l'ensemble de valeurs d'échantillon d'entrée.

Les échantillons filtrés résultants p'_0 et q'_0 sont calculés par:

$$\Delta = \text{Clip3}(-t_C, t_C, ((((q_0 - p_0) << 2) + (p_1 - q_1) + 4) >> 3)) \quad (8\text{-}470)$$

$$p'_0 = \text{Clip1}(p_0 + \Delta) \quad (8\text{-}471)$$

$$q'_0 = \text{Clip1}(q_0 - \Delta) \quad (8\text{-}472)$$

où le seuil t_C est déterminé comme suit.

– Si chromaEdgeFlag est égal à 0,

$$t_C = t_{C0} + ((a_p < \beta) \ ? \ 1 : 0) + ((a_q < \beta) \ ? \ 1 : 0) \quad (8\text{-}473)$$

– Sinon (si chromaEdgeFlag est égal à 1),

$$t_C = t_{C0} + 1 \quad (8\text{-}474)$$

Selon les valeurs de indexA et de bS, la variable t'_{C0} est spécifiée dans le Tableau 8-17. Selon chromaEdgeFlag, la variable de seuil correspondante t_{C0} est calculée comme suit:

– Si chromaEdgeFlag est égal à 0,

$$t_{C0} = t'_{C0} * (1 << (\text{BitDepth}_Y - 8)) \quad (8\text{-}475)$$

– Sinon (si chromaEdgeFlag est égal à 1),

$$t_{C0} = t'_{C0} * (1 << (\text{BitDepth}_C - 8)) \quad (8\text{-}476)$$

Soit a_p et a_q les deux variables de seuil spécifiées par

$$a_p = Abs(p_2 – p_0) \quad (8\text{-}477)$$

$$a_q = Abs(q_2 – q_0) \quad (8\text{-}478)$$

L'échantillon filtré résultant p'_1 est calculé comme suit.

– Si chromaEdgeFlag est égal à 0 et que a_p est inférieur à β,

$$p'_1 = p_1 + Clip3(–t_{C0}, t_{C0}, (p_2 + ((p_0 + q_0 + 1) >> 1) – (p_1 << 1)) >> 1) \quad (8\text{-}479)$$

– Sinon (si chromaEdgeFlag est égal à 1 ou si a_p est supérieur ou égal à β),

$$p'_1 = p_1 \quad (8\text{-}480)$$

L'échantillon filtré résultant q'_1 est calculé comme suit.

– Si chromaEdgeFlag est égal à 0 et que a_q est inférieur à β,

$$q'_1 = q_1 + Clip3(–t_{C0}, t_{C0}, (q_2 + ((p_0 + q_0 + 1) >> 1) – (q_1 << 1)) >> 1) \quad (8\text{-}481)$$

– Sinon (si chromaEdgeFlag est égal à 1 ou si a_q est supérieur ou égal à β),

$$q'_1 = q_1 \quad (8\text{-}482)$$

Les échantillons filtrés résultants p'_2 et q'_2 sont toujours mis égaux aux échantillons d'entrée p_2 et q_2:

$$p'_2 = p_2 \quad (8\text{-}483)$$

$$q'_2 = q_2 \quad (8\text{-}484)$$

Tableau 8-17 – Valeur de la variable d'écrêtage de filtre t'_{C0} en fonction de indexA et bS

	indexA																									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
bS = 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
bS = 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
bS = 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Tableau 8-17 (fin) – Valeur de la variable d'écrêtage de filtre t'_{C0} en fonction de indexA et bS

	indexA																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
bS = 1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13
bS = 2	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5	5	6	7	8	8	10	11	12	13	15	17
bS = 3	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13	14	16	18	20	23	25

8.7.2.4 Processus de filtrage pour bordures avec bS égal à 4

Les entrées dans ce processus sont les valeurs d'échantillon d'entrée p_i et q_i (i = 0..3) d'un ensemble d'échantillons sur une bordure à filtrer, la variable chromaEdgeFlag, et les valeurs des variables de seuil α et β pour l'ensemble d'échantillons, comme spécifié au § 8.7.2.

Les résultats de ce processus sont les valeurs d'échantillon filtré résultant p'_i et q'_i (i = 0..2) pour l'ensemble de valeurs d'échantillon d'entrée.

Soit a_p et a_q deux variables de seuil comme spécifié respectivement par les équations 8-477 et 8-478 au § 8.7.2.3.

Les échantillons filtrés résultants p'_i (i = 0..2) sont calculés comme suit.

– Si chromaEdgeFlag est égal à 0 et que la condition suivante est satisfaite,

$$a_p < \beta \;\&\&\; Abs(p_0 - q_0) < ((\alpha >> 2) + 2) \tag{8-485}$$

les variables p'_0, p'_1, et p'_2 sont alors calculées par

$$p'_0 = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3 \tag{8-486}$$

$$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2 \tag{8-487}$$

$$p'_2 = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3 \tag{8-488}$$

– Sinon (si chromaEdgeFlag est égal à 1 ou si la condition de l'équation 8-485 n'est pas satisfaite), les variables p'_0, p'_1, et p'_2 sont calculées par:

$$p'_0 = (2*p_1 + p_0 + q_1 + 2) >> 2 \tag{8-489}$$

$$p'_1 = p_1 \tag{8-490}$$

$$p'_2 = p_2 \tag{8-491}$$

Les échantillons filtrés résultants q'_i (i = 0..2) sont déduits comme suit.

– Si chromaEdgeFlag est égal à 0 et que la condition suivante est satisfaite,

$$a_q < \beta \;\&\&\; Abs(p_0 - q_0) < ((\alpha >> 2) + 2) \tag{8-492}$$

les variables q'_0, q'_1, et q'_2 sont alors calculées par

$$q'_0 = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3 \tag{8-493}$$

$$q'_1 = (p_0 + q_0 + q_1 + q_2 + 2) >> 2 \tag{8-494}$$

$$q'_2 = (2*q_3 + 3*q_2 + q_1 + q_0 + p_0 + 4) >> 3 \tag{8-495}$$

– Sinon (si chromaEdgeFlag est égal à 1 ou si la condition de l'équation 8-492 n'est pas satisfaite), les variables q'_0, q'_1, et q'_2 sont calculées par:

$$q'_0 = (2*q_1 + q_0 + p_1 + 2) >> 2 \tag{8-496}$$

$$q'_1 = q_1 \tag{8-497}$$

$$q'_2 = q_2 \tag{8-498}$$

9 Processus d'analyse syntaxique

Les entrées dans ce processus sont des bits provenant de la charge utile RBSP.

Les résultats de ce processus sont des valeurs d'éléments syntaxiques.

Ce processus est invoqué lorsque le descripteur d'un élément syntaxique dans les tableaux syntaxiques du § 7.3 est égal à ue(v), me(v), se(v), te(v) (voir le § 9.1), ce(v) (voir le § 9.2), ou ae(v) (voir le § 9.3).

9.1 Processus d'analyse syntaxique pour les codes Exp-Golomb

Ce processus est invoqué lorsque le descripteur d'un élément syntaxique dans les tableaux syntaxiques du § 7.3 est égal à ue(v), me(v), se(v), ou te(v). Pour les éléments syntaxiques des § 7.3.4 et 7.3.5, ce processus n'est invoqué que lorsque entropy_coding_mode_flag est égal à 0.

Les entrées dans ce processus sont des bits provenant de la charge utile RBSP.

Les résultats de ce processus sont des valeurs d'éléments syntaxiques.

Les éléments syntaxiques codés ue(v), me(v), ou se(v) sont des Exp-Golomb codés. Les éléments syntaxiques codés te(v) sont des Exp-Golomb codés tronqués. Le processus d'analyse syntaxique pour ces éléments syntaxiques commence par la lecture des bits débutants à la localisation en cours dans le flux binaire jusqu'à et y compris le premier bit différent de zéro, et en comptant le nombre de bits directeurs qui sont égaux à 0. Ce processus est spécifié comme suit:

```
leadingZeroBits = -1;
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
```

La variable codeNum est alors allouée comme suit:

$$codeNum = 2^{leadingZeroBits} - 1 + read\_bits(leadingZeroBits)$$

où la valeur résultant de ces read_bits(leadingZeroBits) est interprétée comme une représentation binaire d'un entier algébrique avec le bit de plus fort poids écrit en premier.

Le Tableau 9-1 illustre la structure du code Exp-Golomb en séparant la chaîne binaire en bits de "préfixe" et de "suffixe". Les bits de "préfixe" sont les bits qui sont analysés syntaxiquement dans le pseudo-code ci-dessus pour le calcul des leadingZeroBits, et sont indiqués comme 0 ou 1 dans la colonne Forme de chaîne binaire du Tableau 9-1. Les bits de "suffixe" sont les bits qui sont analysés syntaxiquement dans le calcul de codeNum et sont indiqués comme x_i dans le Tableau 9-1, avec i dans la gamme de 0 à leadingZeroBits – 1, inclus. Chaque x_i peut prendre les valeurs 0 ou 1.

Tableau 9-1 – Chaînes binaires avec des bits de "préfixe" et de "suffixe" et allocation aux gammes de codeNum (pour information)

Forme de chaîne binaire	Gamme de codeNum
1	0
0 1 x_0	1-2
0 0 1 x_1 x_0	3-6
0 0 0 1 x_2 x_1 x_0	7-14
0 0 0 0 1 x_3 x_2 x_1 x_0	15-30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31-62
…	…

Le Tableau 9-2 illustre explicitement l'allocation des chaînes binaires aux valeurs de codeNum.

Tableau 9-2 – Chaînes binaires Exp-Golomb et codeNum en forme explicite et utilisées comme ue(v) (pour information)

Chaîne binaire	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	…

En fonction du descripteur, la valeur d'un élément syntaxique est déduite comme suit.

– Si l'élément syntaxique est codé comme ue(v), la valeur de l'élément syntaxique est égale à codeNum.

– Sinon, si l'élément syntaxique est codé comme se(v), la valeur de l'élément syntaxique est déduite en invoquant le processus de mappage pour les codes Exp-Golomb algébriques comme spécifié au § 9.1.1 avec codeNum comme entrée.

– Sinon, si l'élément syntaxique est codé comme me(v), la valeur de l'élément syntaxique est déduite en invoquant le processus de mappage pour schéma de bloc codé comme spécifié au § 9.1.2 avec codeNum comme entrée.

– Sinon (si l'élément syntaxique est codé comme te(v)), la gamme des valeurs possibles de l'élément syntaxique est d'abord déterminée. La gamme de cet élément syntaxique peut être entre 0 et x, avec x supérieur ou égal à 1 et cette gamme est utilisée dans le calcul de la valeur de l'élément syntaxique comme suit:

  – si x est supérieur à 1, les valeurs de codeNum et de l'élément syntaxique sont calculées de la même façon que pour les éléments syntaxiques codés ue(v);

  – sinon (si x est égal à 1), le processus d'analyse syntaxique pour codeNum qui est égal à la valeur de l'élément syntaxique est donné par un processus équivalent à:

    b = read_bits(1)
    codeNum = !b

9.1.1 Processus de mappage pour les codes Exp-Golomb algébriques

L'entrée dans ce processus est codeNum comme spécifié au § 9.1.

Le résultat de ce processus est une valeur d'un élément syntaxique codé se(v).

L'élément syntaxique est alloué à la variable codeNum par ordonnancement de l'élément syntaxique selon sa valeur absolue en ordre croissant et en représentant la valeur positive d'une certaine valeur absolue par la plus basse valeur de codeNum. Le Tableau 9-3 donne la règle d'allocation.

Tableau 9-3 – Allocation d'élément syntaxique à codeNum pour les éléments syntaxiques codés en Exp-Golomb algébrique se(v)

codeNum	Valeur d'élément syntaxique
0	0
1	1
2	–1
3	2
4	–2
5	3
6	–3
k	$(-1)^{k+1}$ Ceil(k÷2)

9.1.2 Processus de mappage pour schéma de bloc codé

L'entrée dans ce processus est codeNum comme spécifié au § 9.1.

Le résultat de ce processus est une valeur de l'élément syntaxique coded_block_pattern codé me(v).

Le Tableau 9-4 montre l'allocation de coded_block_pattern à codeNum selon que le mode de prédiction de macrobloc est égal à Intra_4x4, Intra_8x8 ou Inter.

Tableau 9-4 – Allocation de codeNum aux valeurs de coded_block_pattern pour les modes de prédiction de macrobloc

(a) chroma_format_idc n'est pas égal à 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
0	47	0
1	31	16
2	15	1
3	0	2
4	23	4
5	27	8
6	29	32
7	30	3
8	7	5
9	11	10
10	13	12
11	14	15
12	39	47
13	43	7
14	45	11
15	46	13
16	16	14
17	3	6
18	5	9
19	10	31
20	12	35
21	19	37
22	21	42
23	26	44
24	28	33
25	35	34

Tableau 9-4 – Allocation de codeNum aux valeurs de coded_block_pattern pour les modes de prédiction de macrobloc

(a) chroma_format_idc n'est pas égal à 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
26	37	36
27	42	40
28	44	39
29	1	43
30	2	45
31	4	46
32	8	17
33	17	18
34	18	20
35	20	24
36	24	19
37	6	21
38	9	26
39	22	28
40	25	23
41	32	27
42	33	29
43	34	30
44	36	22
45	40	25
46	38	38
47	41	41

(b) chroma_format_idc est égal à 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
0	15	0
1	0	1
2	7	2
3	11	4
4	13	8
5	14	3
6	3	5
7	5	10
8	10	12
9	12	15
10	1	7
11	2	11
12	4	13
13	8	14
14	6	6
15	9	9

9.2 Processus d'analyse syntaxique CAVLC pour les niveaux de coefficient de transformée

Ce processus est invoqué lorsque les éléments syntaxiques d'analyse syntaxique avec descripteur sont égaux à ce(v) du § 7.3.5.3.1 et lorsque entropy_coding_mode_flag est égal à 0.

Les entrées dans ce processus sont des bits provenant des données de tranche, un nombre maximal des niveaux de coefficient de transformée différents de zéro maxNumCoeff, l'indice de bloc luma luma4x4BlkIdx ou l'indice de bloc chroma chroma4x4BlkIdx du bloc en cours de niveaux de coefficient de transformée.

Le résultat de ce processus est la liste coeffLevel contenant les niveaux de coefficient de transformée du bloc luma avec l'indice de bloc luma4x4BlkIdx ou du bloc chroma avec l'indice de bloc chroma4x4BlkIdx.

Le processus est spécifié selon les étapes ordonnées suivantes:

1. Tous les niveaux de coefficient de transformée, avec les indices de 0 à maxNumCoeff – 1, dans la liste coeffLevel sont mis égaux à 0.
2. Le nombre total des niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token) et le nombre de niveaux de coefficient de transformée de traîne TrailingOnes(coeff_token) sont calculés par l'analyse syntaxique de coeff_token (voir le § 9.2.1) comme suit.
   - Si le nombre des niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token) est égal à 0, la liste coeffLevel contenant des valeurs 0 est retournée et aucune autre étape n'est effectuée;
   - Sinon, les étapes suivantes sont effectuées.
     a. Les niveaux de coefficient de transformée différents de zéro sont calculés en analysant syntaxiquement trailing_ones_sign_flag, level_prefix, et level_suffix (voir le § 9.2.2).
     b. Les exécutions de niveaux de coefficient de transformée zéro avant chaque de niveau de coefficient de transformée différent de zéro sont calculées en analysant syntaxiquement total_zeros et run_before (voir le § 9.2.3).

c. Les informations de niveau et d'occurrence sont combinées dans la liste coeffLevel (voir le § 9.2.4).

9.2.1 Processus d'analyse syntaxique pour le nombre total de niveaux de coefficient de transformée et de traînes

Les entrées dans ce processus sont des bits provenant des données de tranche, un nombre maximal de niveaux de coefficient de transformée différents de zéro maxNumCoeff, l'indice de bloc luma luma4x4BlkIdx ou l'indice de bloc chroma chroma4x4BlkIdx du bloc de transformées en cours.

Les résultats de ce processus sont TotalCoeff(coeff_token) et TrailingOnes(coeff_token).

L'élément syntaxique coeff_token est décodé au moyen d'un des six codages VLC spécifiés dans les six colonnes les plus à droite du Tableau 9-5. Chaque codage VLC spécifie à la fois TotalCoeff(coeff_token) et TrailingOnes(coeff_token) pour un mot de code coeff_token donné. Le choix du codage VLC dépend d'une variable nC qui est déduite comme suit.

– si le processus d'analyse syntaxique CAVLC est invoqué pour ChromaDCLevel, nC est calculé comme suit.
  – Si chroma_format_idc est égal à 1, nC est mis à -1,
  – Sinon, si chroma_format_idc est égal à 2, nC est mis à -2,
  – Sinon (si chroma_format_idc est égal à 3), nC est mis à 0.
– sinon, on applique ce qui suit.
  – lorsque le processus d'analyse syntaxique CAVLC est invoqué pour Intra16x16DCLevel, luma4x4BlkIdx est mise égal à 0;
  – les variables blkA et blkB sont déduites comme suit.
    – Si le processus d'analyse syntaxique CAVLC est invoqué pour Intra16x16DCLevel, Intra16x16ACLevel, ou LumaLevel, le processus spécifié au § 6.4.8.3 est invoqué avec luma4x4BlkIdx comme entrée, et le résultat est alloué à mbAddrA, mbAddrB, luma4x4BlkIdxA, et luma4x4BlkIdxB. Le bloc luma 4x4 spécifié par mbAddrA\luma4x4BlkIdxA est alloué à blkA, et le bloc luma 4x4 spécifié par mbAddrB\luma4x4BlkIdxB est alloué à blkB.
    – Sinon (si le processus d'analyse syntaxique CAVLC est invoqué pour ChromaACLevel), le processus spécifié au § 6.4.8.4 est invoqué avec chroma4x4BlkIdx en entrée, et le résultat est alloué à mbAddrA, mbAddrB, chroma4x4BlkIdxA, et chroma4x4BlkIdxB. Le bloc chroma 4x4 spécifié par mbAddrA\iCbCr\chroma4x4BlkIdxA est alloué à blkA, et le bloc chroma 4x4 spécifié par mbAddrB\iCbCr\chroma4x4BlkIdxB est alloué à blkB.
  – Soit nA et nB le nombre des niveaux de coefficient de transformée différents de zéro (donné par TotalCoeff(coeff_token)) respectivement dans le bloc de niveaux de coefficient de transformée blkA localisé à gauche du bloc en cours et dans le bloc de niveaux de coefficient de transformée blkB localisé au-dessus du bloc en cours.
  – Avec N remplacé par A et B, dans mbAddrN, blkN, et nN, on applique ce qui suit.
    – Si l'une des conditions suivantes est vraie, nN est mis égal à 0.
      – mbAddrN n'est pas disponible.
      – Le macrobloc en cours est codé au moyen d'un mode d'intraprédiction, constrained_intra_pred_flag est égal à 1, mbAddrN est codé au moyen de l'interprédiction et la subdivision de données de tranche est utilisée (nal_unit_type est dans la gamme de 2 à 4, inclus).
      – Le macrobloc mbAddrN a mb_type égal à P_Skip ou B_Skip.
      – Tous les niveaux de coefficient de transformée AC résiduel du bloc voisin blkN sont égaux à 0 du fait du bit correspondant de CodedBlockPatternLuma ou CodedBlockPatternChroma qui est égal à 0.
    – Sinon, si mbAddrN est un macrobloc I_PCM, nN est mis égal à 16.
    – Sinon, nN est mis égal à la valeur TotalCoeff(coeff_token) du bloc voisin blkN.

NOTE 1 – Les valeurs nA et nB qui sont calculées au moyen de TotalCoeff(coeff_token) n'incluent pas les niveaux de coefficient de transformée DC dans les macroblocs Intra 16x16 ou les niveaux de coefficient de transformée DC dans les blocs chroma, parce que ces niveaux de coefficient de transformée sont décodés séparément. Lorsque le bloc au-dessus ou à gauche appartient à un macrobloc Intra 16x16, ou est un bloc chroma, nA et nB est le nombre de niveaux de coefficient de transformée AC décodés différents de zéro.

NOTE 2 – Lors de l'analyse syntaxique de Intra16x16DCLevel, les valeurs nA et nB sont fondées sur le nombre des niveaux de coefficient de transformée différents de zéro dans les blocs 4x4 adjacents et non sur le nombre de niveaux de coefficient de transformée DC différents de zéro dans les blocs 16x16 adjacents.

– Etant donné les valeurs de nA et nB, la variable nC est déduite comme suit.
  – Si mbAddrA et mbAddrB sont tous deux disponibles, la variable nC est mis égal à (nA + nB + 1) >> 1.
  – Sinon (si mbAddrA n'est pas disponible ou mbAddrB n'est pas disponible), la variable nC est mis égal à nA + nB.

La valeur de TotalCoeff(coeff_token) résultant du décodage de coeff_token doit être dans la gamme de 0 à maxNumCoeff, inclus.

Tableau 9-5 – Mappage de coeff_token sur TotalCoeff(coeff_token) et TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = -1	nC = = -2
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010	0001 010
3	4	0000 11	0100	1011	0011 11	0000 000	0000 01
0	5	0000 0000 111	0000 0100	0001 011	0100 00	-	0000 0001 11
1	5	0000 0001 10	0000 110	0100 0	0100 01	-	0000 0001 10
2	5	0000 0010 1	0000 101	0100 1	0100 10	-	0000 0010 0
3	5	0000 100	0011 0	1010	0100 11	-	0001 001
0	6	0000 0000 0111 1	0000 0011 1	0001 001	0101 00	-	0000 0000 111
1	6	0000 0000 110	0000 0110	0011 10	0101 01	-	0000 0000 110
2	6	0000 0001 01	0000 0101	0011 01	0101 10	-	0000 0001 01
3	6	0000 0100	0010 00	1001	0101 11	-	0001 000
0	7	0000 0000 0101 1	0000 0001 111	0001 000	0110 00	-	0000 0000 0111
1	7	0000 0000 0111 0	0000 0011 0	0010 10	0110 01	-	0000 0000 0110
2	7	0000 0000 101	0000 0010 1	0010 01	0110 10	-	0000 0000 101
3	7	0000 0010 0	0001 00	1000	0110 11	-	0000 0001 00

Tableau 9-5 – Mappage de coeff_token sur TotalCoeff(coeff_token) et TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = -1	nC = = -2
0	8	0000 0000 0100 0	0000 0001 011	0000 1111	0111 00	-	0000 0000 0011 1
1	8	0000 0000 0101 0	0000 0001 110	0001 110	0111 01	-	0000 0000 0101
2	8	0000 0000 0110 1	0000 0001 101	0001 101	0111 10	-	0000 0000 0100
3	8	0000 0001 00	0000 100	0110 1	0111 11	-	0000 0000 100
0	9	0000 0000 0011 11	0000 0000 1111	0000 1011	1000 00	-	
1	9	0000 0000 0011 10	0000 0001 010	0000 1110	1000 01	-	
2	9	0000 0000 0100 1	0000 0001 001	0001 010	1000 10	-	
3	9	0000 0000 100	0000 0010 0	0011 00	1000 11	-	
0	10	0000 0000 0010 11	0000 0000 1011	0000 0111 1	1001 00	-	
1	10	0000 0000 0010 10	0000 0000 1110	0000 1010	1001 01	-	
2	10	0000 0000 0011 01	0000 0000 1101	0000 1101	1001 10	-	
3	10	0000 0000 0110 0	0000 0001 100	0001 100	1001 11	-	
0	11	0000 0000 0001 111	0000 0000 1000	0000 0101 1	1010 00	-	
1	11	0000 0000 0001 110	0000 0000 1010	0000 0111 0	1010 01	-	
2	11	0000 0000 0010 01	0000 0000 1001	0000 1001	1010 10	-	
3	11	0000 0000 0011 00	0000 0001 000	0000 1100	1010 11	-	
0	12	0000 0000 0001 011	0000 0000 0111 1	0000 0100 0	1011 00	-	
1	12	0000 0000 0001 010	0000 0000 0111 0	0000 0101 0	1011 01	-	
2	12	0000 0000 0001 101	0000 0000 0110 1	0000 0110 1	1011 10	-	
3	12	0000 0000 0010 00	0000 0000 1100	0000 1000	1011 11	-	
0	13	0000 0000 0000 1111	0000 0000 0101 1	0000 0011 01	1100 00	-	
1	13	0000 0000 0000 001	0000 0000 0101 0	0000 0011 1	1100 01	-	
2	13	0000 0000 0001 001	0000 0000 0100 1	0000 0100 1	1100 10	-	
3	13	0000 0000 0001 100	0000 0000 0110 0	0000 0110 0	1100 11	-	
0	14	0000 0000 0000 1011	0000 0000 0011 1	0000 0010 01	1101 00	-	
1	14	0000 0000 0000 1110	0000 0000 0010 11	0000 0011 00	1101 01	-	
2	14	0000 0000 0000 1101	0000 0000 0011 0	0000 0010 11	1101 10	-	
3	14	0000 0000 0001 000	0000 0000 0100 0	0000 0010 10	1101 11	-	
0	15	0000 0000 0000 0111	0000 0000 0010 01	0000 0001 01	1110 00	-	
1	15	0000 0000 0000 1010	0000 0000 0010 00	0000 0010 00	1110 01	-	
2	15	0000 0000 0000 1001	0000 0000 0010 10	0000 0001 11	1110 10	-	
3	15	0000 0000 0000 1100	0000 0000 0000 1	0000 0001 10	1110 11	-	
0	16	0000 0000 0000 0100	0000 0000 0001 11	0000 0000 01	1111 00	-	
1	16	0000 0000 0000 0110	0000 0000 0001 10	0000 0001 00	1111 01	-	
2	16	0000 0000 0000 0101	0000 0000 0001 01	0000 0000 11	1111 10	-	
3	16	0000 0000 0000 1000	0000 0000 0001 00	0000 0000 10	1111 11	-	

9.2.2 Processus d'analyse syntaxique pour les informations de niveau

Les entrées dans ce processus sont les bits provenant des données de tranche, le nombre des niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token), et le nombre de niveaux de coefficient de transformée de traîne TrailingOnes(coeff_token).

Le résultat de ce processus est une liste avec le niveau de nom qui contient les niveaux de coefficient de transformée.

Un indice i est mis égal à 0 au départ. Puis la procédure itérative suivante est appliquée TrailingOnes(coeff_token) fois afin de décoder les niveaux de coefficient de transformée de traîne (le cas échéant):

– un élément syntaxique à un bit trailing_ones_sign_flag est décodé et évalué comme suit.
  – Si trailing_ones_sign_flag est égal à 0, la valeur +1 est allouée au niveau [i].
  – Sinon (si trailing_ones_sign_flag est égal à 1), la valeur –1 est allouée au niveau [i].
– l'indice i est incrémenté de 1.

A la suite du décodage des niveaux de coefficient de transformée de traîne, on initialise comme suit une variable suffixLength.

– Si TotalCoeff(coeff_token) est supérieur à 10 et que TrailingOnes(coeff_token) est inférieur à 3, suffixLength est mis égal à 1.
– Sinon (si TotalCoeff(coeff_token) est inférieur ou égal à 10 ou TrailingOnes(coeff_token) est égal à 3), suffixLength est mis égal à 0.

On applique alors la procédure itérative suivante (TotalCoeff(coeff_token) – TrailingOnes(coeff_token)) fois afin de décoder les niveaux restants (le cas échéant):

– l'élément syntaxique level_prefix est décodé comme spécifié dans le § 9.2.2.1;
– la variable levelSuffixSize est mise égale à la variable suffixLength à l'exception des deux cas suivants;
– lorsque level_prefix est égal à 14 et que suffixLength est égal à 0, levelSuffixSize est mis égal à 4;
– lorsque level_prefix est supérieur ou égal à 15, levelSuffixSize est mis égal à level_prefix – 3;
– l'élément syntaxique level_suffix est décodé comme suit.
  – Si levelSuffixSize est supérieur à 0, l'élément syntaxique level_suffix est décodé comme représentation d'un entier algébrique u(v) avec levelSuffixSize bits.
  – Sinon (si levelSuffixSize est égal à 0), l'élément syntaxique level_suffix est supposé égal à 0.
– Une variable levelCode est mise égale à (Min(15, level_prefix << suffixLength) + level_suffix.
– Lorsque level_prefix est supérieur ou égal à 15 et que suffixLength est égal à 0, levelCode est incrémentée de 15.
– Lorsque level_prefix est plus grand ou égal à 16, levelCode est incrémenté de (1<<(level_prefix – 3)) – 4096.
– Lorsque l'indice i est égal à TrailingOnes(coeff_token) et que TrailingOnes(coeff_token) estinférieur à 3, levelCode est incrémenté de 2.
– La variable level[i] est déduite comme suit.
  – Si levelCode est un nombre pair, la valeur (levelCode + 2) >> 1 est allouée au niveau [i].
  – Sinon (si levelCode est un nombre impair), la valeur (-levelCode – 1) >> 1 est allouée au niveau [i].
– Lorsque suffixLength est égal à 0, suffixLength est mis égal à 1.
– Lorsque la valeur absolue de niveau [i] est supérieure à (3 << (suffixLength – 1)) et que suffixLength est inférieure à 6, suffixLength est incrémentée de 1.
– L'indice i est incrémenté de 1.

9.2.2.1 Processus d'analyse syntaxique pour level_prefix

Les entrées dans ce processus sont des bits de données de tranche.

La sortie de ce processus est level_prefix.

Le processus d'analyse syntaxique pour cet élément syntaxique consiste en une lecture des bits à partir de l'emplacement actuel dans le flux binaire jusques et y compris le premier bit différent de zéro, et en comptant le nombre de bits initiaux qui sont égaux à 0. Ce processus est spécifié comme suit:

```
leadingZeroBits = -1
pour( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
level_prefix = leadingZeroBits
```

Le Tableau 9-6 décrit le tableau des mots de code pour level_prefix.

Tableau 9-6 – Tableau des mots de code pour level_prefix (pour information)

level_prefix	Chaîne binaire
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
8	0000 0000 1
9	0000 0000 01
10	0000 0000 001
11	0000 0000 0001
12	0000 0000 0000 1
13	0000 0000 0000 01
14	0000 0000 0000 001
15	0000 0000 0000 0001
…	…

9.2.3 Processus d'analyse syntaxique pour information d'exécution

Les entrées dans ce processus sont des bits provenant de données de tranche, le nombre de niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token), et le nombre maximal de niveaux de coefficient de transformée différents de zéro maxNumCoeff.

Le résultat de ce processus est une liste des exécutions des niveaux de coefficient de transformée zéro précédant des niveaux de coefficient de transformée différents de zéro appelés exécutions (*run*).

Au départ, un indice i est mis égal à 0.

La variable zerosLeft est déduite comme suit.

– Si le nombre de niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token) est égal au nombre maximal de niveaux de coefficient de transformée différents de zéro maxNumCoeff, une variable zerosLeft est mise égale à 0.

– Sinon (si le nombre de niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token) est inférieur au nombre maximal de niveaux de coefficient de transformée différents de zéro maxNumCoeff), total_zeros est décodé et zerosLeft est mis égal à sa valeur.

Le code VLC utilisé afin de décoder total_zeros est déduit comme suit:

– si maxNumCoeff est égal à 4, un seul des codes VLC spécifié au Tableau 9-9 (a) est utilisé;

– sinon, si maxNumCoeff est égal à 8, un seul des codes VLC spécifiés dans le Tableau 9-9 (b) est utilisé.

– sinon (si maxNumCoeff n'est pas égal à 4 et n'est pas égal à 8), les codes VLC tirés des Tableaux 9-7 et 9-8 sont utilisés.

On applique alors la procédure itérative suivante (TotalCoeff(coeff_token) – 1) fois:

– La variable run[i] est déduite comme suit.

  – Si zerosLeft est supérieur à zero, une valeur run_before est décodée sur la base du Tableau 9-10 et zerosLeft. run[i] est mis égal à run_before.

  – Sinon (si zerosLeft est égal à 0), run[i] est mis égal à 0.

– La valeur de run[i] est soustraite de zerosLeft et le résultat est alloué à zerosLeft. Le résultat de la soustraction doit être supérieur ou égal à 0.

– L'indice i est incrémenté de 1.

Finalement, la valeur de zerosLeft est allouée à run[i].

Tableau 9-7 – Tableaux de total_zeros pour blocs 4x4 avec TotalCoeff(coeff_token) de 1 à 7

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	111	0101	0001 1	0101	0000 01	0000 01
1	011	110	111	111	0100	0000 1	0000 1
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	0001 1	0101	0011	101	101	100	11
6	0001 0	0100	100	100	100	011	010
7	0000 11	0011	011	0011	011	010	0001
8	0000 10	0010	0010	011	0010	0001	001
9	0000 011	0001 1	0001 1	0010	0000 1	001	0000 00
10	0000 010	0001 0	0001 0	0001 0	0001	0000 00	
11	0000 0011	0000 11	0000 01	0000 1	0000 0		
12	0000 0010	0000 10	0000 1	0000 0			
13	0000 0001 1	0000 01	0000 00				
14	0000 0001 0	0000 00					
15	0000 0000 1						

Tableau 9-8 – Tableaux de total_zeros pour blocs 4x4 avec TotalCoeff(coeff_token) de 8 à 15

total_zeros	TotalCoeff(coeff_token)							
	8	9	10	11	12	13	14	15
0	0000 01	0000 01	0000 1	0000	0000	000	00	0
1	0001	0000 00	0000 0	0001	0001	001	01	1
2	0000 1	0001	001	001	01	1	1	
3	011	11	11	010	1	01		
4	11	10	10	1	001			
5	10	001	01	011				
6	010	01	0001					
7	001	0000 1						
8	0000 00							

Tableau 9-9 – Tableaux de total_zeros pour blocs chroma DC 2x2 et 2x4

(a) Bloc 2x2 chroma DC (échantillonnage chroma 4:2:0)

total_zeros	TotalCoeff(coeff_token)		
	1	2	3
0	1	1	1
1	01	01	0
2	001	00	
3	000		

(b) Bloc 2x4 chroma DC (échantillonnage chroma 4:2:2)

total_zeros	TotalCoeff(coeff_token)						
	1	**2**	**3**	**4**	**5**	**6**	**7**
0	1	000	000	110	00	00	0
1	010	01	001	00	01	01	1
2	011	001	01	01	10	1	
3	0010	100	10	10	11		
4	0011	101	110	111			
5	0001	110	111				
6	0000 1	111					
7	0000 0						

Tableau 9-10 – Tableaux pour run_before

run_before	zerosLeft						
	1	**2**	**3**	**4**	**5**	**6**	**>6**
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8		-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

9.2.4 Combinaison des informations de niveau et d'exécution

Les entrées dans ce processus sont une liste de niveaux de coefficient de transformée appelés niveau (*level*), une liste d'exécutions appelées exécution (*run*) et le nombre de niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token).

Le résultat de ce processus est une liste coeffLevel de niveaux de coefficient de transformée.

Une variable coeffNum est mise égale à –1 et un indice i est mis égal à (TotalCoeff(coeff_token) – 1). On applique la procédure itérative suivante TotalCoeff(coeff_token) fois:

– coeffNum est incrémenté de run[i] + 1;

– coeffLevel[coeffNum] est mis égal à level[i];

– l'indice i est décrémenté de 1.

9.3 Processus d'analyse syntaxique CABAC pour données de tranche

Ce processus est invoqué lors de l'analyse syntaxique d'éléments syntaxiques avec le descripteur ae(v) des § 7.3.4 et 7.3.5 lorsque entropy_coding_mode_flag est égal à 1.

Les entrées dans ce processus sont une demande de valeur d'élément syntaxique et les valeurs des éléments syntaxiques précédemment analysés syntaxiquement.

Le résultat de ce processus est la valeur de l'élément syntaxique.

Lors du démarrage de l'analyse syntaxique des données de tranche d'une tranche du § 7.3.4, le processus d'initialisation du processus d'analyse syntaxique CABAC est invoqué comme spécifié au § 9.3.1.

L'analyse syntaxique des éléments syntaxiques s'effectue comme suit:

pour chaque valeur demandée d'un élément syntaxique est calculée une binarisation, comme décrit au § 9.3.2.

La binarisation pour l'élément syntaxique et la séquence de segments (*bins*) analysés syntaxiquement détermine le flux du processus de décodage, comme décrit au § 9.3.3.

Pour chaque segment de la binarisation de l'élément syntaxique, qui est indexé par la variable binIdx, un indice de contexte ctxIdx est calculé comme spécifié au § 9.3.3.1.

Pour chaque ctxIdx le processus de décodage arithmétique est invoqué comme spécifié au § 9.3.3.2.

La séquence (b_0 .. b_{binIdx}) résultante de bins analysés syntaxiquement est comparée à l'ensemble de la chaîne de segments donnée par le processus de binarisation après le décodage de chaque segment. Lorsque la séquence correspond à une chaîne de segments dans l'ensemble donné, la valeur correspondante est allouée à l'élément syntaxique.

Dans le cas où la demande de valeur d'un élément syntaxique est effectuée pour l'élément syntaxique mb_type et où la valeur décodée de mb_type est égal à I_PCM, le moteur de décodage est initialisé après le décodage de tout élément binaire pcm_alignment_zero_bit et de toutes données pcm_sample_luma et pcm_sample_chroma, comme spécifié au § 9.3.1.2.

L'ensemble du processus d'analyse syntaxique CABAC est illustré dans l'organigramme de la Figure 9-1 avec l'abréviation SE pour élément syntaxique (SE, *syntax element*).

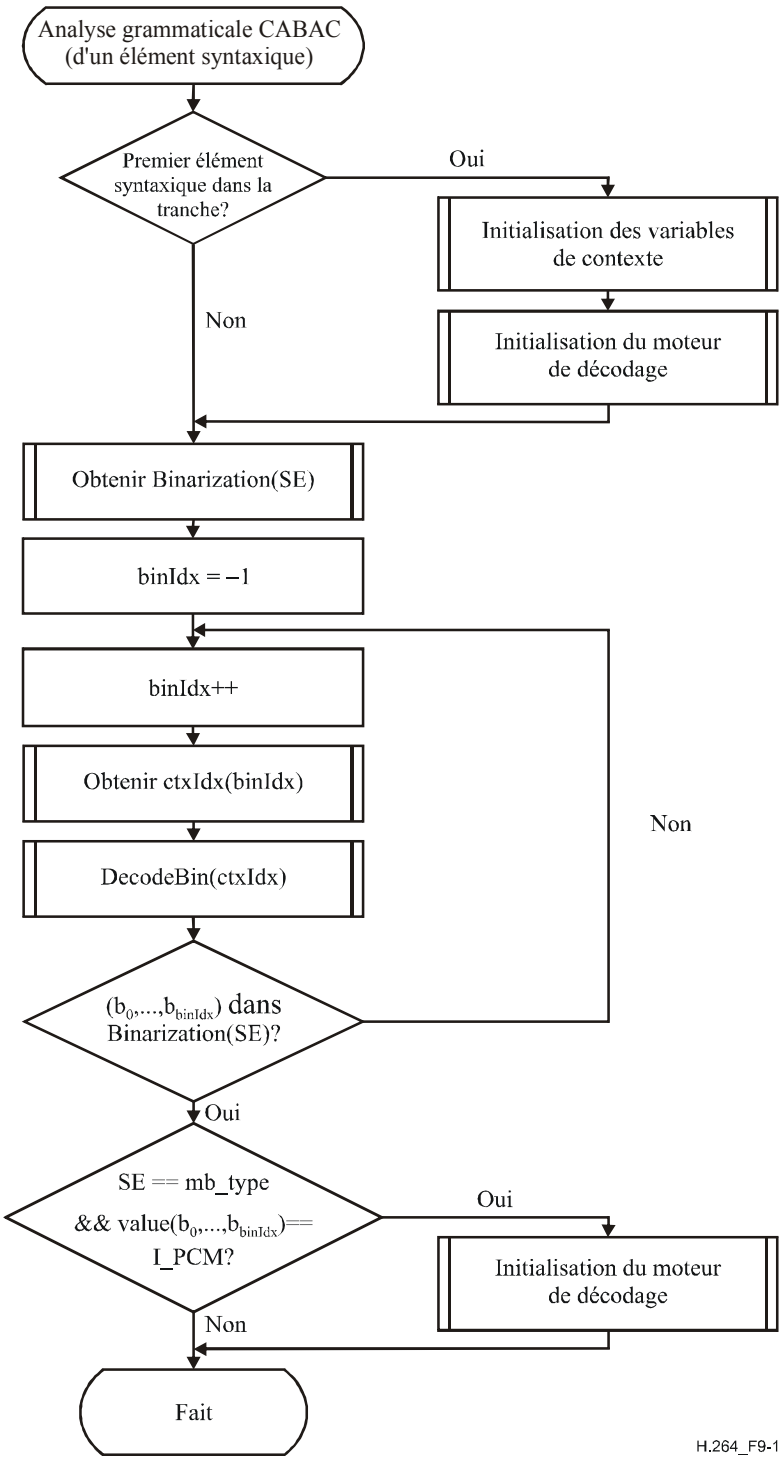


Figure 9-1 – Illustration du processus d'analyse syntaxique CABAC pour un élément syntaxique SE (pour information)

9.3.1 Processus d'initialisation

Les résultats de ce processus sont des variables internes CABAC initialisées.

Les processus des § 9.3.1.1 et 9.3.1.2 sont invoqués lors du début de l'analyse syntaxique des données de tranche d'une tranche du § 7.3.4.

Le processus du § 9.3.1.2 est aussi invoqué après le décodage de tout élément pcm_alignment_zero_bit et de toutes les donnéespcm_sample_luma et pcm_sample_chroma pour un macrobloc de type I_PCM.

9.3.1.1 Processus d'initialisation pour variables de contexte

Les résultats de ce processus sont les variables de contexte CABAC initialisées indexées par ctxIdx.

Du Tableau 9-12 au Tableau 9-23 figurent les valeurs des variables n et m utilisées dans l'initialisation des variables de contexte qui sont allouées à tous les éléments syntaxiques des § 7.3.4 et 7.3.5 excepté pour le fanion de fin de tranche.

Pour chaque variable de contexte, on initialise les deux variables pStateIdx et valMPS.

NOTE 1 – La variable pStateIdx correspond à un indice de probabilité d'état et la variable valMPS correspond à la valeur du symbole le plus probable, comme décrit plus loin au § 9.3.3.2.

Les deux valeurs allouées à pStateIdx et valMPS pour l'initialisation sont dérivées de $SliceQP_Y$, qui est calculé dans l'équation 7-27. Soit les deux entrées de tableau (m, n):

1. preCtxState = Clip3(1, 126, ((m * Clip3(0, 51, $SliceQP_Y$)) >> 4) + n)
2. if(preCtxState <= 63) {

        pStateIdx = 63 - preCtxState

        valMPS = 0

    } else {

        pStateIdx = preCtxState - 64

        valMPS = 1

    }

Dans le Tableau 9-11 figure la liste des ctxIdx pour lesquels l'initialisation est nécessaire pour chacun des types de tranche. Sont aussi énumérés les numéros de tableau qui incluent les valeurs de m et n nécessaires pour l'initialisation. Pour les types de tranche P, SP et B, l'initialisation dépend aussi de la valeur de l'élément syntaxique cabac_init_idc. Noter que les noms d'élément syntaxique n'affectent pas le processus d'initialisation.

Tableau 9-11 – Association des ctxIdx et des éléments syntaxiques pour chaque type de tranche dans le processus d'initialisation

	Elément syntaxique	Tableau	Type de tranche			
			SI	I	P, SP	B
slice_data( )	mb_skip_flag	Tableau 9-13 Tableau 9-14			11-13	24-26
	mb_field_decoding_flag	Tableau 9-18	70-72	70-72	70-72	70-72
macroblock_layer( )	mb_type	Tableau 9-12 Tableau 9-13 Tableau 9-14	0-10	3-10	14-20	27-35
	transform_size_8x8_flag	Tableau 9-16	na	399-401	399-401	399-401
	coded_block_pattern (luma)	Tableau 9-18	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	Tableau 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	Tableau 9-17	60-63	60-63	60-63	60-63
mb_pred( )	prev_intra4x4_pred_mode_flag	Tableau 9-17	68	68	68	68
	rem_intra4x4_pred_mode	Tableau 9-17	69	69	69	69
	prev_intra8x8_pred_mode_flag	Tableau 9-17	na	68	68	68
	rem_intra8x8_pred_mode	Tableau 9-17	na	69	69	69
	intra_chroma_pred_mode	Tableau 9-17	64-67	64-67	64-67	64-67
mb_pred( ) and sub_mb_pred( )	ref_idx_l0	Tableau 9-16			54-59	54-59
	ref_idx_l1	Tableau 9-16				54-59
	mvd_l0[ ][ ][0]	Tableau 9-15			40-46	40-46
	mvd_l1[ ][ ][0]	Tableau 9-15				40-46
	mvd_l0[ ][ ][1]	Tableau 9-15			47-53	47-53
	mvd_l1[ ][ ][1]	Tableau 9-15				47-53
sub_mb_pred( )	sub_mb_type	Tableau 9-13 Tableau 9-14			21-23	36-39
residual_block_cabac( )	coded_block_flag	Tableau 9-18	85-104	85-104	85-104	85-104
	significant_coeff_flag[ ]	Tableau 9-19 Tableau 9-22 Tableau 9-24 Tableau 9-24	105-165 277-337	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450
	last_significant_coeff_flag[ ]	Tableau 9-20 Tableau 9-23 Tableau 9-24 Tableau 9-24	166-226 338-398	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459
	coeff_abs_level_minus1[ ]	Tableau 9-21 Tableau 9-24	227-275	227-275 426-435	227-275 426-435	227-275 426-435

NOTE 2 – ctxIdx égal à 276 est associé au fanion end_of_slice_flag et au segment de mb_type, qui spécifie le type de macrobloc I_PCM. Le processus de décodage spécifié au § 9.3.3.2.4 s'applique à ctxIdx égal à 276. Ce processus de décodage peut cependant être aussi mis en œuvre au moyen du processus de décodage spécifié au § 9.3.3.2.1. Dans ce cas, les valeurs initiales associées à ctxIdx égal à 276 sont spécifiées comme pStateIdx = 63 et valMPS = 0, où pStateIdx = 63 représente un état de probabilité de non-adaptation.

Tableau 9-12 – Valeurs des variables m et n pour ctxIdx de 0 à 10

Variables d'initiatisation	ctxIdx										
	0	1	2	3	4	5	6	7	8	9	10
m	20	2	3	20	2	3	-28	-23	-6	-1	7
n	-15	54	74	-15	54	74	127	104	53	54	51

Tableau 9-13 – Valeurs des variables m et n pour ctxIdx de 11 à 23

Valeur de cabac_init_idc	Variables d'initialisation	ctxIdx												
		11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
0	**m**	23	23	21	1	0	-37	5	-13	-11	1	12	-4	17
	n	33	2	0	9	49	118	57	78	65	62	49	73	50
1	**m**	22	34	16	-2	4	-29	2	-6	-13	5	9	-3	10
	n	25	0	0	9	41	118	65	71	79	52	50	70	54
2	**m**	29	25	14	-10	-3	-27	26	-4	-24	5	6	-17	14
	n	16	0	0	51	62	99	16	85	102	57	57	73	57

Tableau 9-14 – Valeurs des variables m et n pour ctxIdx de 24 à 39

Valeur de cabac_init_idc	Variables d'initialisation	ctxIdx															
		24	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**
0	**m**	18	9	29	26	16	9	-46	-20	1	-13	-11	1	-6	-17	-6	9
	n	64	43	0	67	90	104	127	104	67	78	65	62	86	95	61	45
1	**m**	26	19	40	57	41	26	-45	-15	-4	-6	-13	5	6	-13	0	8
	n	34	22	0	2	36	69	127	101	76	71	79	52	69	90	52	43
2	**m**	20	20	29	54	37	12	-32	-22	-2	-4	-24	5	-6	-14	-6	4
	n	40	10	0	0	42	97	127	117	74	85	102	57	93	88	44	55

Tableau 9-15 – Valeurs des variables m et n pour ctxIdx de 40 à 53

Valeur de cabac_init_idc	Variables d'initialisation	ctxIdx													
		40	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**
0	**m**	-3	-6	-11	6	7	-5	2	0	-3	-10	5	4	-3	0
	n	69	81	96	55	67	86	88	58	76	94	54	69	81	88
1	**m**	-2	-5	-10	2	2	-3	-3	1	-3	-6	0	-3	-7	-5
	n	69	82	96	59	75	87	100	56	74	85	59	81	86	95
2	**m**	-11	-15	-21	19	20	4	6	1	-5	-13	5	6	-3	-1
	n	89	103	116	57	58	84	96	63	85	106	63	75	90	101

Tableau 9-16 – Valeurs des variables m et n pour ctxIdx de 54 à 59 et de 399 à 401

Valeur de cabac_init_idc	Variables d'initialisation	ctxIdx								
		54	**55**	**56**	**57**	**58**	**59**	**399**	**400**	**401**
Tranches I	**m**	na	na	na	na	na	na	31	31	25
	n	na	na	na	na	na	na	21	31	50
0	**m**	-7	-5	-4	-5	-7	1	12	11	14
	n	67	74	74	80	72	58	40	51	59
1	**m**	-1	-1	1	-2	-5	0	25	21	21
	n	66	77	70	86	72	61	32	49	54
2	**m**	3	-4	-2	-12	-7	1	21	19	17
	n	55	79	75	97	50	60	33	50	61

Tableau 9-17 – Valeurs des variables m et n pour ctxIdx de 60 à 69

Variables d'initialisation	ctxIdx									
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**
m	0	0	0	0	-9	4	0	-7	13	3
n	41	63	63	63	83	86	97	72	41	62

Tableau 9-18 – Valeurs des variables m et n pour ctxIdx de 70 à 104

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		**1**		**2**					**0**		**1**		**2**	
	m	**n**	**m**	**n**	**m**	**n**	**m**	**n**		**m**	**n**	**m**	**n**	**m**	**n**	**m**	**n**
70	0	11	0	45	13	15	7	34	**88**	-11	115	-13	108	-4	92	5	78
71	1	55	-4	78	7	51	-9	88	**89**	-12	63	-3	46	0	39	-6	55
72	0	69	-3	96	2	80	-20	127	**90**	-2	68	-1	65	0	65	4	61
73	-17	127	-27	126	-39	127	-36	127	**91**	-15	84	-1	57	-15	84	-14	83
74	-13	102	-28	98	-18	91	-17	91	**92**	-13	104	-9	93	-35	127	-37	127
75	0	82	-25	101	-17	96	-14	95	**93**	-3	70	-3	74	-2	73	-5	79
76	-7	74	-23	67	-26	81	-25	84	**94**	-8	93	-9	92	-12	104	-11	104
77	-21	107	-28	82	-35	98	-25	86	**95**	-10	90	-8	87	-9	91	-11	91
78	-27	127	-20	94	-24	102	-12	89	**96**	-30	127	-23	126	-31	127	-30	127
79	-31	127	-16	83	-23	97	-17	91	**97**	-1	74	5	54	3	55	0	65
80	-24	127	-22	110	-27	119	-31	127	**98**	-6	97	6	60	7	56	-2	79
81	-18	95	-21	91	-24	99	-14	76	**99**	-7	91	6	59	7	55	0	72
82	-27	127	-18	102	-21	110	-18	103	**100**	-20	127	6	69	8	61	-4	92
83	-21	114	-13	93	-18	102	-13	90	**101**	-4	56	-1	48	-3	53	-6	56
84	-30	127	-29	127	-36	127	-37	127	**102**	-5	82	0	68	0	68	3	68
85	-17	123	-7	92	0	80	11	80	**103**	-7	76	-4	69	-7	74	-8	71
86	-12	115	-5	89	-5	89	5	76	**104**	-22	125	-8	88	-9	88	-13	98
87	-16	122	-7	96	-7	94	2	84									

Tableau 9-19 – Valeurs des variables m et n pour ctxIdx de 105 à 165

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
105	-7	93	-2	85	-13	103	-4	86	**136**	-13	101	5	53	0	58	-5	75
106	-11	87	-6	78	-13	91	-12	88	**137**	-13	91	-2	61	-1	60	-8	80
107	-3	77	-1	75	-9	89	-5	82	**138**	-12	94	0	56	-3	61	-21	83
108	-5	71	-7	77	-14	92	-3	72	**139**	-10	88	0	56	-8	67	-21	64
109	-4	63	2	54	-8	76	-4	67	**140**	-16	84	-13	63	-25	84	-13	31
110	-4	68	5	50	-12	87	-8	72	**141**	-10	86	-5	60	-14	74	-25	64
111	-12	84	-3	68	-23	110	-16	89	**142**	-7	83	-1	62	-5	65	-29	94
112	-7	62	1	50	-24	105	-9	69	**143**	-13	87	4	57	5	52	9	75
113	-7	65	6	42	-10	78	-1	59	**144**	-19	94	-6	69	2	57	17	63
114	8	61	-4	81	-20	112	5	66	**145**	1	70	4	57	0	61	-8	74
115	5	56	1	63	-17	99	4	57	**146**	0	72	14	39	-9	69	-5	35
116	-2	66	-4	70	-78	127	-4	71	**147**	-5	74	4	51	-11	70	-2	27
117	1	64	0	67	-70	127	-2	71	**148**	18	59	13	68	18	55	13	91
118	0	61	2	57	-50	127	2	58	**149**	-8	102	3	64	-4	71	3	65
119	-2	78	-2	76	-46	127	-1	74	**150**	-15	100	1	61	0	58	-7	69
120	1	50	11	35	-4	66	-4	44	**151**	0	95	9	63	7	61	8	77
121	7	52	4	64	-5	78	-1	69	**152**	-4	75	7	50	9	41	-10	66
122	10	35	1	61	-4	71	0	62	**153**	2	72	16	39	18	25	3	62
123	0	44	11	35	-8	72	-7	51	**154**	-11	75	5	44	9	32	-3	68
124	11	38	18	25	2	59	-4	47	**155**	-3	71	4	52	5	43	-20	81
125	1	45	12	24	-1	55	-6	42	**156**	15	46	11	48	9	47	0	30
126	0	46	13	29	-7	70	-3	41	**157**	-13	69	-5	60	0	44	1	7
127	5	44	13	36	-6	75	-6	53	**158**	0	62	-1	59	0	51	-3	23
128	31	17	-10	93	-8	89	8	76	**159**	0	65	0	59	2	46	-21	74
129	1	51	-7	73	-34	119	-9	78	**160**	21	37	22	33	19	38	16	66
130	7	50	-2	73	-3	75	-11	83	**161**	-15	72	5	44	-4	66	-23	124
131	28	19	13	46	32	20	9	52	**162**	9	57	14	43	15	38	17	37
132	16	33	9	49	30	22	0	67	**163**	16	54	-1	78	12	42	44	-18
133	14	62	-7	100	-44	127	-5	90	**164**	0	62	0	60	9	34	50	-34
134	-13	108	9	53	0	54	1	67	**165**	12	72	9	69	0	89	-22	127
135	-15	100	2	53	-5	61	-15	72									

Tableau 9-20 – Valeurs des variables m et n pour ctxIdx de 166 à 226

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
166	24	0	11	28	4	45	4	39	**197**	26	-17	28	3	36	-28	28	-3
167	15	9	2	40	10	28	0	42	**198**	30	-25	28	4	38	-28	24	10
168	8	25	3	44	10	31	7	34	**199**	28	-20	32	0	38	-27	27	0
169	13	18	0	49	33	-11	11	29	**200**	33	-23	34	-1	34	-18	34	-14
170	15	9	0	46	52	-43	8	31	**201**	37	-27	30	6	35	-16	52	-44
171	13	19	2	44	18	15	6	37	**202**	33	-23	30	6	34	-14	39	-24
172	10	37	2	51	28	0	7	42	**203**	40	-28	32	9	32	-8	19	17
173	12	18	0	47	35	-22	3	40	**204**	38	-17	31	19	37	-6	31	25
174	6	29	4	39	38	-25	8	33	**205**	33	-11	26	27	35	0	36	29
175	20	33	2	62	34	0	13	43	**206**	40	-15	26	30	30	10	24	33
176	15	30	6	46	39	-18	13	36	**207**	41	-6	37	20	28	18	34	15
177	4	45	0	54	32	-12	4	47	**208**	38	1	28	34	26	25	30	20
178	1	58	3	54	102	-94	3	55	**209**	41	17	17	70	29	41	22	73
179	0	62	2	58	0	0	2	58	**210**	30	-6	1	67	0	75	20	34
180	7	61	4	63	56	-15	6	60	**211**	27	3	5	59	2	72	19	31
181	12	38	6	51	33	-4	8	44	**212**	26	22	9	67	8	77	27	44
182	11	45	6	57	29	10	11	44	**213**	37	-16	16	30	14	35	19	16
183	15	39	7	53	37	-5	14	42	**214**	35	-4	18	32	18	31	15	36
184	11	42	6	52	51	-29	7	48	**215**	38	-8	18	35	17	35	15	36
185	13	44	6	55	39	-9	4	56	**216**	38	-3	22	29	21	30	21	28
186	16	45	11	45	52	-34	4	52	**217**	37	3	24	31	17	45	25	21
187	12	41	14	36	69	-58	13	37	**218**	38	5	23	38	20	42	30	20
188	10	49	8	53	67	-63	9	49	**219**	42	0	18	43	18	45	31	12
189	30	34	-1	82	44	-5	19	58	**220**	35	16	20	41	27	26	27	16
190	18	42	7	55	32	7	10	48	**221**	39	22	11	63	16	54	24	42
191	10	55	-3	78	55	-29	12	45	**222**	14	48	9	59	7	66	0	93
192	17	51	15	46	32	1	0	69	**223**	27	37	9	64	16	56	14	56
193	17	46	22	31	0	0	20	33	**224**	21	60	-1	94	11	73	15	57
194	0	89	-1	84	27	36	8	63	**225**	12	68	-2	89	10	67	26	38
195	26	-19	25	7	33	-25	35	-18	**226**	2	97	-9	108	-10	116	-24	127
196	22	-17	30	-7	34	-30	33	-25									

Tableau 9-21 – Valeurs des variables m et n pour ctxIdx de 227 à 275

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
227	-3	71	-6	76	-23	112	-24	115	**252**	-12	73	-6	55	-16	72	-14	75
228	-6	42	-2	44	-15	71	-22	82	**253**	-8	76	0	58	-7	69	-10	79
229	-5	50	0	45	-7	61	-9	62	**254**	-7	80	0	64	-4	69	-9	83
230	-3	54	0	52	0	53	0	53	**255**	-9	88	-3	74	-5	74	-12	92
231	-2	62	-3	64	-5	66	0	59	**256**	-17	110	-10	90	-9	86	-18	108
232	0	58	-2	59	-11	77	-14	85	**257**	-11	97	0	70	2	66	-4	79
233	1	63	-4	70	-9	80	-13	89	**258**	-20	84	-4	29	-9	34	-22	69
234	-2	72	-4	75	-9	84	-13	94	**259**	-11	79	5	31	1	32	-16	75
235	-1	74	-8	82	-10	87	-11	92	**260**	-6	73	7	42	11	31	-2	58
236	-9	91	-17	102	-34	127	-29	127	**261**	-4	74	1	59	5	52	1	58
237	-5	67	-9	77	-21	101	-21	100	**262**	-13	86	-2	58	-2	55	-13	78
238	-5	27	3	24	-3	39	-14	57	**263**	-13	96	-3	72	-2	67	-9	83
239	-3	39	0	42	-5	53	-12	67	**264**	-11	97	-3	81	0	73	-4	81
240	-2	44	0	48	-7	61	-11	71	**265**	-19	117	-11	97	-8	89	-13	99
241	0	46	0	55	-11	75	-10	77	**266**	-8	78	0	58	3	52	-13	81
242	-16	64	-6	59	-15	77	-21	85	**267**	-5	33	8	5	7	4	-6	38
243	-8	68	-7	71	-17	91	-16	88	**268**	-4	48	10	14	10	8	-13	62
244	-10	78	-12	83	-25	107	-23	104	**269**	-2	53	14	18	17	8	-6	58
245	-6	77	-11	87	-25	111	-15	98	**270**	-3	62	13	27	16	19	-2	59
246	-10	86	-30	119	-28	122	-37	127	**271**	-13	71	2	40	3	37	-16	73
247	-12	92	1	58	-11	76	-10	82	**272**	-10	79	0	58	-1	61	-10	76
248	-15	55	-3	29	-10	44	-8	48	**273**	-12	86	-3	70	-5	73	-13	86
249	-10	60	-1	36	-10	52	-8	61	**274**	-13	90	-6	79	-1	70	-9	83
250	-6	62	1	38	-10	57	-8	66	**275**	-14	97	-8	85	-4	78	-10	87
251	-4	65	2	43	-9	58	-7	70									

Tableau 9-22 – Valeurs des variables m et n pour ctxIdx de 277 à 337

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
277	-6	93	-13	106	-21	126	-22	127	**308**	-16	96	-1	51	-16	77	-10	67
278	-6	84	-16	106	-23	124	-25	127	**309**	-7	88	7	49	-2	64	1	68
279	-8	79	-10	87	-20	110	-25	120	**310**	-8	85	8	52	2	61	0	77
280	0	66	-21	114	-26	126	-27	127	**311**	-7	85	9	41	-6	67	2	64
281	-1	71	-18	110	-25	124	-19	114	**312**	-9	85	6	47	-3	64	0	68
282	0	62	-14	98	-17	105	-23	117	**313**	-13	88	2	55	2	57	-5	78
283	-2	60	-22	110	-27	121	-25	118	**314**	4	66	13	41	-3	65	7	55
284	-2	59	-21	106	-27	117	-26	117	**315**	-3	77	10	44	-3	66	5	59
285	-5	75	-18	103	-17	102	-24	113	**316**	-3	76	6	50	0	62	2	65
286	-3	62	-21	107	-26	117	-28	118	**317**	-6	76	5	53	9	51	14	54
287	-4	58	-23	108	-27	116	-31	120	**318**	10	58	13	49	-1	66	15	44
288	-9	66	-26	112	-33	122	-37	124	**319**	-1	76	4	63	-2	71	5	60
289	-1	79	-10	96	-10	95	-10	94	**320**	-1	83	6	64	-2	75	2	70
290	0	71	-12	95	-14	100	-15	102	**321**	-7	99	-2	69	-1	70	-2	76
291	3	68	-5	91	-8	95	-10	99	**322**	-14	95	-2	59	-9	72	-18	86
292	10	44	-9	93	-17	111	-13	106	**323**	2	95	6	70	14	60	12	70
293	-7	62	-22	94	-28	114	-50	127	**324**	0	76	10	44	16	37	5	64
294	15	36	-5	86	-6	89	-5	92	**325**	-5	74	9	31	0	47	-12	70
295	14	40	9	67	-2	80	17	57	**326**	0	70	12	43	18	35	11	55
296	16	27	-4	80	-4	82	-5	86	**327**	-11	75	3	53	11	37	5	56
297	12	29	-10	85	-9	85	-13	94	**328**	1	68	14	34	12	41	0	69
298	1	44	-1	70	-8	81	-12	91	**329**	0	65	10	38	10	41	2	65
299	20	36	7	60	-1	72	-2	77	**330**	-14	73	-3	52	2	48	-6	74
300	18	32	9	58	5	64	0	71	**331**	3	62	13	40	12	41	5	54
301	5	42	5	61	1	67	-1	73	**332**	4	62	17	32	13	41	7	54
302	1	48	12	50	9	56	4	64	**333**	-1	68	7	44	0	59	-6	76
303	10	62	15	50	0	69	-7	81	**334**	-13	75	7	38	3	50	-11	82
304	17	46	18	49	1	69	5	64	**335**	11	55	13	50	19	40	-2	77
305	9	64	17	54	7	69	15	57	**336**	5	64	10	57	3	66	-2	77
306	-12	104	10	41	-7	69	1	67	**337**	12	70	26	43	18	50	25	42
307	-11	97	7	46	-6	67	0	68									

Tableau 9-23 – Valeurs des variables m et n pour ctxIdx de 338 à 398

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
338	15	6	14	11	19	-6	17	-13	**369**	32	-26	31	-4	40	-37	37	-17
339	6	19	11	14	18	-6	16	-9	**370**	37	-30	27	6	38	-30	32	1
340	7	16	9	11	14	0	17	-12	**371**	44	-32	34	8	46	-33	34	15
341	12	14	18	11	26	-12	27	-21	**372**	34	-18	30	10	42	-30	29	15
342	18	13	21	9	31	-16	37	-30	**373**	34	-15	24	22	40	-24	24	25
343	13	11	23	-2	33	-25	41	-40	**374**	40	-15	33	19	49	-29	34	22
344	13	15	32	-15	33	-22	42	-41	**375**	33	-7	22	32	38	-12	31	16
345	15	16	32	-15	37	-28	48	-47	**376**	35	-5	26	31	40	-10	35	18
346	12	23	34	-21	39	-30	39	-32	**377**	33	0	21	41	38	-3	31	28
347	13	23	39	-23	42	-30	46	-40	**378**	38	2	26	44	46	-5	33	41
348	15	20	42	-33	47	-42	52	-51	**379**	33	13	23	47	31	20	36	28
349	14	26	41	-31	45	-36	46	-41	**380**	23	35	16	65	29	30	27	47
350	14	44	46	-28	49	-34	52	-39	**381**	13	58	14	71	25	44	21	62
351	17	40	38	-12	41	-17	43	-19	**382**	29	-3	8	60	12	48	18	31
352	17	47	21	29	32	9	32	11	**383**	26	0	6	63	11	49	19	26
353	24	17	45	-24	69	-71	61	-55	**384**	22	30	17	65	26	45	36	24
354	21	21	53	-45	63	-63	56	-46	**385**	31	-7	21	24	22	22	24	23
355	25	22	48	-26	66	-64	62	-50	**386**	35	-15	23	20	23	22	27	16
356	31	27	65	-43	77	-74	81	-67	**387**	34	-3	26	23	27	21	24	30
357	22	29	43	-19	54	-39	45	-20	**388**	34	3	27	32	33	20	31	29
358	19	35	39	-10	52	-35	35	-2	**389**	36	-1	28	23	26	28	22	41
359	14	50	30	9	41	-10	28	15	**390**	34	5	28	24	30	24	22	42
360	10	57	18	26	36	0	34	1	**391**	32	11	23	40	27	34	16	60
361	7	63	20	27	40	-1	39	1	**392**	35	5	24	32	18	42	15	52
362	-2	77	0	57	30	14	30	17	**393**	34	12	28	29	25	39	14	60
363	-4	82	-14	82	28	26	20	38	**394**	39	11	23	42	18	50	3	78
364	-3	94	-5	75	23	37	18	45	**395**	30	29	19	57	12	70	-16	123
365	9	69	-19	97	12	55	15	54	**396**	34	26	22	53	21	54	21	53
366	-12	109	-35	125	11	65	0	79	**397**	29	39	22	61	14	71	22	56
367	36	-35	27	0	37	-33	36	-16	**398**	19	66	11	86	11	83	25	61
368	36	-34	28	0	39	-36	37	-14									

Tableau 9-24 – Valeurs des variables m et n pour ctxIdx de 402 à 459

ctxIdx	Tranches I		Valeur de cabac_init_idc						ctxIdx	Tranches I		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
402	-17	120	-4	79	-5	85	-3	78	**431**	-2	55	-12	56	-9	57	-12	59
403	-20	112	-7	71	-6	81	-8	74	**432**	0	61	-6	60	-6	63	-8	63
404	-18	114	-5	69	-10	77	-9	72	**433**	1	64	-5	62	-4	65	-9	67
405	-11	85	-9	70	-7	81	-10	72	**434**	0	68	-8	66	-4	67	-6	68
406	-15	92	-8	66	-17	80	-18	75	**435**	-9	92	-8	76	-7	82	-10	79
407	-14	89	-10	68	-18	73	-12	71	**436**	-14	106	-5	85	-3	81	-3	78
408	-26	71	-19	73	-4	74	-11	63	**437**	-13	97	-6	81	-3	76	-8	74
409	-15	81	-12	69	-10	83	-5	70	**438**	-15	90	-10	77	-7	72	-9	72
410	-14	80	-16	70	-9	71	-17	75	**439**	-12	90	-7	81	-6	78	-10	72
411	0	68	-15	67	-9	67	-14	72	**440**	-18	88	-17	80	-12	72	-18	75
412	-14	70	-20	62	-1	61	-16	67	**441**	-10	73	-18	73	-14	68	-12	71
413	-24	56	-19	70	-8	66	-8	53	**442**	-9	79	-4	74	-3	70	-11	63
414	-23	68	-16	66	-14	66	-14	59	**443**	-14	86	-10	83	-6	76	-5	70
415	-24	50	-22	65	0	59	-9	52	**444**	-10	73	-9	71	-5	66	-17	75
416	-11	74	-20	63	2	59	-11	68	**445**	-10	70	-9	67	-5	62	-14	72
417	23	-13	9	-2	17	-10	9	-2	**446**	-10	69	-1	61	0	57	-16	67
418	26	-13	26	-9	32	-13	30	-10	**447**	-5	66	-8	66	-4	61	-8	53
419	40	-15	33	-9	42	-9	31	-4	**448**	-9	64	-14	66	-9	60	-14	59
420	49	-14	39	-7	49	-5	33	-1	**449**	-5	58	0	59	1	54	-9	52
421	44	3	41	-2	53	0	33	7	**450**	2	59	2	59	2	58	-11	68
422	45	6	45	3	64	3	31	12	**451**	21	-10	21	-13	17	-10	9	-2
423	44	34	49	9	68	10	37	23	**452**	24	-11	33	-14	32	-13	30	-10
424	33	54	45	27	66	27	31	38	**453**	28	-8	39	-7	42	-9	31	-4
425	19	82	36	59	47	57	20	64	**454**	28	-1	46	-2	49	-5	33	-1
426	-3	75	-6	66	-5	71	-9	71	**455**	29	3	51	2	53	0	33	7
427	-1	23	-7	35	0	24	-7	37	**456**	29	9	60	6	64	3	31	12
428	1	34	-7	42	-1	36	-8	44	**457**	35	20	61	17	68	10	37	23
429	1	43	-8	45	-2	42	-11	49	**458**	29	36	55	34	66	27	31	38
430	0	54	-5	48	-2	52	-10	56	**459**	14	67	42	62	47	57	20	64

9.3.1.2 Processus d'initialisation pour le moteur de décodage arithmétique

Ce processus est invoqué avant le décodage du premier macrobloc d'une tranche ou après le décodage de tout élément pcm_alignment_zero_bit et de toutes les données pcm_sample_luma et pcm_sample_chroma pour un macrobloc de type I_PCM.

Les résultats de ce processus sont les registres initialisés du moteur de décodage codIRange et codIOffset, tous deux avec une précision de registre de 16 bit.

L'état du moteur de décodage arithmétique est représenté par les variables codIRange et codIOffset. Dans la procédure d'initialisation du processus de décodage arithmétique, codIRange est mis égal à 0x01FE et codIOffset est mis égal à la valeur résultant de read_bits(9) interprété comme une représentation binaire de 9 bit d'un entier non signé avec le bit de plus fort poids écrit en premier.

Le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de codIOffset égale à 0x01FE ou 0x01FF.

NOTE – La description du moteur de décodage arithmétique utilise dans la présente Recommandation | Norme internationale une précision de registre de 16 bit. Cependant, la précision de registre minimale pour les variables codIRange et codIOffset est de 9 bit.

9.3.2 Processus de binarisation

L'entrée dans ce processus est une demande d'élément syntaxique.

Le résultat de ce processus est la binarisation de l'élément syntaxique, maxBinIdxCtx, ctxIdxOffset, et bypassFlag.

Le Tableau 9-25 spécifie le type de processus de binarisation, maxBinIdxCtx, et ctxIdxOffset associé à chaque élément syntaxique.

La spécification du processus de binarisation unaire (U), le processus de binarisation unaire tronqué (TU, *truncated unary*), le processus de binarisation Exp-Golomb de $k^{ième}$ ordre unaire concaténé (UEGk, *concatenated unary / k-th order Exp-Golomb*), et le processus de binarisation de longueur fixe (FL, *fixed-length*) sont donnés respectivement aux § 9.3.2.1 à 9.3.2.4. D'autres binarisations sont spécifiées aux § 9.3.2.5 à 9.3.2.7.

Excepté pour les tranches I, les binarisations pour l'élément syntaxique mb_type, comme spécifié au § 9.3.2.5, consistent en une chaîne de segments donnés par un enchaînement de chaînes binaires de préfixes et de suffixes. La binarisation UEGk, spécifiée au § 9.3.2.3, qui est utilisée pour la binarisation des éléments syntaxiques mvd_lX (X = 0, 1) et coeff_abs_level_minus1, et la binarisation du coded_block_pattern consistent aussi en un enchaînement de chaînes binaires de préfixes et de suffixes. Pour ces processus de binarisation, la chaîne binaire de préfixes et celle de suffixes sont indexées séparément au moyen de la variable binIdx comme spécifié plus loin au § 9.3.3. Les deux ensembles de chaînes binaires de préfixes et de suffixes sont désignés respectivement comme la partie préfixe de la binarisation et la partie suffixe de la binarisation.

Une valeur spécifique de la variable de décalage d'indice de contexte (ctxIdxOffset) et une valeur spécifique de la variable maxBinIdxCtx sont associées à chaque binarisation ou partie de binarisation d'un élément syntaxique, comme indiqué au Tableau 9-25. Lorsque deux valeurs pour chacune de ces variables sont spécifiées pour un élément syntaxique dans le Tableau 9-25, la valeur dans la rangée supérieure se rapporte à la partie préfixe tandis que la valeur dans la rangée inférieure se rapporte à la partie suffixe de la binarisation de l'élément syntaxique correspondant.

L'utilisation du processus DecodeBypass et de la variable bypassFlag est déduite comme suit.

– Si aucune valeur n'est allouée à ctxIdxOffset pour la binarisation ou partie de binarisation correspondante marquée "na" dans le Tableau 9-25, tous les segments de la chaîne binaire de la binarisation ou partie de préfixe/suffixe de binarisation correspondante sont décodés en invoquant le processus DecodeBypass comme spécifié au § 9.3.3.2.3. Dans un tel cas, bypassFlag est mis égal à 1, et bypassFlag est utilisé afin d'indiquer que pour l'analyse syntaxique de la valeur du segment à partir du flux binaire, le processus DecodeBypass est appliqué.

– Sinon, pour chaque valeur possible de binIdx jusqu'à la valeur spécifiée de maxBinIdxCtx donnée au Tableau 9-25, une valeur spécifique de la variable ctxIdx est spécifiée plus en détail au § 9.3.3. bypassFlag est mis égal à 0.

Les valeurs possibles de l'indice de contexte ctxIdx sont dans la gamme de 0 à 459 inclus. La valeur allouée à ctxIdxOffset spécifie la valeur inférieure de la gamme de ctxIdx allouée à la binarisation ou partie de binarisation correspondante d'un élément syntaxique.

ctxIdx = ctxIdxOffset = 276 est alloué à l'élément syntaxique end_of_slice_flag et au segment de mb_type, qui spécifie le type de macrobloc I_PCM comme spécifié plus en détail au § 9.3.3.1. Pour l'analyse syntaxique de la valeur du segment correspondant provenant du flux binaire, on applique le processus arithmétique de décodage pour les décisions avant la terminaison (DecodeTerminate), comme spécifié au § 9.3.3.2.4.

NOTE – Les segments de mb_type dans les tranches I et les segments du suffixe pour mb_type dans les tranches SI qui correspondent à la même valeur de binIdx partagent le même ctxIdx. Le dernier segment du préfixe de mb_type et le premier segment du suffixe de mb_type dans les tranches P, SP, et B peuvent partager le même ctxIdx.

Tableau 9-25 – Eléments syntaxiques et types de binarisation associés, maxBinIdxCtx et ctxIdxOffset

Elément syntaxique	Type de binarisation	maxBinIdxCtx	ctxIdxOffset
mb_type (seulement tranches SI)	Préfixe et suffixe comme spécifié au § 9.3.2.5	Préfixe: 0 Suffixe: 6	Préfixe: 0 Suffixe: 3
mb_type (seulement tranches I)	Comme spécifié au § 9.3.2.5	6	3
mb_skip_flag (seulement tranches P, SP)	FL, cMax=1	0	11
mb_type (seulement tranches P, SP)	Préfixe et suffixe comme spécifié au § 9.3.2.5	Préfixe: 2 Suffixe: 5	Préfixe: 14 Suffixe: 17
sub_mb_type (seulement tranches P, SP)	Comme spécifié au § 9.3.2.5	2	21
mb_skip_flag (seulement tranches B)	FL, cMax=1	0	24
mb_type (seulement tranches B)	Préfixe et suffixe comme spécifié au § 9.3.2.5	Préfixe: 3 Suffixe: 5	Préfixe: 27 Suffixe: 32
sub_mb_type (seulement tranches B)	Comme spécifié au § 9.3.2.5	3	36
mvd_l0[ ][ ][0], mvd_l1[ ][ ][0]	Préfixe et suffixe comme donné par UEG3, avec signedValFlag=1, uCoff=9	Préfixe: 4 Suffixe: na	Préfixe: 40 Suffixe: na (utilise DecodeBypass)
mvd_l0[ ][ ][1], mvd_l1[ ][ ][1]		Préfixe: 4 Suffixe: na	Préfixe: 47 Suffixe: na (utilise DecodeBypass)
ref_idx_l0, ref_idx_l1	U	2	54
mb_qp_delta	Comme spécifié au § 9.3.2.7	2	60
intra_chroma_pred_mode	TU, cMax=3	1	64
prev_intra4x4_pred_mode_flag, prev_intra8x8_pred_mode_flag	FL, cMax=1	0	68
rem_intra4x4_pred_mode, rem_intra8x8_pred_mode	FL, cMax=7	0	69
mb_field_decoding_flag	FL, cMax=1	0	70
coded_block_pattern	Préfixe et suffixe comme spécifié au § 9.3.2.6	Préfixe: 3 Suffixe: 1	Préfixe: 73 Suffixe: 77
coded_block_flag	FL, cMax=1	0	85
significant_coeff_flag (blocs codés de trame avec ctxBlockCat < 5)	FL, cMax=1	0	105
last_significant_coeff_flag (blocs codés de trame avec ctxBlockCat < 5)	FL, cMax=1	0	166
coeff_abs_level_minus1 (avec ctxBlockCat < 5)	Préfixe et suffixe comme donné par UEG0 avec signedValFlag=0, uCoff=14	Préfixe: 1 Suffixe: na	Préfixe: 227 Suffixe: na, (utilise DecodeBypass)
coeff_sign_flag	FL, cMax=1	0	na, (utilise DecodeBypass)
end_of_slice_flag	FL, cMax=1	0	276
significant_coeff_flag (blocs codés de sous-trame avec ctxBlockCat < 5)	FL, cMax=1	0	277
last_significant_coeff_flag (blocs codés de sous-trame avec ctxBlockCat < 5)	FL, cMax=1	0	338
transform_size_8x8_flag	FL, cMax=1	0	399
significant_coeff_flag (frame codé blocs avec ctxBlocCat == 5)	FL, cMax=1	0	402
last_significant_coeff_flag (frame codé blocs avec ctxBlocCat == 5)	FL, cMax=1	0	417
coeff_abs_level_minus1 (blocs avec ctxBlocCat == 5)	préfixe et suffixe tels que donnés par UEG0 avec signedValFlag=0, uCoff=14	préfixe: 1 suffixe: na	préfixe: 426 suffixe: na, (utilise DecodeBypass)
significant_coeff_flag (blocs de sous-trame codés avec ctxBlocCat == 5)	FL, cMax=1	0	436
last_significant_coeff_flag (blocs de sous-trame codés avec ctxBlocCat == 5)	FL, cMax=1	0	451

9.3.2.1 Processus de binarisation unaire (U)

L'entrée dans ce processus est une demande de binarisation U pour un élément syntaxique.

Le résultat de ce processus est la binarisation U de l'élément syntaxique.

La chaîne de segments d'un élément syntaxique ayant une valeur (un entier arithmétique) synElVal est une chaîne binaire de longueur synElVal + 1 indexée par BinIdx. Les segments qui pour binIdx sont inférieurs à synElVal sont égaux à 1. Le segment avec binIdx égal à synElVal est égal à 0.

Le Tableau 9-26 illustre les chaînes de segments de la binarisation unaire pour un élément syntaxique.

Tableau 9-26 – Chaîne de segments de la binarisation unaire (pour information)

Valeur de l'élément syntaxique	Chaîne de segments					
0 (I_NxN)	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
…						
binIdx	0	1	2	3	4	5

9.3.2.2 Processus de binarisation unaire tronqué (TU)

L'entrée dans ce processus est une demande de binarisation TU pour un élément syntaxique et cMax.

Le résultat de ce processus est la binarisation TU de l'élément syntaxique.

Pour les valeurs d'élément syntaxique (entier non signé) inférieures à cMax, le processus de binarisation U est invoqué, comme spécifié au § 9.3.2.1. Pour la valeur d'élément syntaxique égale à cMax la chaîne de segments est une chaîne binaire de longueur cMax avec tous les segments égaux à 1.

NOTE – La binarisation TU est toujours invoquée avec une valeur de cMax égale à la plus grande valeur possible de l'élément syntaxique en cours de décodage.

9.3.2.3 Processus de binarisation Exp-Golomb de k^{ième} ordre unaire concaténé (UEGk)

L'entrée dans ce processus est une demande de binarisation UEGk pour un élément syntaxique, signedValFlag et uCoff.

Le résultat de ce processus est la binarisation UEGk de l'élément syntaxique.

Une chaîne de segments UEGk est un enchaînement d'une chaîne binaire de préfixe et d'une chaîne binaire de suffixe. Le préfixe de la binarisation est spécifié par l'invocation du processus de binarisation TU pour la partie préfixe Min(uCoff, Abs(synElVal)) d'une valeur d'élément syntaxique synElVal comme spécifié au § 9.3.2.2 avec cMax = uCoff, où uCoff > 0.

La chaîne de segments UEGk est déduite comme suit.

– Si une des conditions suivantes est vraie, la chaîne de segments d'un élément syntaxique qui a une valeur synElVal ne comporte seulement qu'une chaîne binaire de préfixe,

  – signedValFlag est égal à 0 et la chaîne binaire préfixe n'est pas égale à la chaîne binaire de longueur uCoff avec tous les bits égaux à 1;

  – signedValFlag est égal à 1 et la chaîne binaire préfixe est égale à la chaîne binaire qui comporte un seul bit d'une valeur égale à 0.

– Sinon, la chaîne de segments de la partie suffixe UEGk d'une valeur d'un élément syntaxique synElVal est spécifiée par un processus équivalent au pseudo-code suivant:

```
si( Abs( synElVal ) >= uCoff ) {
    sufS = Abs( synElVal ) − uCoff
```

```
    stopLoop = 0
    faire {
        si( sufS >= ( 1 << k ) ) {
            mettre( 1 )
            sufS = sufS − ( 1<<k )
            k++
        } ou {
            mettre( 0 )
            tant que( k− − )
                mettre( ( sufS >> k ) & 0x01 )
            stopLoop = 1
        }
    } tant que( !stopLoop )
}
si( signedValFlag && synElVal != 0)
    si( synElVal > 0 )
        mettre( 0 )
    ou
        mettre( 1 )
```

NOTE – La spécification pour le code Exp-Golomb de k^{ième} ordre (EGk) utilise une signification inversée pour les 1 et les 0 pour la partie unaire du code Exp-Golomb d'ordre 0, comme spécifié au § 9.1.

9.3.2.4 Processus de binarisation de longueur fixe (FL)

L'entrée dans ce processus est une demande de binarisation FL pour un élément syntaxique et cMax.

Le résultat de ce processus est la binarisation FL de l'élément syntaxique.

La binarisation FL est construite au moyen d'une chaîne de segments d'entiers arithmétiques fixedLength-bit de la valeur de l'élément syntaxique, où fixedLength = Ceil(Log2(cMax + 1)). L'indexation des segments pour la binarisation FL est telle que binIdx = 0 se rapporte au bit de plus faible poids et que les valeurs croissantes de binIdx vont vers le bit de plus fort poids.

9.3.2.5 Processus de binarisation pour le type de macrobloc et de sous-macrobloc

L'entrée dans ce processus est une demande de binarisation pour les éléments syntaxiques mb_type ou sub_mb_type.

Le résultat de ce processus est la binarisation de l'élément syntaxique.

Le schéma de binarisation pour le décodage du type de macrobloc dans les tranches I est spécifié au Tableau 9-27.

Pour les types de macrobloc de tranches SI, la binarisation consiste en chaînes de segments spécifiées comme la concaténation d'une chaîne binaire de préfixe et de suffixe comme suit.

La chaîne binaire de préfixe consiste en un seul bit, qui est spécifié par b_0 = ((mb_type == SI) ? 0: 1). Pour la valeur de l'élément syntaxique pour laquelle b_0 est égal à 0, la chaîne de segments ne comporte que la chaîne binaire de préfixe. Pour la valeur d'élément syntaxique pour laquelle b_0 est égal à 1, la binarisation est donnée par la concaténation du préfixe b_0 et de la chaîne binaire de suffixe comme spécifié au Tableau 9-27 pour le type de macrobloc des tranches I indexé en soustrayant 1 de la valeur de mb_type dans les tranches SI.

Tableau 9-27 – Binarisation pour les types de macrobloc dans les tranches I

Valeur (nom) de mb_type	Chaîne de segments						
0 (I_4x4)	0						
1 (I_16x16_0_0_0)	1	0	0	0	0	0	
2 (I_16x16_1_0_0)	1	0	0	0	0	1	
3 (I_16x16_2_0_0)	1	0	0	0	1	0	
4 (I_16x16_3_0_0)	1	0	0	0	1	1	
5 (I_16x16_0_1_0)	1	0	0	1	0	0	0
6 (I_16x16_1_1_0)	1	0	0	1	0	0	1
7 (I_16x16_2_1_0)	1	0	0	1	0	1	0
8 (I_16x16_3_1_0)	1	0	0	1	0	1	1
9 (I_16x16_0_2_0)	1	0	0	1	1	0	0
10 (I_16x16_1_2_0)	1	0	0	1	1	0	1
11 (I_16x16_2_2_0)	1	0	0	1	1	1	0
12 (I_16x16_3_2_0)	1	0	0	1	1	1	1
13 (I_16x16_0_0_1)	1	0	1	0	0	0	
14 (I_16x16_1_0_1)	1	0	1	0	0	1	
15 (I_16x16_2_0_1)	1	0	1	0	1	0	
16 (I_16x16_3_0_1)	1	0	1	0	1	1	
17 (I_16x16_0_1_1)	1	0	1	1	0	0	0
18 (I_16x16_1_1_1)	1	0	1	1	0	0	1
19 (I_16x16_2_1_1)	1	0	1	1	0	1	0
20 (I_16x16_3_1_1)	1	0	1	1	0	1	1
21 (I_16x16_0_2_1)	1	0	1	1	1	0	0
22 (I_16x16_1_2_1)	1	0	1	1	1	0	1
23 (I_16x16_2_2_1)	1	0	1	1	1	1	0
24 (I_16x16_3_2_1)	1	0	1	1	1	1	1
25 (I_PCM)	1	1					
binIdx	0	1	2	3	4	5	6

Les schémas de binarisation pour les types de macrobloc P dans les tranches P et SP et pour les macroblocs B dans les tranches B sont spécifiés au Tableau 9-28.

La chaîne de segments pour les types de macrobloc I dans les tranches P et SP correspondant aux valeurs mb_type de 5 à 30 consiste en une concaténation d'un préfixe, qui consiste en un seul bit avec une valeur égale à 1, comme spécifié au Tableau 9-28 et d'un suffixe, comme spécifié au Tableau 9-27, indexé en soustrayant 5 de la valeur de mb_type.

mb_type égal à 4 (P_8x8ref0) n'est pas autorisé.

Pour les types de macrobloc I dans les tranches B (valeurs de mb_type de 23 à 48) la binarisation consiste en chaînes de segments spécifiées comme concaténation d'une chaîne binaire de préfixe, comme spécifié au Tableau 9-28, et de chaînes binaires de suffixe, comme spécifié au Tableau 9-27, indexées en subtrayant 23 de la valeur de mb_type.

Tableau 9-28 – Binarisation pour les types de macrobloc dans les tranches P, SP, et B

Type de tranche	Valeur (nom) de mb_type	Chaîne de segments						
Tranches P, SP	0 (P_L0_16x16)	0	0	0				
	1 (P_L0_L0_16x8)	0	1	1				
	2 (P_L0_L0_8x16)	0	1	0				
	3 (P_8x8)	0	0	1				
	4 (P_8x8ref0)	na						
	5 to 30 (Intra, seulement préfixe)	1						
Tranche B	0 (B_Direct_16x16)	0						
	1 (B_L0_16x16)	1	0	0				
	2 (B_L1_16x16)	1	0	1				
	3 (B_Bi_16x16)	1	1	0	0	0	0	
	4 (B_L0_L0_16x8)	1	1	0	0	0	1	
	5 (B_L0_L0_8x16)	1	1	0	0	1	0	
	6 (B_L1_L1_16x8)	1	1	0	0	1	1	
	7 (B_L1_L1_8x16)	1	1	0	1	0	0	
	8 (B_L0_L1_16x8)	1	1	0	1	0	1	
	9 (B_L0_L1_8x16)	1	1	0	1	1	0	
	10 (B_L1_L0_16x8)	1	1	0	1	1	1	
	11 (B_L1_L0_8x16)	1	1	1	1	1	0	
	12 (B_L0_Bi_16x8)	1	1	1	0	0	0	0
	13 (B_L0_Bi_8x16)	1	1	1	0	0	0	1
	14 (B_L1_Bi_16x8)	1	1	1	0	0	1	0
	15 (B_L1_Bi_8x16)	1	1	1	0	0	1	1
	16 (B_Bi_L0_16x8)	1	1	1	0	1	0	0
	17 (B_Bi_L0_8x16)	1	1	1	0	1	0	1
	18 (B_Bi_L1_16x8)	1	1	1	0	1	1	0
	19 (B_Bi_L1_8x16)	1	1	1	0	1	1	1
	20 (B_Bi_Bi_16x8)	1	1	1	1	0	0	0
	21 (B_Bi_Bi_8x16)	1	1	1	1	0	0	1
	22 (B_8x8)	1	1	1	1	1	1	
	23 to 48 (Intra, seulement préfixe)	1	1	1	1	0	1	
binIdx		0	1	2	3	4	5	6

Pour les tranches P, SP, et B la spécification de la binarisation pour sub_mb_type est donnée au Tableau 9-29.

Tableau 9-29 – Binarisation pour les types de sous-macrobloc dans les tranches P, SP, et B

Type de tranche	Valeur (nom) de mb_type	Chaîne de segments					
Tranches P, SP	0 (P_L0_8x8)	1					
	1 (P_L0_8x4)	0	0				
	2 (P_L0_4x8)	0	1	1			
	3 (P_L0_4x4)	0	1	0			
Tranche B	0 (B_Direct_8x8)	0					
	1 (B_L0_8x8)	1	0	0			
	2 (B_L1_8x8)	1	0	1			
	3 (B_Bi_8x8)	1	1	0	0	0	
	4 (B_L0_8x4)	1	1	0	0	1	
	5 (B_L0_4x8)	1	1	0	1	0	
	6 (B_L1_8x4)	1	1	0	1	1	
	7 (B_L1_4x8)	1	1	1	0	0	0
	8 (B_Bi_8x4)	1	1	1	0	0	1
	9 (B_Bi_4x8)	1	1	1	0	1	0
	10 (B_L0_4x4)	1	1	1	0	1	1
	11 (B_L1_4x4)	1	1	1	1	0	
	12 (B_Bi_4x4)	1	1	1	1	1	
binIdx		0	1	2	3	4	5

9.3.2.6 Processus de binarisation pour schéma de bloc codé

L'entrée dans ce processus est une demande de binarisation pour l'élément syntaxique coded_block_pattern.

Le résultat de ce processus est la binarisation de l'élément syntaxique.

La binarisation de coded_block_pattern consiste en une partie de préfixe et en (le cas échéant) une partie de suffixe. La partie préfixe de la binarisation est donnée par la binarisation FL de CodedBlockPatternLuma avec cMax = 15. Lorsque chroma_format_idc n'est pas égal à 0 (format monochrome), la partie suffixe est présente et consiste en la binarisation TU de CodedBlockPatternChroma avec cMax = 2. La relation entre la valeur de l'élément syntaxique coded_block_pattern et les valeurs de CodedBlockPatternLuma et CodedBlockPatternChroma est donnée comme spécifié au § 7.4.5.

9.3.2.7 Processus de binarisation pour mb_qp_delta

L'entrée dans ce processus est une demande de binarisation pour l'élément syntaxique mb_qp_delta.

Le résultat de ce processus est la binarisation de l'élément syntaxique.

La chaîne de segments de mb_qp_delta est déduite par la binarisation U de la valeur correspondante de l'élément syntaxique mb_qp_delta, où la règle d'allocation entre la valeur algébrique de mb_qp_delta et sa valeur correspondante est donnée comme spécifié au Tableau 9-3.

9.3.3 Flux de processus de décodage

L'entrée dans ce processus est une binarisation de l'élément syntaxique demandé, maxBinIdxCtx, bypassFlag et ctxIdxOffset comme spécifié au § 9.3.2.

Le résultat de ce processus est la valeur de l'élément syntaxique.

Ce processus spécifie comment chaque bit d'une chaîne binaire est analysé syntaxiquement pour chaque élément syntaxique.

Après l'analyse syntaxique de chaque bit, la chaîne binaire résultante est comparée à toutes les chaînes de segments de la binarisation de l'élément syntaxique et on applique ce qui suit.

– Si la chaîne binaire est égale à une des chaînes de segments, la valeur correspondante de l'élément syntaxique est le résultat.

– Sinon (si la chaîne binaire n'est pas égale à une des chaînes de segments), le bit suivant est analysé syntaxiquement.

Lors de l'analyse syntaxique de chaque segment, la variable binIdx est incrémentée de 1 à partir de binIdx mise égale à 0 pour le premier segment.

Lorsque la binarisation de l'élément syntaxique correspondant consiste en une partie de binarisation de préfixe et de suffixe, la variable binIdx est mise égale à 0 pour le premier segment de chaque partie de la chaîne de segments (partie préfixe ou partie suffixe). Dans ce cas, après l'analyse syntaxique de la chaîne binaire de préfixe, le processus d'analyse syntaxique de la chaîne binaire de suffixe se rapportant aux binarisations spécifiées aux § 9.3.2.3 et 9.3.2.5 est invoquée en fonction de la chaîne binaire de préfixe résultante, comme spécifié aux § 9.3.2.3 et 9.3.2.5. Noter que pour la binarisation de l'élément syntaxique coded_block_pattern, la chaîne binaire de suffixe est présente sans considération de la chaîne binaire de préfixe de longueur 4 comme spécifié au § 9.3.2.6.

En fonction de la variable bypassFlag, on applique ce qui suit.

– Si bypassFlag est égal à 1, on applique le processus de décodage de dérivation, comme spécifié au § 9.3.3.2.3, afin d'analyser syntaxiquement la valeur des segments provenant du flux binaire.

– Sinon (si bypassFlag est égal à 0), l'analyse syntaxique de chaque segment est spécifiée par les deux étapes ordonnées suivantes:

  1. étant donné binIdx, maxBinIdxCtx et ctxIdxOffset, ctxIdx est calculé comme spécifié au § 9.3.3.1;

  2. étant donné ctxIdx, on décode la valeur du segment provenant du flux binaire, comme spécifié au § 9.3.3.2.

9.3.3.1 Processus de déduction pour ctxIdx

Les entrées dans ce processus sont binIdx, maxBinIdxCtx et ctxIdxOffset.

Le résultat de ce processus est ctxIdx.

Le Tableau 9-30 donne les allocations des incréments de ctxIdx (ctxIdxInc) à binIdx pour toutes les valeurs de ctxIdxOffset excepté celles qui se rapportent aux éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1.

Le ctxIdx à utiliser avec un binIdx spécifique est spécifié en déterminant d'abord le ctxIdxOffset associé à la chaîne de segments donnée ou sa partie. Le ctxIdx est déterminé comme suit.

– Si le ctxIdxOffset figure dans le Tableau 9-30, le ctxIdx pour un binIdx est la somme de ctxIdxOffset et de ctxIdxInc, qu'on trouve au Tableau 9-30. Lorsque plus d'une valeur figure dans le Tableau 9-30 pour un binIdx, le processus d'allocation pour ctxIdxInc pour ce binIdx est spécifiée plus en détail dans les paragraphes donnés entre parenthèses dans l'entrée de tableau correspondante.

– Sinon (si ctxIdxOffset ne figure pas dans le Tableau 9-30), le ctxIdx est spécifié comme la somme des termes suivants: ctxIdxOffset et ctxIdxBlockCatOffset(ctxBlockCat) comme spécifié au Tableau 9-31 et ctxIdxInc(ctxBlockCat). Le paragraphe 9.3.3.1.3 spécifie quel ctxBlockCat est utilisé. Le § 9.3.3.1.1.9 spécifie l'allocation de ctxIdxInc(ctxBlockCat) pour coded_block_flag, et le § 9.3.3.1.3 spécifie l'allocation de ctxIdxInc(ctxBlockCat) pour significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1.

Tous les segments avec binIdx plus grand que maxBinIdxCtx sont analysés syntaxiquement au moyen de la valeur de ctxIdx qui est allouée à binIdx égal à maxBinIdxCtx.

Toutes les entrées du Tableau 9-30 marquées "na" correspondent aux valeurs de binIdx qui n'interviennent pas pour le ctxIdxOffset correspondant.

ctxIdx = 276 est alloué au binIdx of mb_type indiquant le mode I_PCM. afin d'analyser syntaxiquement la valeur des segments correspondants provenant du flux binaire, on applique le processus de décodage arithmétique pour les décisions avant achèvement, comme spécifié au § 9.3.3.2.4.

Tableau 9-30 – Allocation de ctxIdxInc à binIdx pour toutes les valeurs de ctxIdxOffset excepté celles se rapportant aux éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1

ctxIdxOffset	binIdx						
	0	**1**	**2**	**3**	**4**	**5**	**>= 6**
0	0,1,2 (§ 9.3.3.1.1.3)	na	na	na	na	na	na
3	0,1,2 (§ 9.3.3.1.1.3)	ctxIdx=276	3	4	5,6 (§ 9.3.3.1.2)	6,7 (§ 9.3.3.1.2)	7
11	0,1,2 (§ 9.3.3.1.1.1)	na	na	na	na	na	na
14	0	1	2,3 (§ 9.3.3.1.2)	na	na	na	na
17	0	ctxIdx=276	1	2	2,3 (§ 9.3.3.1.2)	3	3
21	0	1	2	na	na	na	na
24	0,1,2 (§ 9.3.3.1.1.1)	na	na	na	na	na	na
27	0,1,2 (§ 9.3.3.1.1.3)	3	4,5 (§ 9.3.3.1.2)	5	5	5	5
32	0	ctxIdx=276	1	2	2,3 (§ 9.3.3.1.2)	3	3
36	0	1	2,3 (§ 9.3.3.1.2)	3	3	3	na
40	0,1,2 (§ 9.3.3.1.1.7)	3	4	5	6	6	6
47	0,1,2 (§ 9.3.3.1.1.7)	3	4	5	6	6	6
54	0,1,2,3 (§ 9.3.3.1.1.6)	4	5	5	5	5	5
60	0,1 (§ 9.3.3.1.1.5)	2	3	3	3	3	3
64	0,1,2 (§ 9.3.3.1.1.8)	3	3	na	na	na	na
68	0	na	na	na	na	na	na
69	0	0	0	na	na	na	na
70	0,1,2 (§ 9.3.3.1.1.2)	na	na	na	na	na	na
73	0,1,2,3 (§ 9.3.3.1.1.4)	0,1,2,3 (§ 9.3.3.1.1.4)	0,1,2,3 (§ 9.3.3.1.1.4)	0,1,2,3 (§ 9.3.3.1.1.4)	na	na	na
77	0,1,2,3 (§ 9.3.3.1.1.4)	4,5,6,7 (§ 9.3.3.1.1.4)	na	na	na	na	na
276	0	na	na	na	na	na	na
399	0,1,2 § 9.3.3.1.1.10	na	na	na	na	na	na

Le Tableau 9-31 donne les valeurs de ctxIdxBlockCatOffset en fonction de ctxBlockCat pour les éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1. La spécification de ctxBlockCat est donnée au Tableau 9-33.

Tableau 9-31 – Allocation de ctxIdxBlockCatOffset à ctxBlockCat pour les éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1

Elément syntaxique	ctxBlocCat (comme spécifié dans le Tableau 9-33)					
	0	**1**	**2**	**3**	**4**	**5**
coded_block_flag	0	4	8	12	16	na
significant_coeff_flag	0	15	29	44	47	0
last_significant_coeff_flag	0	15	29	44	47	0
coeff_abs_level_minus1	0	10	20	30	39	0

9.3.3.1.1 Processus d'allocation de ctxIdxInc au moyen des éléments syntaxiques du voisinage

Le paragraphe 9.3.3.1.1.1 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_skip_flag.

Le paragraphe 9.3.3.1.1.2 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_field_decoding_flag.

Le paragraphe 9.3.3.1.1.3 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_type.

Le paragraphe 9.3.3.1.1.4 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_pattern.

Le paragraphe 9.3.3.1.1.5 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_qp_delta.

Le paragraphe 9.3.3.1.1.6 spécifie le processus de déduction de ctxIdxInc pour les éléments syntaxiques ref_idx_l0 et ref_idx_l1.

Le paragraphe 9.3.3.1.1.7 spécifie le processus de déduction de ctxIdxInc pour les éléments syntaxiques mvd_l0 et mvd_l1.

Le paragraphe 9.3.3.1.1.8 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique intra_chroma_pred_mode.

Le paragraphe 9.3.3.1.1.9 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_flag.

Le paragraphe 9.3.3.1.1.10 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique transform_size_8x8_flag.

9.3.3.1.1.1 Processus de déduction de ctxIdxInc pour l'élément syntaxique mb_skip_flag

Le résultat de ce processus est ctxIdxInc.

Lorsque MbaffFrameFlag est égal à 1 et que mb_field_decoding_flag n'a pas été décodé (pas encore) pour la paire de macroblocs en cours avec l'adresse de macrobloc supérieur 2 * (CurrMbAddr/2), on applique la règle d'inférence pour l'élément syntaxique mb_field_decoding_flag, comme spécifié au § 7.4.4.

Le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.8.1 est invoqué et le résultat est alloué à mbAddrA et mbAddrB.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

– Si mbAddrN n'est pas disponible ou si mb_skip_flag pour le macrobloc mbAddrN est égal à 1, condTermFlagN est mis égal à 0.

– Sinon (si mbAddrN est disponible et mb_skip_flag pour le macrobloc mbAddrN est égal à 0), condTermFlagN est mis égal à 1.

La variable ctxIdxInc est calculée par

$$ctxIdxInc = condTermFlagA + condTermFlagB \tag{9-1}$$

9.3.3.1.1.2 Processus de déduction de ctxIdxInc pour l'élément syntaxique mb_field_decoding_flag

Le résultat de ce processus est ctxIdxInc.

On invoque le processus de déduction pour les adresses des macroblocs du voisinage et leur disponibilité dans les trames MBAFF, comme spécifié au § 6.4.7, et le résultat est alloué à mbAddrA et mbAddrB.

Lorsque les macroblocs mbAddrN et mbAddrN + 1 ont tous deux mb_type égal à P_Skip ou B_Skip, on applique la règle d'inférence pour l'élément syntaxique mb_field_decoding_flag, comme spécifié au § 7.4.4, pour le macrobloc mbAddrN.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

– Si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 0,
  – mbAddrN n'est pas disponible
  – le macrobloc mbAddrN est un macrobloc de trame.
– Sinon, condTermFlagN est mis égal à 1.

La variable ctxIdxInc est calculée par

ctxIdxInc = condTermFlagA + condTermFlagB (9-2)

9.3.3.1.1.3 Processus de déduction de ctxIdxInc pour l'élément syntaxique mb_type

L'entrée dans ce processus est ctxIdxOffset.

Le résultat de ce processus est ctxIdxInc.

Le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.8.1 est invoqué et le résultat est alloué à mbAddrA et mbAddrB.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

– Si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 0
  – mbAddrN n'est pas disponible
  – ctxIdxOffset est égal à 0 et mb_type pour le macrobloc mbAddrN est égal à SI
  – ctxIdxOffset est égal à 3 et mb_type pour le macrobloc mbAddrN est égal à I_NxN
  – ctxIdxOffset est égal à 27 et mb_type pour le macrobloc mbAddrN est égal à P_Skip, B_Skip ou B_Direct_16x16
– Sinon, condTermFlagN est mis égal à 1.

La variable ctxIdxInc est calculée par

ctxIdxInc = condTermFlagA + condTermFlagB (9-3)

9.3.3.1.1.4 Processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_pattern

Les entrées dans ce processus sont ctxIdxOffset et binIdx.

Le résultat de ce processus est ctxIdxInc.

En fonction de la valeur de la variable ctxIdxOffset, on applique ce qui suit.

– Si ctxIdxOffset est égal à 73, on applique ce qui suit
  – Le processus de déduction pour les blocs luma 8x8 du voisinage, spécifié au § 6.4.8.2, est invoqué avec luma8x8BlkIdx = binIdx en entrée et le résultat est alloué à mbAddrA, mbAddrB, luma8x8BlkIdxA, et luma8x8BlkIdxB.
  – Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.
    – Si l'une quelconque des conditions suivantes est vraie, condTermFlagN est mis égal à 0
      – mbAddrN n'est pas disponible
      – mb_type pour le macrobloc mbAddrN est égal à I_PCM
      – le macrobloc mbAddrN n'est pas le macrobloc actuel CurrMbAddr et le macrobloc mbAddrN ne contient pas mb_type égal à P_Skip ou B_Skip, et
        ((CodedBlockPatternLuma >> luma8x8BlkIdxN) & 1) n'est pas égal à 0 pour la valeur de CodedBlockPatternLuma pour le macrobloc mbAddrN

– le macrobloc mbAddrN est le macrobloc actuel CurrMbAddr et la valeur de segment déjà décodée b_k de coded_block_pattern avec k = luma8x8BlkIdxN n'est pas égale à 0.

– Sinon, condTermFlagN est mis égal à 1.

– La variable ctxIdxInc est calculée par

$$ctxIdxInc = condTermFlagA + 2 * condTermFlagB \quad (9\text{-}4)$$

– Sinon (si ctxIdxOffset est égal à 77), on applique ce qui suit.

– Le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.8.1 est invoqué et le résultat est alloué à mbAddrA et mbAddrB.

– Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

– Si mbAddrN est disponible et que mb_type pour le macrobloc mbAddrN est égal à I_PCM, condTermFlagN est mis égal à 1

– Sinon, si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 0

– mbAddrN n'est pas disponible ou le macrobloc mbAddrN contient mb_type égal à P_Skip ou B_Skip

– binIdx est égal à 0 et CodedBlockPatternChroma pour le macrobloc mbAddrN est égal à 0

– binIdx est égal à 1 et CodedBlockPatternChroma pour le macrobloc mbAddrN n'est pas égal à 2

– Sinon, condTermFlagN est mis égal à 1.

– La variable ctxIdxInc est calculée par

$$ctxIdxInc = condTermFlagA + 2 * condTermFlagB + ((binIdx == 1) ? 4 : 0) \quad (9\text{-}5)$$

NOTE – Lorsqu'un macrobloc utilise un mode de prédiction Intra_16x16, les valeurs de CodedBlockPatternLuma et CodedBlockPatternChroma pour le macrobloc sont calculées à partir de mb_type comme spécifié au Tableau 7-11.

9.3.3.1.1.5 Processus de déduction de ctxIdxInc pour l'élément syntaxique mb_qp_delta

Le résultat de ce processus est ctxIdxInc.

Soit prevMbAddr l'adresse de macrobloc du macrobloc qui précède le macrobloc en cours dans l'ordre de décodage. Lorsque le macrobloc en cours est le premier macrobloc d'une tranche, prevMbAddr est marqué comme indisponible.

Soit la variable ctxIdxInc calculée comme suit.

– Si l'une des conditions suivantes est vraie, ctxIdxInc est mis égal à 0

– prevMbAddr n'est pas disponible ou le macrobloc prevMbAddr contient mb_type égal à P_Skip ou B_Skip

– mb_type du macrobloc prevMbAddr est égal à I_PCM

– le macrobloc prevMbAddr n'est pas codé en mode de prédiction Intra_16x16 et CodedBlockPatternLuma et CodedBlockPatternChroma sont tous deux égaux à 0 pour le macrobloc prevMbAddr

– mb_qp_delta est égal à 0 pour le macrobloc prevMbAddr

– Sinon, ctxIdxInc est mis égal à 1.

9.3.3.1.1.6 Processus de déduction de ctxIdxInc pour les éléments syntaxiques ref_idx_l0 et ref_idx_l1

L'entrée dans ce processus est mbPartIdx.

Le résultat de ce processus est ctxIdxInc.

L'interprétation de ref_idx_lX et de Pred_LX, dans le cadre du présent paragraphe, est spécifiée comme suit:

– Si ce processus est invoqué pour la déduction de ref_idx_l0, ref_idx_lX est interprété comme ref_idx_l0 et Pred_LX est interprété comme Pred_L0.

– Sinon (si ce processus est invoqué pour la déduction de ref_idx_l1), ref_idx_lX est interprété comme ref_idx_l1 et Pred_LX est interprété comme Pred_L1.

Soit currSubMbType mis égal à sub_mb_type[mbPartIdx].

Le processus de déduction pour les subdivisions du voisinage spécifiées au § 6.4.8.5 est invoqué avec mbPartIdx, currSubMbType et subMbPartIdx = 0 en entrée et le résultat est alloué à mbAddrA\mbPartIdxA et mbAddrB\mbPartIdxB.

Avec ref_idx_lX[mbPartIdxN] (N étant A ou B) spécifiant l'élément syntaxique pour le macrobloc mbAddrN, soit la variable refIdxZeroFlagN calculée comme suit.

– Si MbaffFrameFlag est égal à 1, le macrobloc en cours est un macrobloc de trame, et le macrobloc mbAddrN est un macrobloc de sous-trame

$$refIdxZeroFlagN = ((ref\_idx\_lX[mbPartIdxN] > 1) ? 0 : 1) \quad (9\text{-}6)$$

– Sinon,

$$refIdxZeroFlagN = ((ref\_idx\_lX[mbPartIdxN] > 0) ? 0 : 1) \quad (9\text{-}7)$$

Soit la variable predModeEqualFlagN spécifiée comme suit.

– Si le macrobloc mbAddrN a le mb_type égal à P_8x8 ou B_8x8, on applique ce qui suit.
  – Si SubMbPredMode(sub_mb_type[mbPartIdxN]) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlagN est mis égal à 0, où sub_mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.
  – Sinon, predModeEqualFlagN est mis égal à 1.
– Sinon, on applique ce qui suit.
  – Si MbPartPredMode(mb_type, mbPartIdxN) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlagN est mis égal à 0, où mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.
  – Sinon, predModeEqualFlagN est mis égal à 1.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

– Si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 0
  – mbAddrN n'est pas disponible
  – le macrobloc mbAddrN a le mb_type égal à P_Skip ou B_Skip
  – le macrobloc mbAddrN est codé en mode d'intraprédiction
  – predModeEqualFlagN est égal à 0
  – refIdxZeroFlagN est égal à 1
– Sinon, condTermFlagN est mis égal à 1.

La variable ctxIdxInc est calculée par

$$ctxIdxInc = condTermFlagA + 2 * condTermFlagB \quad (9\text{-}8)$$

9.3.3.1.1.7 Processus de déduction de ctxIdxInc pour les éléments syntaxiques mvd_l0 et mvd_l1

Les entrées dans ce processus sont mbPartIdx, subMbPartIdx et ctxIdxOffset.

Le résultat de ce processus est ctxIdxInc.

L'interprétation de mvd_lX et Pred_LX dans le cadre du présent paragraphe est spécifiée comme suit.

– Si ce processus est invoqué pour la déduction de mvd_l0, mvd_lX est interprété comme mvd_l0 et Pred_LX est interprété comme Pred_L0.
– Sinon (si ce processus est invoqué pour la déduction de mvd_l1), mvd_lX est interprété comme mvd_l1 et Pred_LX est interprété comme Pred_L1.

Soit currSubMbType mis égal à sub_mb_type[mbPartIdx].

Le processus de déduction pour les subdivisions du voisinage, spécifié au § 6.4.8.5, est invoqué avec mbPartIdx, currSubMbType et subMbPartIdx en entrée et le résultat est alloué à mbAddrA\mbPartIdxA\ subMbPartIdxA et mbAddrB\mbPartIdxB\subMbPartIdxB.

Soit la variable compIdx calculée comme suit.

– Si ctxIdxOffset est égal à 40, compIdx est mis égal à 0.

– Sinon (si ctxIdxOffset est égal à 47), compIdx est mis égal à 1.

Soit la variable predModeEqualFlagN spécifiée comme suit.

– Si le macrobloc mbAddrN a le mb_type égal à P_8x8 ou B_8x8, on applique ce qui suit.

  – Si SubMbPredMode(sub_mb_type[mbPartIdxN]) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlagN est mis égal à 0, où sub_mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.

  – Sinon, predModeEqualFlag est mis égal à 1.

– Sinon, on applique ce qui suit.

  – Si MbPartPredMode(mb_type, mbPartIdxN) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlagN est mis égal à 0, où mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.

  – Sinon, predModeEqualFlagN est mis égal à 1.

Soit la variable absMvdCompN (N étant A ou B) calculée comme suit.

– Si l'une des conditions suivantes est vraie, absMvdCompN est mis égal à 0

  – mbAddrN n'est pas disponible

  – le macrobloc mbAddrN a le mb_type égal à P_Skip ou B_Skip

  – le macrobloc mbAddrN est codé en un mode d'intraprédiction

  – predModeEqualFlagN est égal à 0

– Sinon, on applique ce qui suit

  – Si compIdx est égal à 1, MbaffFrameFlag est égal à 1, le macrobloc en cours est un macrobloc de trame, et le macrobloc mbAddrN est un macrobloc de sous-trame

$$absMvdCompN = Abs(mvd\_lX[mbPartIdxN][subMbPartIdxN][compIdx]) * 2 \quad (9\text{-}9)$$

  – Sinon, si compIdx est égal à 1, MbaffFrameFlag est égal à 1, le macrobloc en cours est un macrobloc de sous-trame, et le macrobloc mbAddrN est un macrobloc de trame

$$absMvdCompN = Abs(mvd\_lX[mbPartIdxN][subMbPartIdxN][compIdx]) / 2 \quad (9\text{-}10)$$

  – Sinon,

$$absMvdCompN = Abs(mvd\_lX[mbPartIdxN][subMbPartIdxN][compIdx]) \quad (9\text{-}11)$$

La variable ctxIdxInc est déduite comme suit

– Si (absMvdCompA + absMvdCompB) est inférieur à 3, ctxIdxInc est mis égal à 0.

– Sinon, si (absMvdCompA + absMvdCompB) est supérieur à 32, ctxIdxInc est mis égal à 2.

– Sinon ((absMvdCompA + absMvdCompB) est dans la gamme de 3 à 32, inclus), ctxIdxInc est mis égal à 1.

9.3.3.1.1.8 Processus de déduction de ctxIdxInc pour l'élément syntaxique intra_chroma_pred_mode

Le résultat de ce processus est ctxIdxInc.

On invoque le processus de déduction pour les macroblocs du voisinage, spécifié au § 6.4.8.1, et le résultat est alloué à mbAddrA et mbAddrB.

Soit la variable condTermFlagN (avec N remplacé par A ou B) calculée comme suit.

– Si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 0

  – mbAddrN n'est pas disponible;

  – Le macrobloc mbAddrN est codé en mode d'interprédiction;

  – mb_type pour le macrobloc mbAddrN est égal à I_PCM;

  – intra_chroma_pred_mode pour le macrobloc mbAddrN est égal à 0.

– Sinon, condTermFlagN est mis égal à 1.

La variable ctxIdxInc est calculée par

$$ctxIdxInc = condTermFlagA + condTermFlagB \quad (9\text{-}12)$$

9.3.3.1.1.9 Processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_flag

L'entrée dans ce processus est ctxBlockCat et une entrée supplémentaire est spécifiée comme suit.

– Si ctxBlockCat est égal à 0, pas d'entrée supplémentaire.

– Sinon, si ctxBlockCat est égal à 1 ou 2, luma4x4BlkIdx.

– Sinon, si ctxBlockCat est égal à 3, l'indice de composante chromatique iCbCr.

– Sinon (si ctxBlockCat est égal à 4), chroma4x4BlkIdx et l'indice de composante chromatiqueiCbCr.

Le résultat de ce processus est ctxIdxInc(ctxBlockCat).

Soit la variable transBlockN (N étant A ou B) calculée comme suit.

– Si ctxBlockCat est égal à 0, on applique ce qui suit.

  – On invoque le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.8.1 et le résultat est alloué à mbAddrN (N étant A ou B).

  – La variable transBlockN est déduite comme suit.

    – Si mbAddrN est disponible et que le macrobloc mbAddrN est codé en mode de prédiction Intra_16x16, le bloc DC luma du macrobloc mbAddrN est alloué à transBlockN.

    – Sinon, transBlockN est marqué indisponible.

– Sinon, si ctxBlockCat est égal à 1 ou 2, on applique ce qui suit.

  – Le processus de déduction pour les blocs luma 4x4 du voisinage, spécifié au § 6.4.8.3, est invoqué avec luma4x4BlkIdx en entrée et le résultat est alloué à mbAddrN, luma4x4BlkIdxN (N étant A ou B).

  – La variable transBlockN est déduite comme suit.

    – Si mbAddrN est disponible, le macrobloc mbAddrN ne contient pas mb_type égal à P_Skip, B_Skip, ou I_PCM, si ((CodedBlockPatternLuma >> (luma4x4BlkIdxN >>2)) & 1) n'est pas égal à 0 pour le macrobloc mbAddrN, et si transform_size_8x8_flag est égal à 0 pour le macrobloc mbAddrN, le bloc luma 4x4 avec l'indice luma4x4BlkIdxN du macrobloc mbAddrN est alloué à transBlockN.

    – Sinon, si mbAddrN est disponible, si le macrobloc mbAddrN ne contient pas mb_type égal à P_Skip ou B_Skip, si ((CodedBlockPatternLuma >> (luma4x4BlkIdxN >>2)) & 1) n'est pas égal à 0 pour le macrobloc mbAddrN, et si transform_size_8x8_flag est égal à 1 pour le macrobloc mbAddrN, le bloc de 8x8 échantillons luma ayant l'indice (luma4x4BlkIdxN >> 2) de macrobloc mbAddrN est attribué à transBlockN.

    – Sinon, transBlockN est marqué indisponible.

– Sinon, si ctxBlockCat est égal à 3, on applique ce qui suit.

  – Le processus de déduction pour les macroblocs du voisinage, spécifié au § 6.4.8.1, est invoqué et le résultat est alloué à mbAddrN (N étant A ou B).

  – La variable transBlockN est déduite comme suit.

    – Si mbAddrN est disponible, le macrobloc mbAddrN ne contient pas mb_type égal à P_Skip, B_Skip ou I_PCM, et si CodedBlockPatternChroma n'est pas égal à 0 pour le macrobloc mbAddrN, le bloc DC chroma de la composante chromatique iCbCr du macrobloc mbAddrN est alloué à transBlockN.

    – Sinon, le bloc transBlockN est marqué indisponible.

– Sinon (si ctxBlockCat est égal à 4), on applique ce qui suit.

  – Le processus de déduction pour les blocs chroma 4x4 voisins, spécifié au § 6.4.8.4, est invoqué avec chroma4x4BlkIdx en entrée et le résultat est alloué à mbAddrN, chroma4x4BlkIdxN (N étant A ou B).

  – La variable transBlockN est déduite comme suit.

    – Si mbAddrN est disponible, le macrobloc mbAddrN ne contient pas mb_type égal à P_Skip, B_Skip ou I_PCM, et si CodedBlockPatternChroma est égal à 2 pour le macrobloc mbAddrN, le bloc chroma 4x4 avec chroma4x4BlkIdxN de la composante chromatique iCbCr du macrobloc mbAddrN est alloué à transBlockN.

    – Sinon, transBlockN est marqué comme indisponible.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

– Si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 0

  – mbAddrN n'est pas disponible et le macrobloc en cours est codé en mode d'interprédiction;

  – mbAddrN est disponible et transBlockN n'est pas disponible et mb_type pour le macrobloc mbAddrN n'est pas égal à I_PCM;

  – Le macrobloc en cours est codé en mode d'intraprédiction, constrained_intra_pred_flag est égal à 1, le macrobloc mbAddrN est disponible et est codé en mode d'interprédiction, et la subdivision des données de tranche est utilisée (nal_unit_type est dans la gamme de 2 à 4, inclus).

– Sinon, si l'une des conditions suivantes est vraie, condTermFlagN est mis égal à 1

  – mbAddrN n'est pas disponible et le macrobloc en cours est codé en mode d'intraprédiction;

  – mb_type pour le macrobloc mbAddrN est égal à I_PCM.

– Sinon, condTermFlagN est mis égal à la valeur du fanion coded_block_flag du bloc de transformée transBlockN qui a été décodé pour le macrobloc mbAddrN.

La variable ctxIdxInc(ctxBlockCat) est calculée par

$$ctxIdxInc(ctxBlockCat) = condTermFlagA + 2 * condTermFlagB \quad (9\text{-}13)$$

9.3.3.1.1.10 Processus de déduction de ctxIdxInc pour l'élément syntaxique transform_size_8x8_flag

La sortie de ce processus est ctxIdxInc.

Le processus de calcul pour les macroblocs voisins, spécifié dans le § 6.4.8.1, est invoqué et la sortie est attribuée à mbAddrA et mbAddrB.

Soit la variable condTermFlagN (N étant A ou B) à calculer comme suit:

– Si toutes les conditions suivantes sont vraies, condTermFlagN est mis à 0.

  – mbAddrN est indisponible

  – transform_size_8x8_flag pour le macrobloc mbAddrN est égal à 0

– Sinon, condTermFlagN est mis à 1.

La variable ctxIdxInc est calculée par:

$$ctxIdxInc = condTermFlagA + condTermFlagB \quad (9\text{-}14)$$

9.3.3.1.2 Processus d'allocation de ctxIdxInc utilisant les valeurs de segments précédemment décodés

Les entrées dans ce processus sont ctxIdxOffset et binIdx.

Le résultat de ce processus est ctxIdxInc.

Le Tableau 9-32 contient la spécification de ctxIdxInc pour les valeurs données de ctxIdxOffset et binIdx.

Pour chaque valeur de ctxIdxOffset et binIdx, ctxIdxInc est calculé au moyen de certaines des valeurs des segments précédemment décodés (b_0, b_1, b_2,…, b_k), où la valeur de l'indice k est inférieure à la valeur de binIdx.

Tableau 9-32 – Spécification de ctxIdxInc pour des valeurs spécifiques de ctxIdxOffset et binIdx

Valeur (nom) de ctxIdxOffset	binIdx	ctxIdxInc
3	4	(b_3 != 0) ? 5: 6
	5	(b_3 != 0) ? 6: 7
14	2	(b_1 != 1) ? 2: 3
17	4	(b_3 != 0) ? 2: 3
27	2	(b_1 != 0) ? 4: 5
32	4	(b_3 != 0) ? 2: 3
36	2	(b_1 != 0) ? 2: 3

9.3.3.1.3 Processus d'allocation de ctxIdxInc pour les éléments syntaxiques significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1

Les entrées dans ce processus sont ctxIdxOffset et binIdx.

Le résultat de ce processus est ctxIdxInc.

Le processus d'allocation de ctxIdxInc pour les éléments syntaxiques significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1 ainsi que pour coded_block_flag dépend des catégories des différents blocs mentionnés par la variable ctxBlockCat. La spécification des catégories de ces blocs est donnée au Tableau 9-33.

Tableau 9-33 – Spécification de ctxBlockCat pour les différents blocs

Description du bloc	maxNumCoeff	ctxBlockCat
Bloc de niveaux de coefficient de transformée d'échantillon apériodique luma (c'est-à-dire liste Intra16x16DCLevel comme décrit dans le § 7.4.5.3)	16	0
Bloc de niveaux de coefficient de transformée d'échantillons périodiques luma (c'est-à-dire liste Intra16x16ACLevel[i] comme décrit dans le § 7.4.5.3)	15	1
Bloc de 16 niveaux de coefficient de transformée d'échantillons luma (c'est-à-dire liste LumaLevel[i] comme décrit dans le § 7.4.5.3)	16	2
Bloc de niveaux de coefficient de transformée d'échantillon apériodique chroma	4 * NumC8x8	3
Bloc de niveaux de coefficient de transformée d'échantillons périodiques chroma	15	4
Bloc de 64 niveaux de coefficient de transformée d'échantillons luma (c'est-à-dire liste LumaLevel8x8[i] comme décrit au § 7.4.5.3)	64	5

Soit la variable levelListIdx mise égale à l'indice de la liste des niveaux de coefficient de transformée comme spécifié au § 7.4.5.3.

Pour les éléments syntaxiques significant_coeff_flag et last_significant_coeff_flag, contenus dans les blocs avec ctxBlockCat < 5 et ctxBlockCat != 3, la variable ctxIdxInc est calculée par:

$$ctxIdxInc = levelListIdx \quad (9\text{-}15)$$

où l'étendue de levelListIdx va de 0 à maxNumCoeff – 2, inclus.

Pour les éléments syntaxiques significant_coeff_flag et last_significant_coeff_flag dans les blocs avec ctxBlockCat = = 3, la variable ctxIdxInc est calculée par:

$$ctxIdxInc = Min(levelListIdx / NumC8x8, 2) \quad (9\text{-}16)$$

où l'étendue de levelListIdx va de 0 à 4 * NumC8x8 – 2, inclus.

Pour les éléments syntaxiques significant_coeff_flag et last_significant_coeff_flag dans les blocs luma 8x8 avec ctxBlockCat = = 5, le Tableau 9-34 contient la spécification de ctxIdxInc pour les valeurs données de levelListIdx, où l'étendue de levelListIdx va de 0 à 62, inclus.

Tableau 9-34 – Mappage de la position de balayage sur ctxIdxInc pour ctxBlockCat = = 5

levelListIdx	ctxIdxInc pour significant_coeff_flag (macroblocs codés en trames)	ctxIdxInc pour significant_coeff_flag (macroblocs codés en sous-trames)	ctxIdxInc pour last_significant_coeff_flag	levelListIdx	ctxIdxInc pour significant_coeff_flag (macroblocs codés en trames)	ctxIdxInc pour significant_coeff_flag (macroblocs codés en sous-trames)	ctxIdxInc pour last_significant_coeff_flag
0	0	0	0	**32**	7	9	3
1	1	1	1	**33**	6	9	3
2	2	1	1	**34**	11	10	3
3	3	2	1	**35**	12	10	3
4	4	2	1	**36**	13	8	3
5	5	3	1	**37**	11	11	3
6	5	3	1	**38**	6	12	3
7	4	4	1	**39**	7	11	3
8	4	5	1	**40**	8	9	4
9	3	6	1	**41**	9	9	4
10	3	7	1	**42**	14	10	4
11	4	7	1	**43**	10	10	4
12	4	7	1	**44**	9	8	4
13	4	8	1	**45**	8	13	4
14	5	4	1	**46**	6	13	4
15	5	5	1	**47**	11	9	4
16	4	6	2	**48**	12	9	5
17	4	9	2	**49**	13	10	5
18	4	10	2	**50**	11	10	5
19	4	10	2	**51**	6	8	5
20	3	8	2	**52**	9	13	6
21	3	11	2	**53**	14	13	6
22	6	12	2	**54**	10	9	6
23	7	11	2	**55**	9	9	6
24	7	9	2	**56**	11	10	7
25	7	9	2	**57**	12	10	7
26	8	10	2	**58**	13	14	7
27	9	10	2	**59**	11	14	7
28	10	8	2	**60**	14	14	8
29	9	11	2	**61**	10	14	8
30	8	12	2	**62**	12	14	8
31	7	11	2				

Soit numDecodAbsLevelEq1 la notation du nombre cumulé de niveaux décodés de coefficient de transformée avec une valeur absolue égale à 1, et soit numDecodAbsLevelGt1 la notation du nombre cumulé de niveaux décodés de coefficient de transformée avec une valeur absolue supérieure à 1. Les deux nombres se rapportent au même bloc de coefficient de transformée, dans lequel s'effectue le processus de décodage en cours. Puis, pour le décodage de coeff_abs_level_minus1, ctxIdxInc pour coeff_abs_level_minus1 est spécifié en fonction de binIdx comme suit:

– Si binIdx est égal à 0, ctxIdxInc est calculé par

ctxIdxInc = ((numDecodAbsLevelGt1 != 0) ? 0: Min(4, 1 + numDecodAbsLevelEq1)) (9-17)

– Sinon (si binIdx est supérieur à 0), ctxIdxInc est calculé par

$$ctxIdxInc = 5 + Min(4 – (ctxBlockCat == 3), numDecodAbsLevelGt1) \quad (9\text{-}18)$$

9.3.3.2 Processus de décodage arithmétique

Les entrées dans ce processus sont le fanion bypassFlag, l'indice ctxIdx comme calculé au § 9.3.3.1, et les variables d'état codIRange et codIOffset du moteur de décodage arithmétique.

Le résultat de ce processus est la valeur du segment.

La Figure 9-2 illustre l'ensemble du processus de décodage arithmétique pour un seul segment. Afin de décoder la valeur d'un segment, l'indice de contexte ctxIdx est passé au processus de décodage arithmétique DecodeBin(ctxIdx), qui est spécifié comme suit.

– Si bypassFlag est égal à 1, DecodeBypass( ) comme spécifié au § 9.3.3.2.3 est invoqué.

– Sinon, si bypassFlag est égal à 0 et que ctxIdx est égal à 276, DecodeTerminate( ) comme spécifié au § 9.3.3.2.4 est invoqué.

– Sinon (si bypassFlag est égal à 0 et si ctxIdx n'est pas égal à 276), DecodeDecision( ) comme spécifié au § 9.3.3.2.1 est appliqué.

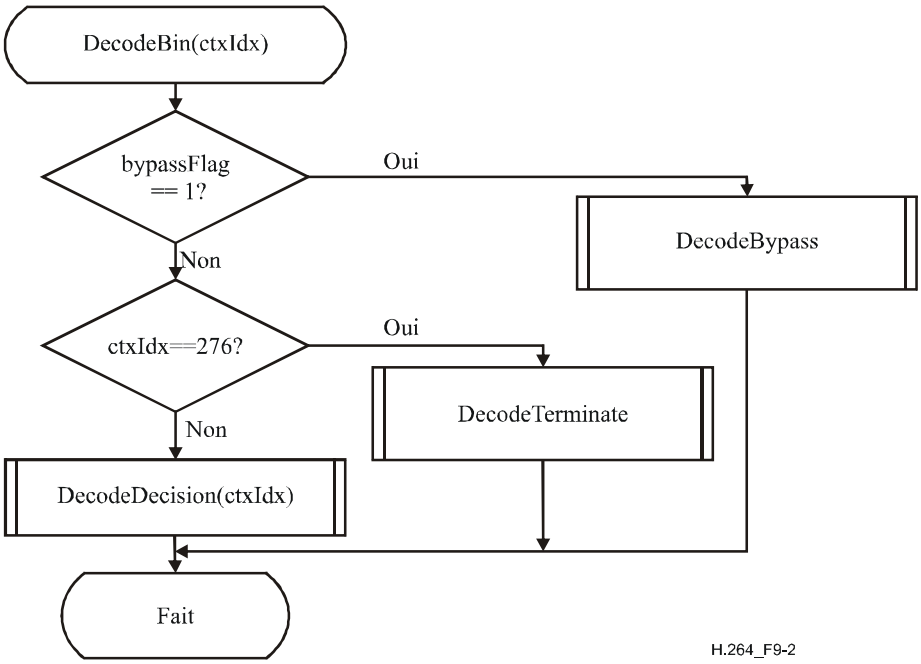


Figure 9-2 – Aperçu général du processus de décodage arithmétique pour un seul segment (pour information)

NOTE – Le codage arithmétique se fonde sur le principe d'une subdivision à intervalle récursif. Etant donné une estimation de probabilité p(0) et p(1) = 1 – p(0) d'une décision binaire (0, 1), un sous-intervalle de code donné initialement avec la gamme codIRange sera subdivisé en deux sous-intervalles ayant respectivement la gamme p(0) * codIRange et codIRange – p(0) * codIRange. En fonction de la décision, qui a été observée, le sous-intervalle correspondant sera choisi comme le nouvel intervalle de code, et une chaîne de code binaire pointant dans cet intervalle représentera la séquence de décisions binaires observées. Il est utile de distinguer entre le symbole le plus probable (MPS, *most probable symbol*) et le symbole le moins probable (LPS, *least probable symbol*), de sorte que les décisions binaires doivent être identifiées comme MPS ou LPS, plutôt que comme 0 ou 1. Etant donné cette terminologie, chaque contexte est spécifié par la probabilité p_{LPS} du LPS et la valeur de MPS (valMPS), qui est 0 ou 1.

Le moteur de codage arithmétique de la présente Recommandation | Norme internationale a trois propriétés distinctes:

– l'estimation de probabilité est effectuée au moyen d'une machine à états finis avec un processus de transition fondé sur un tableau entre 64 états de probabilité représentatifs différents { p_{LPS}(pStateIdx) | 0 <= pStateIdx < 64 } pour la probabilité LPS p_{LPS}. La numérotation des états est arrangée

de telle sorte que l'état de probabilité avec l'indice pStateIdx = 0 corresponde à une valeur de probabilité LPS de 0,5, avec une probabilité LPS décroissante vers les indices d'état les plus élevés;

– la gamme de codIRange représentant l'état du moteur de codification est quantifiée par un petit ensemble $\{Q_1,\ldots,Q_4\}$ de valeurs de quantification préétablies avant le calcul de la nouvelle gamme d'intervalles. L'enregistrement d'un tableau contenant toutes les valeurs de produit précalculées de 64x4 de $Q_i * p_{LPS}(pStateIdx)$ permet une approximation sans multiplication du produit codIRange * $p_{LPS}(pStateIdx)$;

– pour des éléments syntaxiques, ou leurs parties, pour lesquels une distribution de probabilité approximativement uniforme est supposée être donnée, on utilise un processus de dérivation séparé et simplifié de codage et décodage.

9.3.3.2.1 Processus de décodage arithmétique pour une décision binaire

Les entrées dans ce processus sont ctxIdx, codIRange, et codIOffset.

Les résultats de ce processus sont la valeur décodée binVal, et les variables mises à jour codIRange et codIOffset.

La Figure 9-3 indique le diagramme de décodage d'une décision unique (DecodeDecision).

1. La valeur de la variable codIRangeLPS est déduite comme suit.

– Etant donné la valeur en cours de codIRange, la variable qCodIRangeIdx est calculée par

$$qCodIRangeIdx = (codIRange >> 6) \& 0x03 \quad (9\text{-}19)$$

– Etant donné qCodIRangeIdx et pStateIdx associés à ctxIdx, la valeur de la variable rangeTabLPS comme spécifié au Tableau 9-35 est allouée à codIRangeLPS:

$$codIRangeLPS = rangeTabLPS[pStateIdx][qCodIRangeIdx] \quad (9\text{-}20)$$

2. La variable codIRange est mise égale à codIRange – codIRangeLPS et on applique ce qui suit.

– Si codIOffset est supérieur ou égal à codIRange, la variable binVal est mise égale à 1 – valMPS, codIOffset est décrémenté de codIRange, et codIRange est mis égal à codIRangeLPS.

– Sinon, la variable binVal est mise égale à valMPS.

Etant donné la valeur de binVal, la transition d'état est effectuée comme spécifié au § 9.3.3.2.1.1. En fonction de la valeur en cours de codIRange, la renormalisation est effectuée comme spécifié au § 9.3.3.2.2.

9.3.3.2.1.1 Processus de transition d'état

Les entrées dans ce processus sont l'indice pStateIdx en cours, la valeur décodée binVal et les valeurs valMPS de la variable de contexte associée à ctxIdx.

Les résultats de ce processus sont les indices pStateIdx et valMPS mis à jour de la variable de contexte associée à ctxIdx.

En fonction de la valeur décodée binVal, la mise à jour des deux variables pStateIdx et valMPS associées à ctxIdx est déduite comme suit:

```
si( binVal  = =  valMPS )
    pStateIdx = transIdxMPS( pStateIdx )
ou {                                                    (9-21)
    si( pStateIdx  = =  0 )
        valMPS = 1 – valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
```

Le Tableau 9-36 spécifie les règles de transition transIdxMPS() et transIdxLPS() après le décodage de la valeur de valMPS et 1 – valMPS, respectivement.

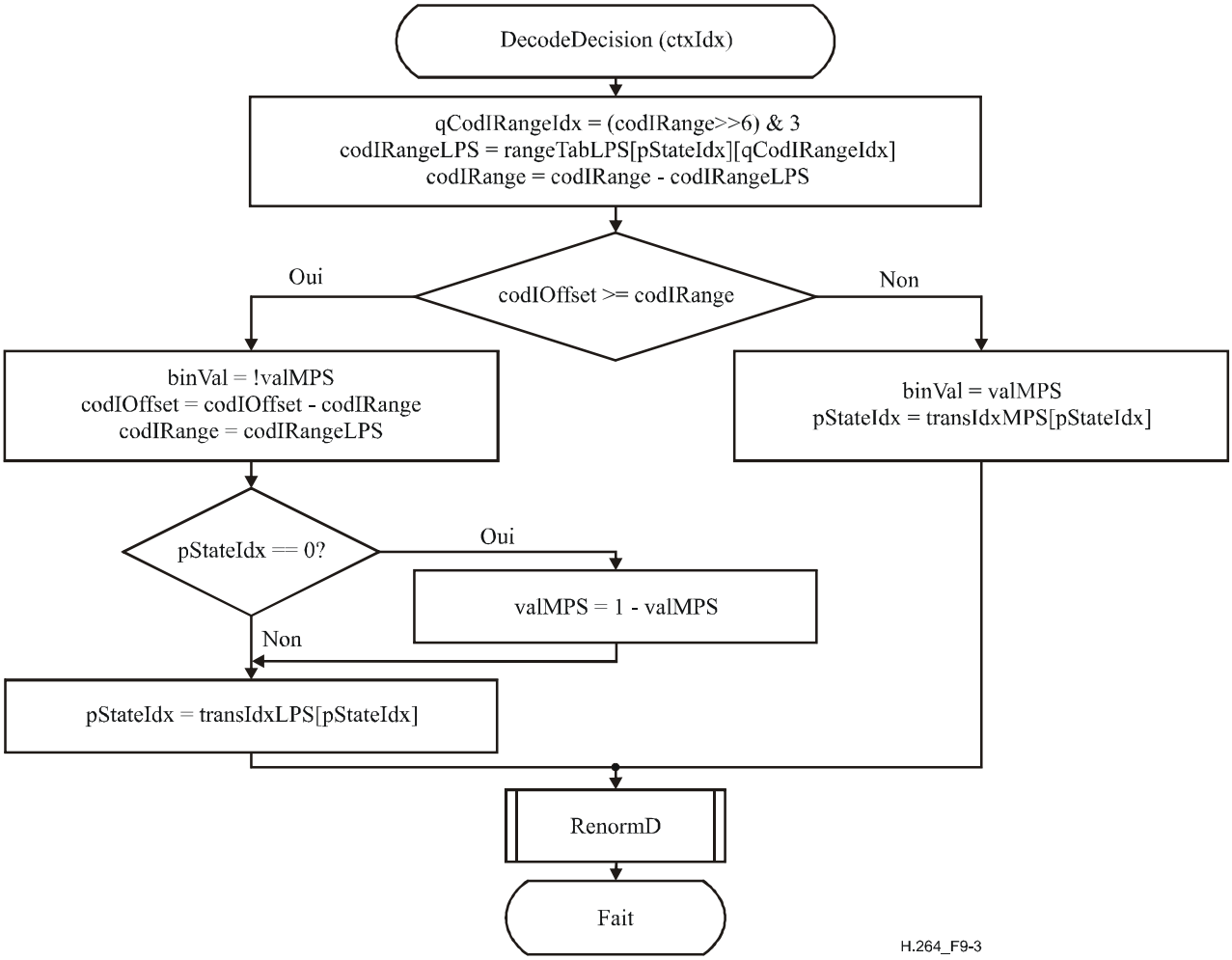


Figure 9-3 – Diagramme pour le décodage d'une décision

Tableau 9-35 – Spécification de rangeTabLPS en fonction de pStateIdx et qCodIRangeIdx

pStateIdx	qCodIRangeIdx				pStateIdx	qCodIRangeIdx			
	0	**1**	**2**	**3**		**0**	**1**	**2**	**3**
0	128	176	208	240	**32**	27	33	39	45
1	128	167	197	227	**33**	26	31	37	43
2	128	158	187	216	**34**	24	30	35	41
3	123	150	178	205	**35**	23	28	33	39
4	116	142	169	195	**36**	22	27	32	37
5	111	135	160	185	**37**	21	26	30	35
6	105	128	152	175	**38**	20	24	29	33
7	100	122	144	166	**39**	19	23	27	31
8	95	116	137	158	**40**	18	22	26	30
9	90	110	130	150	**41**	17	21	25	28
10	85	104	123	142	**42**	16	20	23	27
11	81	99	117	135	**43**	15	19	22	25
12	77	94	111	128	**44**	14	18	21	24
13	73	89	105	122	**45**	14	17	20	23
14	69	85	100	116	**46**	13	16	19	22
15	66	80	95	110	**47**	12	15	18	21
16	62	76	90	104	**48**	12	14	17	20
17	59	72	86	99	**49**	11	14	16	19
18	56	69	81	94	**50**	11	13	15	18
19	53	65	77	89	**51**	10	12	15	17
20	51	62	73	85	**52**	10	12	14	16
21	48	59	69	80	**53**	9	11	13	15
22	46	56	66	76	**54**	9	11	12	14
23	43	53	63	72	**55**	8	10	12	14
24	41	50	59	69	**56**	8	9	11	13
25	39	48	56	65	**57**	7	9	11	12
26	37	45	54	62	**58**	7	9	10	12
27	35	43	51	59	**59**	7	8	10	11
28	33	41	48	56	**60**	6	8	9	11
29	32	39	46	53	**61**	6	7	9	10
30	30	37	43	50	**62**	6	7	8	9
31	29	35	41	48	**63**	2	2	2	2

Tableau 9-36 – Tableau de transition d'état

pStateIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
transIdxLPS	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
transIdxLPS	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMPS	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
transIdxLPS	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMPS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
transIdxLPS	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMPS	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.3.2.2 Processus de renormalisation dans le moteur de décodage arithmétique

Les entrées dans ce processus sont des bits provenant de données de tranche et les variables codIRange et codIOffset.

Les résultats de ce processus sont les variables mises à jour codIRange et codIOffset.

Un diagramme de la renormalisation est donné à la Figure 9-4. La valeur en cours de codIRange est d'abord comparée à 0x0100 et les étapes suivantes sont spécifiées comme suit.

– Si codIRange est supérieur ou égal à 0x0100, aucune renormalisation n'est nécessaire et le processus RenormD est terminé.

– Sinon (si codIRange est inférieur à 0x0100), on entre dans la boucle de renormalisation. Dans cette boucle, la valeur de codIRange est doublée, c'est-à-dire décalée à gauche de 1 et un seul bit est déplacé dans codIOffset au moyen de read_bits(1).

Le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de codIOffset supérieure ou égale à codIRange à l'achèvement de ce processus.

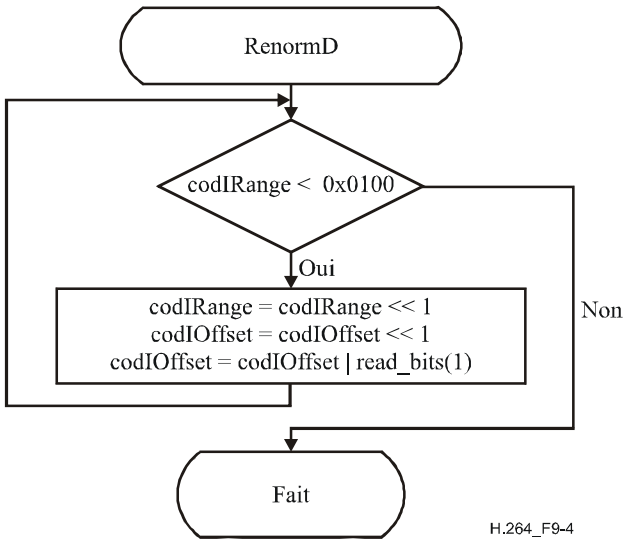


Figure 9-4 – Diagramme de renormalisation

9.3.3.2.3 Processus de décodage d'aiguillage pour décisions binaires

Les entrées dans ce processus sont des bits provenant de données de tranche et les variables codIRange et codIOffset.

Les résultats de ce processus sont la variable mise à jour codIOffset et la valeur décodée binVal.

Le processus de décodage d'aiguillage est invoqué lorsque bypassFlag est égal à 1. La Figure 9-5 donne un diagramme du processus correspondant.

D'abord, la valeur de codIOffset est doublée, c'est-à-dire, décalée à gauche de 1 et un seul bit est déplacé dans codIOffset au moyen de read_bits(1). Puis, la valeur de codIOffset est comparée à la valeur de codIRange et les étapes suivantes sont spécifiées comme suit.

– Si codIOffset est supérieur ou égal à codIRange, la variable binVal est mise égale à 1 et codIOffset est décrémentée de codIRange.

– Sinon (si codIOffset est inférieur à codIRange), la variable binVal est mise égale à 0.

Le flux binaire ne doit pas contenir de données qui se traduiraient par une valeur de codIOffset supérieure ou égale à codIRange à l'achèvement de ce processus.

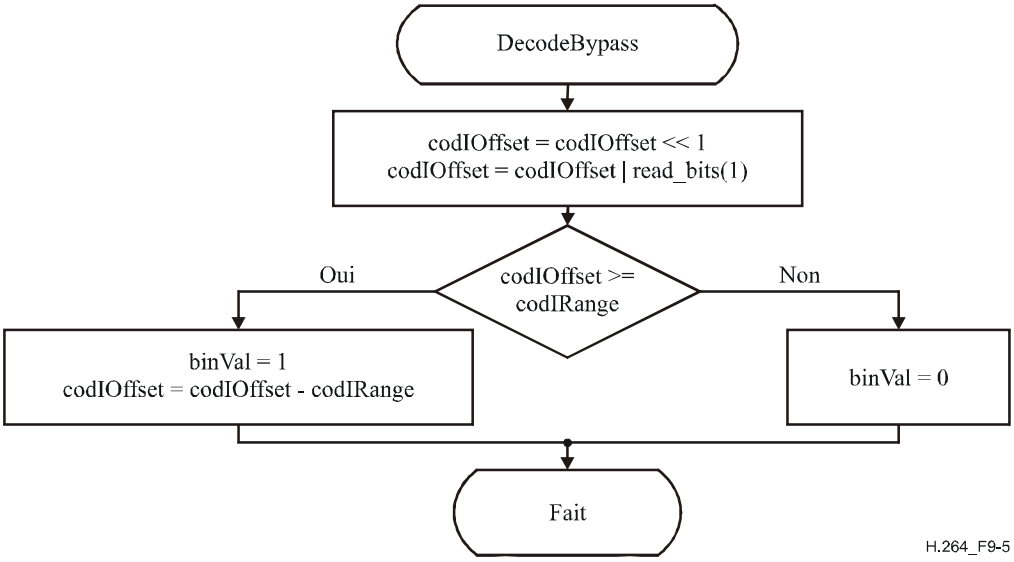


Figure 9-5 – Diagramme du processus de décodage d'aiguillage

9.3.3.2.4 Processus de décodage pour décisions binaires avant achèvement

Les entrées dans ce processus sont des bits provenant de données de tranche et les variables codIRange et codIOffset.

Les résultats de ce processus sont les variables mises à jour codIRange et codIOffset, et la valeur décodée binVal.

Ce programme spécial de décodage s'applique au décodage de end_of_slice_flag (*fanion de fin de tranche*) et du segment indiquant le mode I_PCM correspondant à ctxIdx égal à 276. La Figure 9-6 donne le diagramme du processus de décodage correspondant, qui est spécifié comme suit.

D'abord, la valeur de codIRange est décrémentée de 2. Puis, la valeur de codIOffset est comparée à la valeur de codIRange et les étapes suivantes sont spécifiées comme suit.

– Si codIOffset est supérieur ou égal à codIRange, la variable binVal est mise égale à 1, aucune renormalisation n'est effectuée, et le décodage CABAC s'achève. Le dernier bit inséré dans le registre codIOffset est égal à 1. Lors du décodage de end_of_slice_flag, ce dernier bit inséré dans le registre codIOffset est interprété comme rbsp_stop_one_bit.

– Sinon (si codIOffset est inférieur à codIRange), la variable binVal est mise égale à 0 et la renormalisation est effectuée comme spécifié au § 9.3.3.2.2.

NOTE – Cette procédure peut aussi être mise en œuvre au moyen de DecodeDecision(ctxIdx) avec ctxIdx = 276. Dans le cas où la valeur décodée est égale à 1, sept bits de plus seraient lus par DecodeDecision(ctxIdx) et un processus de décodage aurait à ajuster en conséquence son pointeur de flux binaire afin de décoder correctement les éléments syntaxiques suivants.

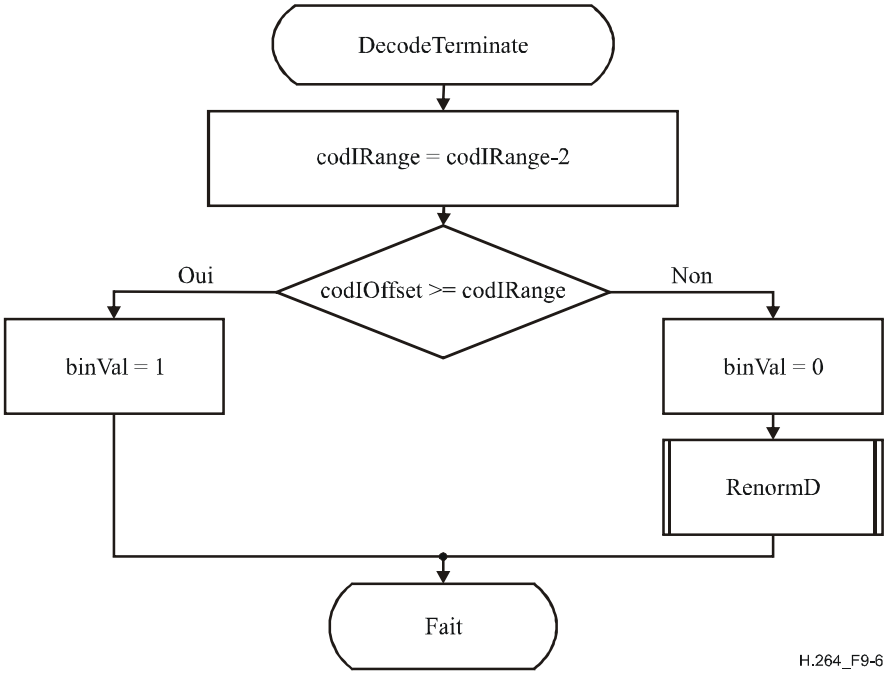


Figure 9-6 – Diagramme de décodage d'une décision avant achèvement

9.3.4 Processus de codage arithmétique (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Les entrées dans ce processus sont les décisions qui sont à coder et à écrire.

Les résultats de ce processus sont les bits qui sont écrits à la charge utile RBSP.

Ce paragraphe pour information décrit un moteur de codage arithmétique qui correspond au moteur de décodage arithmétique décrit au § 9.3.3.2. Le moteur de codage est essentiellement symétrique du moteur de décodage, c'est-à-dire que les procédures sont appelées dans le même ordre. Les procédures suivantes sont décrites dans la présente section: InitEncoder, EncodeDecision, EncodeBypass, EncodeTerminate, qui correspondent respectivement à InitDécodeur, DecodeDecision, DecodeBypass, et DecodeTerminate. L'état du moteur arithmétique de codage est représenté par une valeur de la variable codILow pointant sur l'extrémité inférieure d'un sous-intervalle et une valeur de la variable codIRange spécifiant la gamme correspondante de ce sous-intervalle.

9.3.4.1 Processus d'initialisation pour le moteur arithmétique de codage (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Ce processus est invoqué avant le codage du premier macrobloc d'une tranche, et après le codage de toute donnée pcm_alignment_zero_bit et de toutes les données pcm_sample_luma et pcm_sample_chroma pour un macrobloc de type I_PCM.

Les résultats de ce processus sont les valeurs codILow, codIRange, firstBitFlag, bitsOutstanding, et symCnt du moteur de codage arithmétique.

Dans la procédure d'initialisation de l'encodeur, codILow est mis égal à 0, et codIRange est mis égal à 0x01FE. De plus, un firstBitFlag est mis égal à 1, et les compteurs bitsOutstanding et symCnt sont mis égaux à 0.

NOTE – La précision minimale de registre requise pour codILow est de 10 bit et pour CodIRange elle est de 9 bit. La précision requise pour les compteurs bitsOutstanding et symCnt devrait être suffisement grande afin d'empêcher le débordement des registres en question. Lorsque MaxBinCountInSlice note le nombre total maximal de décisions binaires à coder en une tranche, la précision minimale de registre requise pour les variables bitsOutstanding et symCnt est donnée par Ceil(Log2(MaxBinCountInSlice + 1)).

9.3.4.2 Processus de codage pour une décision binaire (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Les entrées dans ce processus sont l'indice de contexte ctxIdx, la valeur de binVal à coder, et les variables codIRange, codILow et symCnt.

Les résultats de ce processus sont les variables codIRange, codILow, et symCnt.

La Figure 9-7 donne le diagramme pour le codage d'une seule décision. Dans une première étape, la variable codIRangeLPS est déduite comme suit.

Etant donné la valeur en cours de codIRange, codIRange est mappé avec l'indice qCodIRangeIdx d'une valeur quantifiée de codIRange au moyen de l'équation 9-19. La valeur de qCodIRangeIdx et la valeur de pStateIdx associées à ctxIdx sont utilisées afin de déterminer la valeur de la variable rangeTabLPS comme spécifié au Tableau 9-35, qui est alloué à codIRangeLPS. La valeur de codIRange – codIRangeLPS est allouée à codIRange.

Dans une seconde étape, la valeur de binVal est comparée à valMPS associé à ctxIdx. Lorsque binVal est différent de valMPS, codIRange est ajouté à codILow et codIRange est mis égal à la valeur codIRangeLPS. Etant donné la décision codée, la transition d'état est effectuée comme spécifié au § 9.3.3.2.1.1. En fonction de la valeur en cours de codIRange, la renormalisation est effectuée comme spécifié au § 9.3.4.3. Finalement, la variable symCnt est incrémentée de 1.

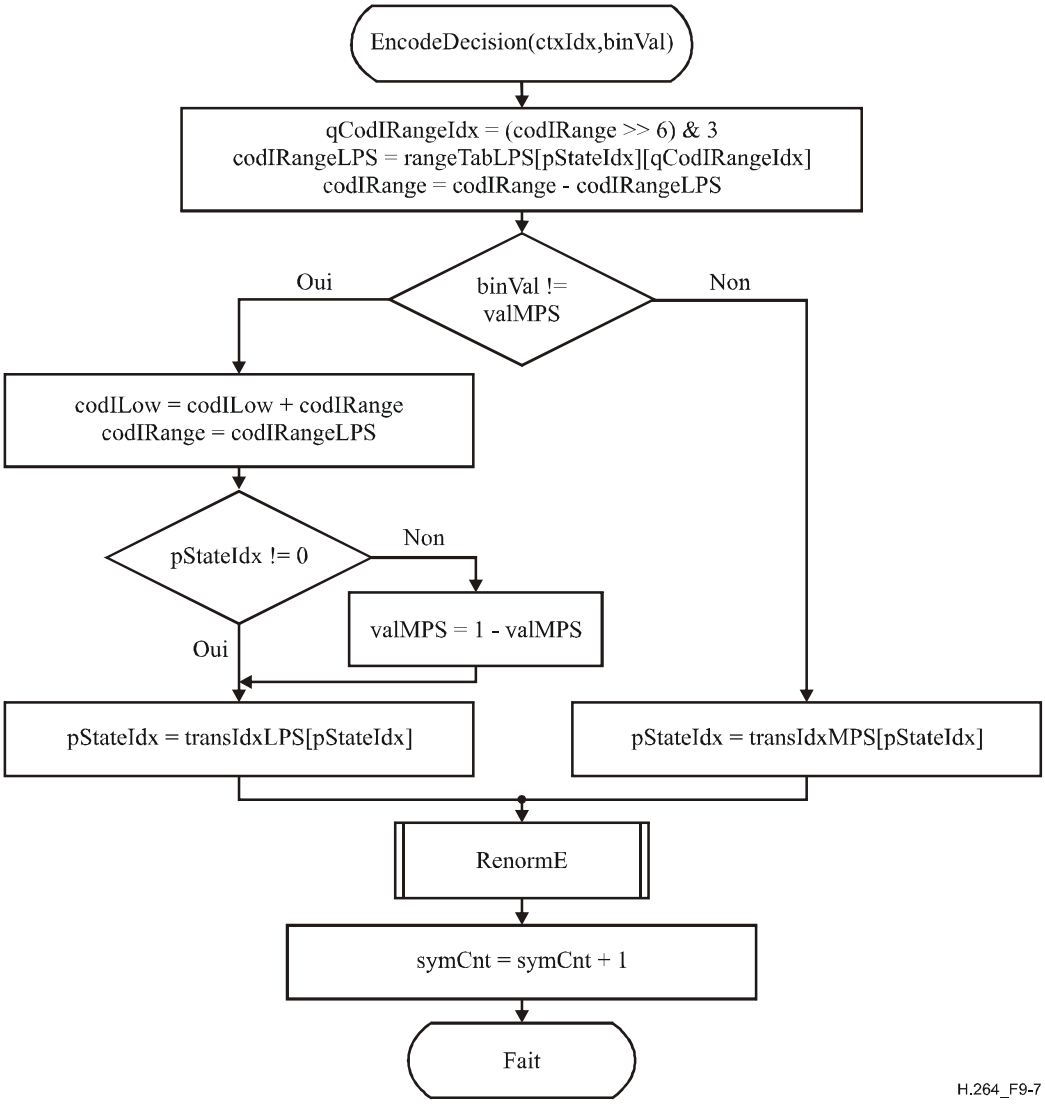


Figure 9-7 – Diagramme pour le codage d'une décision

9.3.4.3 Processus de renormalisation dans le moteur de codage arithmétique (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Les entrées dans ce processus sont les variables codIRange, codILow, firstBitFlag, et bitsOutstanding.

Les résultats de ce processus sont zéro ou plusieurs bits écrits dans la charge utile RBSP et les variables mises à jour codIRange, codILow, firstBitFlag, et bitsOutstanding.

La renormalisation est illustrée à la Figure 9-8.

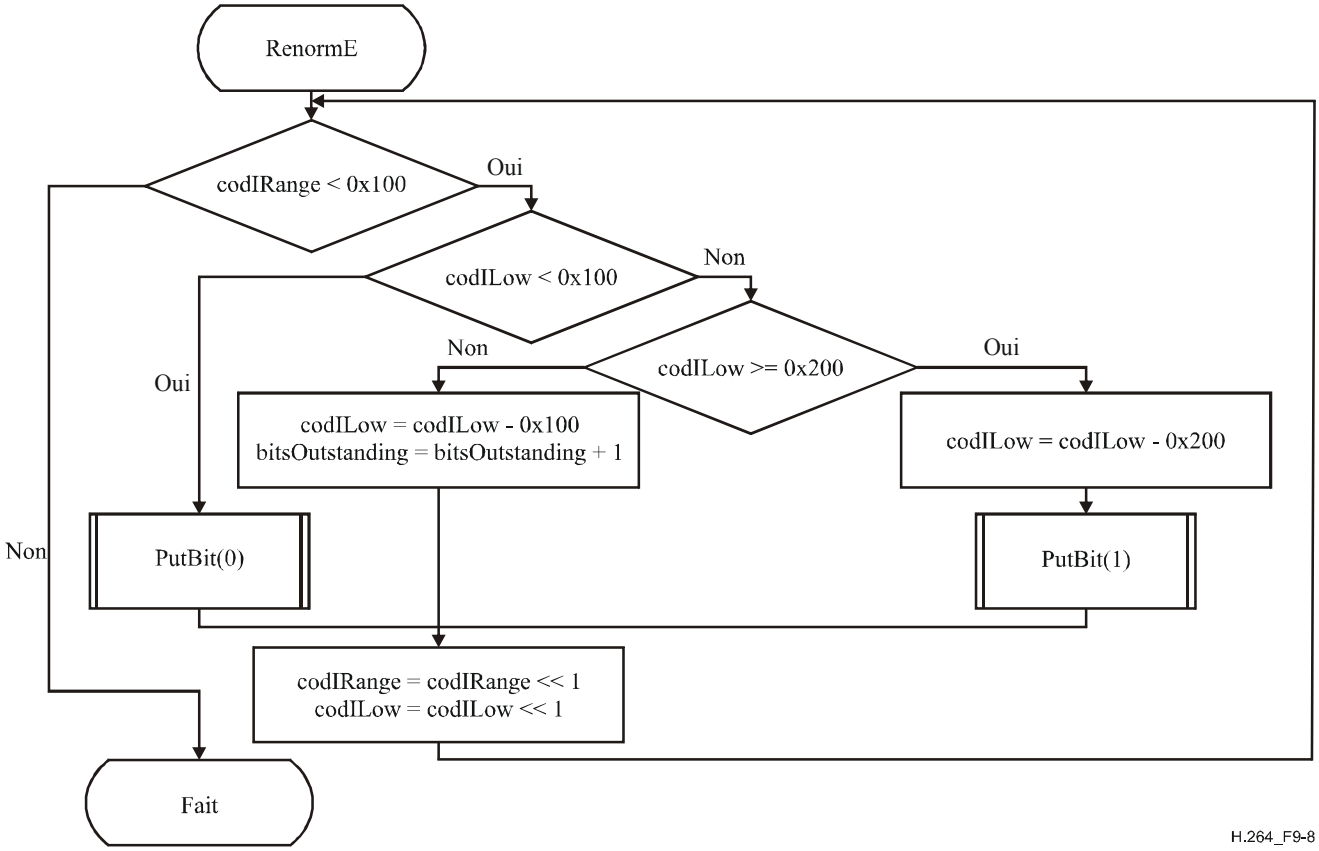


Figure 9-8 – Diagramme de la renormalisation dans le codeur

La procédure PutBit( ) décrite à la Figure 9-9 donne la commande du report. Elle utilise la fonction WriteBits(B, N) qui écrit N bits avec la valeur B au flux binaire et avance le pointeur du flux binaire de N positions de bit. Cette fonction suppose l'existence d'un pointeur de flux binaire avec une indication de la position du prochain bit que le processus de codage doit écrire au flux binaire.

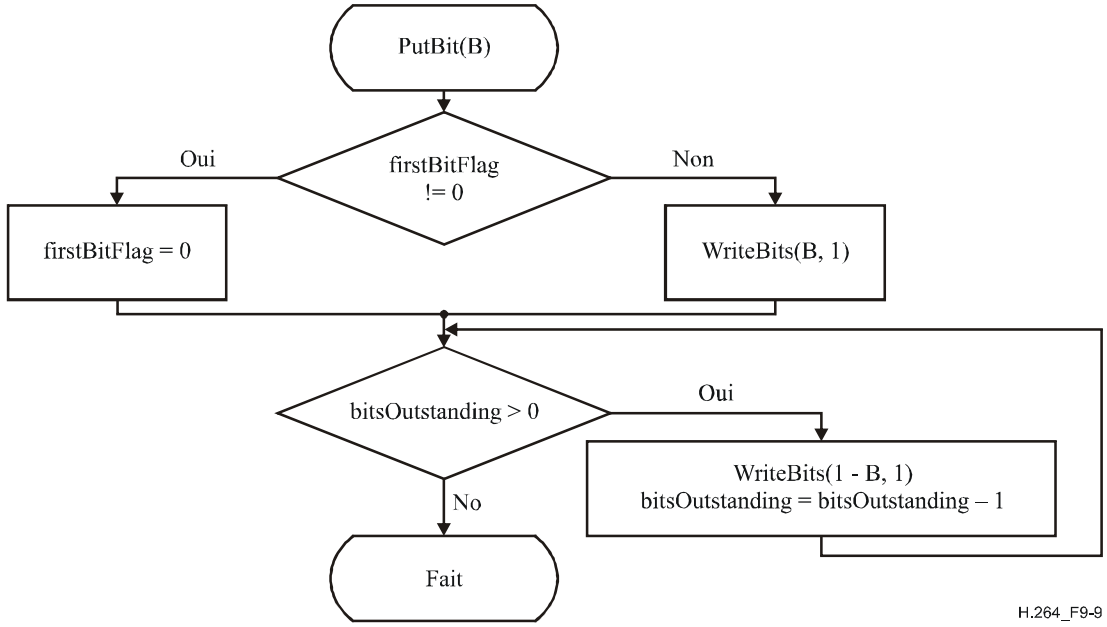


Figure 9-9 – Diagramme de PutBit(B)

9.3.4.4 Processus de codage d'aiguillage pour décisions binaires (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Les entrées dans ce processus sont les variables binVal, codILow, codIRange, bitsOutstanding, et symCnt.

Le résultat de ce processus est un bit écrit à la charge utile RBSP et les variables mises à jour codILow, codIRange, bitsOutstanding, et symCnt.

Ce processus de codage s'applique à toutes les décisions binaires avec bypassFlag égal à 1. La renormalisation est incluse dans la spécification de ce processus comme indiqué à la Figure 9-10.

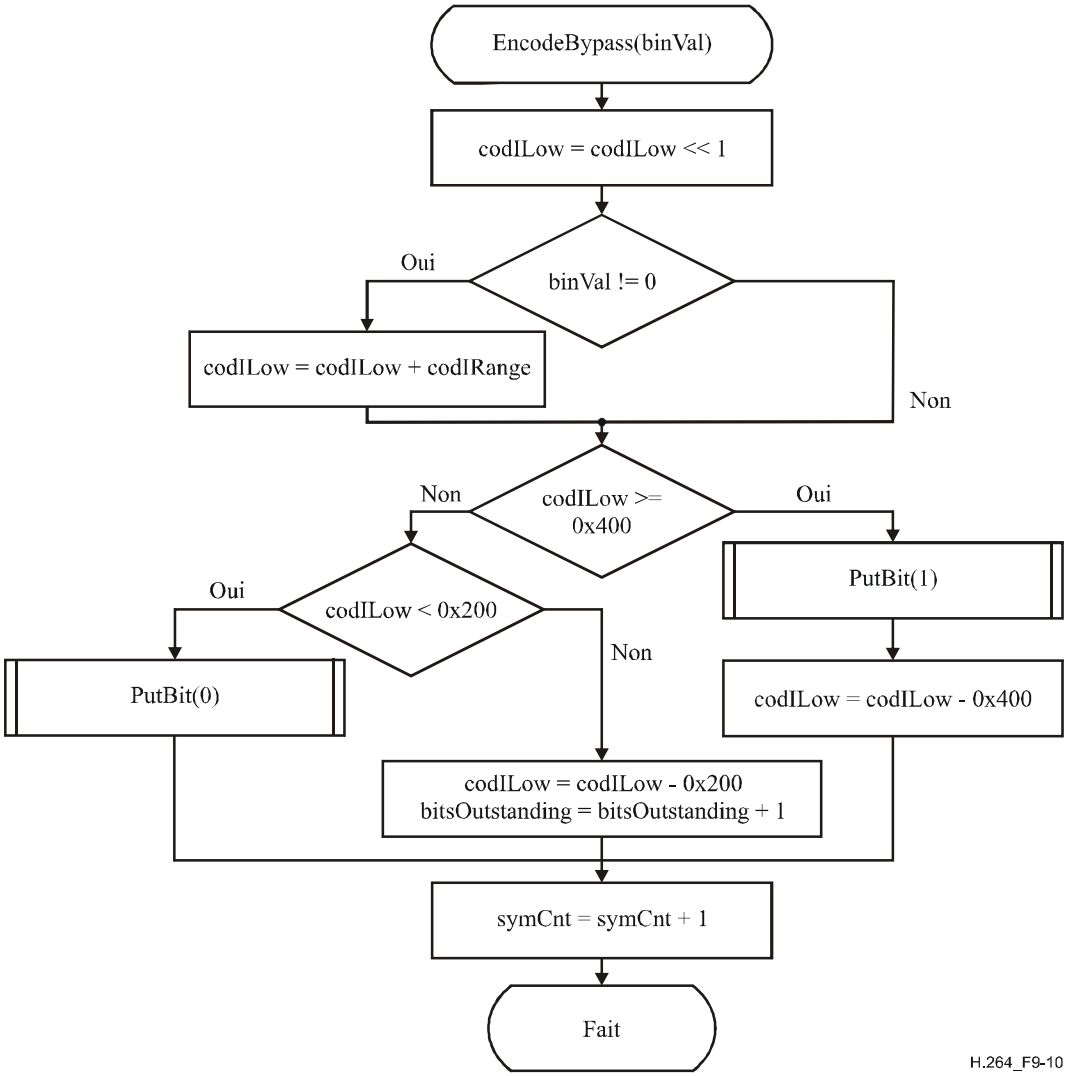


Figure 9-10 – Diagramme de l'aiguillage de codage

9.3.4.5 Processus de codage pour une décision binaire avant achèvement (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Les entrées dans ce processus sont les variables binVal, codIRange, codILow, bitsOutstanding, et symCnt.

Les résultats de ce processus sont zéro ou plusieurs bits écrits à la charge utile RBSP et les variables mises à jour codILow, codIRange, bitsOutstanding, et symCnt.

Ce programme de codage indiqué à la Figure 9-11 s'applique au codage de end_of_slice_flag et du segment indiquant le I_PCM mb_type tous deux associés à ctxIdx égal à 276.

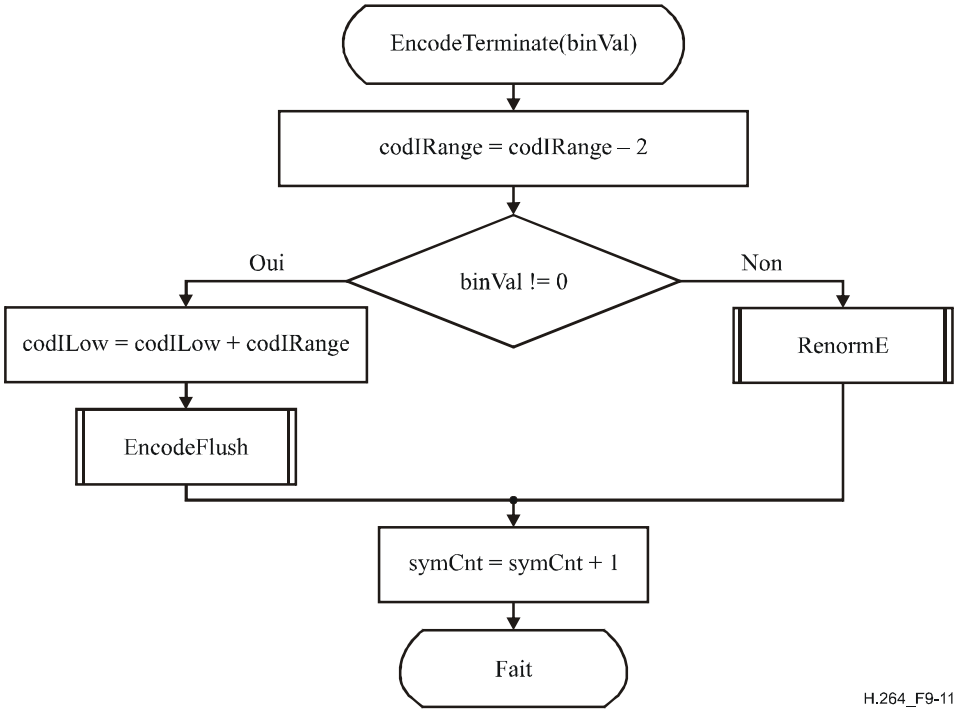


Figure 9-11 – Diagramme de codage d'une décision avant achèvement

Lorsque la valeur de binVal à coder est égale à 1, le codage CABAC est terminé et on applique la procédure de vidage indiquée à la Figure 9-12. Dans cette procédure de vidage, le dernier bit écrit par WriteBits (B, N) est égal à 1. Lors du codage de end_of_slice_flag, ce dernier bit est interprété comme le rbsp_stop_one_bit.

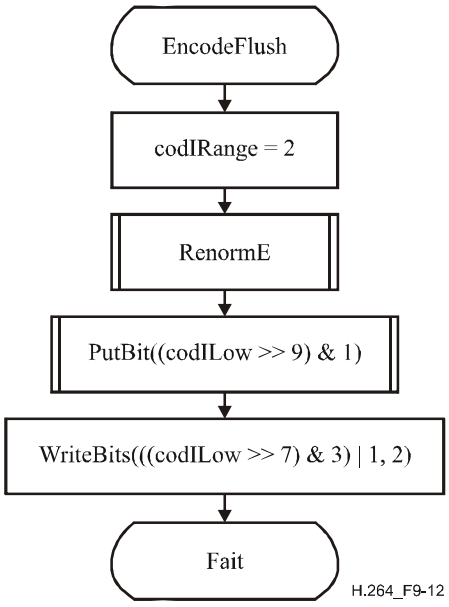


Figure 9-12 – Diagramme du vidage à achèvement

9.3.4.6 Processus de bourrage d'octet (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Ce processus est invoqué après le codage du dernier macrobloc de la dernière tranche d'une image et après encapsulation.

Les entrées dans ce processus sont le nombre d'octets NumBytesInVclNALunits de toutes les unités NAL de couche VCL d'une image, le nombre de macroblocs PicSizeInMbs dans l'image, et le nombre de symboles binaires BinCountsInNALunits résultant du codage du contenu de toutes les unités NAL de couche VCL de l'image.

Les résultats de ce processus sont zéro ou plusieurs octets accolés à l'unité NAL.

Soit la variable k qui est mise égale à Ceil((Ceil((3 * (32 * BinCountsInNALunits – RawMbBits * PicSizeInMbs) ÷ 1024) – NumBytesInVclNALunits) ÷ 3). En fonction de la variable k on applique ce qui suit.

– Si k est inférieur ou égale à 0, aucun mot cabac_zero_word n'est accolé à l'unité NAL.

– Sinon (si k est supérieur à 0), la séquence 0x000003 de trois octets est accolée k fois à l'unité NAL après encapsulation, où les deux premiers octets 0x0000 représentent un mot cabac_zero_word et le troisième octet 0x03 représente un octet emulation_prevention_three_byte.

Annexe A

Profils et niveaux

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Les profils et niveaux spécifient des restrictions sur les flux binaires et donc des limites sur les capacités nécessaires afin de décoder les flux binaires. Les profils et niveaux peuvent aussi être utilisés afin d'indiquer les points d'interfonctionnement entre les mises en œuvre de décodeurs individuels.

NOTE 1 – La présente Recommandation | Norme internationale n'inclut pas d'"options" individuelles à choisir pour le décodeur, car cela augmenterait les difficultés d'interfonctionnement.

Chaque profil spécifie un sous-ensemble de caractéristiques d'algorithme et de limites qui doivent être acceptées par tous les décodeurs conformes à ce profil.

NOTE 2 – Les codeurs ne sont pas tenus de faire usage d'un sous-ensemble particulier de caractéristiques constituant un profil.

Chaque niveau spécifie un ensemble de limites sur les valeurs que peuvent prendre les éléments syntaxiques de la présente Recommandation | Norme internationale. Le même ensemble de définitions de niveau est utilisé avec tous les profils, mais des mises en œuvre individuelles peuvent accepter un niveau différent pour chaque profil accepté. Pour tout profil donné, les niveaux correspondent généralement à la charge de traitement du décodeur et à la capacité de mémorisation.

A.1 Exigences sur la capacité du décodeur vidéo

Les capacités des décodeurs vidéo conformes à la présente Recommandation | Norme internationale sont spécifiées en termes de capacité à décoder les flux vidéo conformes aux contraintes des profils et niveaux spécifiés dans la présente annexe. Pour chacun de ces profils, le niveau accepté pour ce profil doit aussi être exprimé.

Des valeurs spécifiques sont spécifiées dans la présente annexe pour les éléments syntaxiques profile_idc et level_idc. Toutes les autres valeurs de profile_idc et level_idc sont réservées pour être utilisées à l'avenir par l'UIT-T | ISO/CEI.

NOTE – Les décodeurs ne devraient pas conclure que, lorsqu'une valeur réservée de profile_idc ou level_idc tombe entre les valeurs spécifiées dans la présente Recommandation | Norme internationale, cela indique des capacités intermédiaires entre les profils ou niveaux spécifiés, car il n'y a pas de restrictions sur la méthode que choisira l'UIT-T | ISO/CEI pour l'utilisation de telles valeurs réservées pour l'avenir.

A.2 Profils

A.2.1 Profil de base

Les flux binaires conformes au profil de base doivent respecter les contraintes suivantes:

– Seuls les types de tranche I et P peuvent être présents.

– Les flux d'unités NAL ne doivent pas contenir de valeurs de nal_unit_type dans la gamme de 2 à 4, inclus.

– Les ensembles de paramètres de séquence doivent avoir frame_mbs_only_flag (*fanion de macroblocs de trame uniquement*) égal à 1.

– Les éléments syntaxiques chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, qpprime_y_zero_transform_bypass_flag, et seq scaling_ matrix_ present flag ne doivent pas être présents dans des ensembles de paramètres de séquence.

– Les ensembles de paramètres d'image doivent avoir weighted_pred_flag (*fanion de prédiction pondérée*) et weighted_bipred_idc (*indice de biprédiction pondérée*) égaux tous deux à 0.

– Les ensembles de paramètres d'image doivent avoir entropy_coding_mode_flag (*fanion de mode de codage entropique*) égal à 0.

– Les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 (*numéro de groupes de tranche moins 1*) dans la gamme de 0 à 7, inclus.

– Les éléments syntaxiques transform_8x8_mode_flag, pic_scaling_matrix_present_flag, et second_chroma_qp_indice_offset ne doivent pas être présents dans des ensembles de paramètres d'image.

– L'élément syntaxique level_prefix ne doit pas être plus grand que 15.

– La contrainte de niveau spécifiée pour le profil de base au § A.3 doit être respectée.

La conformité d'un flux binaire au profil de base est spécifiée par profile_idc égal à 66.

Les décodeurs conformes au profil de base à un niveau spécifique doivent être capables de décoder tous les flux binaires dans lesquels profile_idc est égal à 66 ou dans lesquels constraint_set0_flag est égal à 1 et dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié.

A.2.2 Profil principal

Les flux binaires conformes au profil principal doivent respecter les contraintes suivantes:

– seuls les types de tranche I, P, et B peuvent être présents;

– les flux d'unités NAL ne doivent pas contenir de valeurs de nal_unit_type dans la gamme de 2 à 4 inclus;

– l'ordre arbitraire de tranche n'est pas admis;

– les éléments syntaxiques chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, qpprime_y_zero_transform_bypass_flag,et seq_ scaling_ matrix _present _flag ne doivent pas être présents dans des ensembles de paramètres de séquence;

– les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 égal à 0 uniquement;

– les ensembles de paramètres d'image doivent avoir redundant_pic_cnt_present_flag égal à 0 uniquement;

– les éléments syntaxiques transform_8x8_mode_flag, pic_scaling_matrix_present_flag, et second_chroma_qp_indice_offset ne doivent pas être présents dans des ensembles de paramètres d'image;

– l'élément syntaxique level_prefix ne doit pas être plus grand que 15 (lorsqu'il est présent);

– les contraintes de niveau spécifiées pour le profil principal au § A.3 doivent être respectées.

La conformité d'un flux binaire au profil principal est spécifiée par profile_idc égal à 77.

Les décodeurs conformes au profil principal à un niveau spécifié doivent être capables de décoder tous les flux binaires dans lesquels profile_idc est égal à 77 ou dans lesquels constraint_set1_flag est égal à 1 et dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié.

A.2.3 Profil étendu

Les flux binaires conformes au profil étendu doivent respecter les contraintes suivantes:

– les ensembles de paramètres de séquence doivent avoir direct_8x8_inference_flag égal à 1;

– les éléments syntaxiques chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, qpprime_y_zero_transform_bypass_flag, et seq_ scaling_ matrix_ present_flag ne doivent pas être présents dans des ensembles de paramètres de séquence;

– les ensembles de paramètres d'image doivent avoir entropy_coding_mode_flag égal à 0;

– les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 dans la gamme de 0 à 7 inclus;

– les éléments syntaxiques transform_8x8_mode_flag, pic_scaling_matrix_present_flag, et second_chroma_qp_indice_offset ne doivent pas être présents dans des ensembles de paramètres d'image;

– l'élément syntaxique level_prefix ne doit pas être plus grand que 15 (lorsqu'il est présent);

– les contraintes de niveau spécifiées pour le profil étendu au § A.3 doivent être respectées.

La conformité d'un flux binaire au profil étendu est spécifiée par profile_idc égal à 88.

Les décodeurs conformes au profil étendu à un niveau spécifié doivent être capables de décoder tous les flux binaires dans lesquels profile_idc est égal à 88 ou dans lesquels constraint_set2_flag est égal à 1 et dans lesquels level_idc représente un niveau inférieur ou égal au niveau spécifié.

Les décodeurs conformes au profil étendu à un niveau spécifié doivent être capables de décoder tous les flux binaires dans lesquels profile_idc est égal à 66 ou dans lesquels constraint_set0_flag est égal à 1, dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié.

A.2.4 Profil élevé

Les flux binaires conformes au profil élevé doivent respecter les contraintes suivantes:

– seuls les types de tranche I, P et B peuvent être présents;

– les flux d'unités de couche NAL ne doivent pas contenir de valeurs nal_unit_type dans l'étendue de 2 à 4, inclus;

– un ordre arbitraire des tranches n'est pas autorisé;

– les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 égal à 0 seulement;

– les ensembles de paramètres d'image doivent avoir redundant_pic_cnt_present_flag égal à 0 seulement;

– les ensembles de paramètres de séquence doivent avoir chroma_format_idc dans l'étendue de 0 à 1 inclus;

– les ensembles de paramètres de séquence doivent avoir bit_depth_luma_minus8 égal à 0 seulement;

– les ensembles de paramètres de séquence doivent avoir bit_depth_chroma_minus8 égal à 0 seulement;

– les ensembles de paramètres de séquence doivent avoir qpprime_y_zero_transform_bypass_flag égal à 0 seulement;

– les contraintes de niveau spécifiées pour le profil élevé dans le § A.3 doivent être satisfaites.

La conformité d'un flux binaire au profil élevé est spécifiée par: profil_idc égal à 100. Les décodeurs conformes au profil élevé à un niveau spécifique doivent être capables de décoder tous les flux binaires dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié et une des deux ou les deux conditions suivantes sont vraies:

– profile_idc est égal à 77 ou 100, ou

– constraint_set1_flag est égal à 1.

A.2.5 Profil élevé 10

Les flux binaires conformes au profil élevé 10 doivent respecter les contraintes suivantes:

– seuls les types de tranche I, P et B peuvent être présents;

– les flux d'unités de couche NAL ne doivent pas contenir de valeurs nal_unit_type dans l'étendue de 2 à 4, inclus;

– un ordre arbitraire des tranches n'est pas autorisé;

– les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 égal à 0 seulement;

– les ensembles de paramètres d'image doivent avoir redundant_pic_cnt_present_flag égal à 0 seulement;

– les ensembles de paramètres de séquence doivent avoir chroma_format_idc dans l'étendue de 0 à 1 inclus;

– les ensembles de paramètres de séquence doivent avoir bit_depth_luma_minus8 dans l'étendue de 0 à 2 inclus;

– les ensembles de paramètres de séquence doivent avoir bit_depth_chroma_minus8 dans l'étendue de 0 à 2 inclus;

– les ensembles de paramètres de séquence doivent avoir qpprime_y_zero_transform_bypass_flag égal à 0 seulement;

– les contraintes de niveau spécifiées pour le profil élevé 10 dans le § A.3 doivent être satisfaites.

La conformité d'un flux binaire au profil élevé 10 est spécifiée par: profil_idc égal à 110. Les décodeurs conformes au profil élevé 10 à un niveau spécifique doivent être capables de décoder tous les flux binaires dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié et une des deux ou les deux conditions suivantes sont vraies:

– profile_idc est égal à 77, 100, ou 110, ou

– constraint_set1_flag est égal à 1.

A.2.6 Profil élevé 4:2:2

Les flux binaires conformes au profil élevé 4:2:2 doivent respecter les contraintes suivantes:

– seuls les types de tranche I, P et B peuvent être présents;

– les flux d'unités de couche NAL ne doivent pas contenir de valeurs nal_unit_type dans l'étendue de 2 à 4, inclus;

– un ordre arbitraire des tranches n'est pas autorisé;

– les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 égal à 0 seulement;

– les ensembles de paramètres d'image doivent avoir redundant_pic_cnt_present_flag égal à 0 seulement;

– les ensembles de paramètres de séquence doivent avoir chroma_format_idc dans l'étendue de 0 à 2 inclus;

– les ensembles de paramètres de séquence doivent avoir bit_depth_luma_minus8 dans l'étendue de 0 à 2 inclus;

– les ensembles de paramètres de séquence doivent avoir bit_depth_chroma_minus8 dans l'étendue de 0 à 2 inclus;

– les ensembles de paramètres de séquence doivent avoir qpprime_y_zero_transform_bypass_flag égal à 0 seulement;

– Les contraintes de niveau spécifiées pour le profil élevé 4:2:2 dans le § A.3 doivent être satisfaites.

La conformité d'un flux binaire au profil élevé 4:2:2 est spécifiée par: profil_idc égal à 122. Les décodeurs conformes au profil élevé 4:2:2 à un niveau spécifique doivent être capables de décoder tous les flux binaires dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié et une des deux ou les deux conditions suivantes sont vraies:

– profile_idc est égal à 77, 100, 110, ou 122, ou

– constraint_set1_flag est égal à 1.

A.2.7 Profil élevé 4:4:4

Les flux binaires conformes au profil élevé 4:4:4 doivent respecter les contraintes suivantes:

– seuls les types de tranche I, P et B peuvent être présents;

– les flux d'unités de couche NAL ne doivent pas contenir de valeurs nal_unit_type dans l'étendue de 2 à 4, inclus;

– un ordre arbitraire des tranches n'est pas autorisé;

– les ensembles de paramètres d'image doivent avoir num_slice_groups_minus1 égal à 0 seulement;

– les ensembles de paramètres d'image doivent avoir redundant_pic_cnt_present_flag égal à 0 seulement;

– les ensembles de paramètres de séquence doivent avoir bit_depth_luma_minus8 dans l'étendue de 0 à 4 inclus;

– les ensembles de paramètres de séquence doivent avoir bit_depth_chroma_minus8 dans l'étendue de 0 à 4 inclus;

– les contraintes de niveau spécifiées pour le profil élevé 4:4:4 dans le § A.3 doivent être satisfaites.

La conformité d'un flux binaire au profil élevé 4:4:4 est spécifiée par: profil_idc égal à 144. Les décodeurs conformes au profil élevé 4:4:4 à un niveau spécifique doivent être capables de décoder tous les flux binaires dans lesquels level_idc et constraint_set3_flag représentent un niveau inférieur ou égal au niveau spécifié et une des deux ou les deux conditions suivantes sont vraies:

– profile_idc est égal à 77, 100, 110, 122, ou 144, ou

– constraint_set1_flag est égal à 1.

A.3 Niveaux

On spécifie ce qui suit afin d'exprimer les contraintes de la présente annexe.

– Soit l'unité d'accès n la $n^{ième}$ unité d'accès dans l'ordre de décodage, la première unité d'accès étant l'unité d'accès 0.

– Soit l'image n l'image codée primaire ou l'image décodée correspondante de l'unité d'accès n.

A.3.1 Limites de niveau communes aux profils de base, principal et étendu

Soit la variable fR calculée comme suit.

– Si l'image n est une trame, fR est mis égal à 1 ÷ 172.

– Sinon (si l'image n est une sous-trame), fR est mis égal à 1 ÷ (172 * 2).

Les flux binaires conformes aux profils de base, principal et étendu à un niveau spécifié doivent respecter les contraintes suivantes:

a) Le délai nominal de retrait de l'unité d'accès n (avec n > 0) de la mémoire tampon CPB, comme spécifié au § C.1.2, satisfait à la contrainte que $t_{r,n}(n) – t_r(n – 1)$ soit supérieur ou égal à Max(PicSizeInMbs ÷ MaxMBPS, fR), où MaxMBPS est la valeur spécifiée au Tableau A-1 qui s'applique à l'image n – 1, et PicSizeInMbs est le nombre de macroblocs dans l'image n – 1.

b) La différence entre les temps consécutifs de sortie des images de la mémoire tampon DPB, comme spécifié au § C.2.2, satisfait à la contrainte que $\Delta t_{o,dpb}(n) >=$ Max(PicSizeInMbs ÷ MaxMBPS, fR), où MaxMBPS est la valeur spécifiée au Tableau A-1 pour l'image n, et PicSizeInMbs est le nombre de macroblocs de l'image n, pourvu que l'image n soit une image qui soit sortie et ne soit pas la dernière image du flux binaire qui est sorti.

c) La somme des variables NumBytesInNALunit pour l'unité d'accès 0 est inférieure ou égale à 384 * (PicSizeInMbs + MaxMBPS * (t_r(0) – $t_{r,n}$(0))) ÷ MinCR, où MaxMBPS et MinCR sont les valeurs spécifiées au Tableau A-1 qui s'appliquent à l'image 0 et PicSizeInMbs est le nombre de macroblocs dans l'image 0.

d) La somme des variables NumBytesInNALunit pour l'unité d'accès n (avec n > 0) est inférieure ou égale à 384 * MaxMBPS * (t_r(n) – t_r(n – 1)) ÷ MinCR, où MaxMBPS et MinCR sont les valeurs spécifiées au Tableau A-1 qui s'appliquent à l'image n.

e) PicWidthInMbs * FrameHeightInMbs <= MaxFS, où MaxFS est spécifié au Tableau A-1.

f) PicWidthInMbs <= Sqrt(MaxFS * 8).

g) FrameHeightInMbs <= Sqrt(MaxFS * 8).

h) max_dec_frame_buffering <= MaxDpbSize, où MaxDpbSize est égal à Min(1024 * MaxDPB / (PicWidthInMbs * FrameHeightInMbs * 384), 16) et MaxDPB est donné au Tableau A-1 en unités de 1024 octets.

i) Pour les paramètres VCL HRD, BitRate[SchedSelIdx] <= 1000 * MaxBR et CpbSize[SchedSelIdx] <= 1000 * MaxCPB pour au moins une valeur de SchedSelIdx, où BitRate[SchedSelIdx] est donné par l'équation E-37 et CpbSize[SchedSelIdx] est donné par l'équation E-38 lorsque vcl_hrd_parameters_present_flag est égal à 1. MaxBR et MaxCPB sont spécifiés au Tableau A-1 en unités de 1000 bit/s et 1000 bit, respectivement. Le flux binaire doit satisfaire à ces conditions pour au moins une valeur de SchedSelIdx dans la gamme de 0 à cpb_cnt_minus1, inclus.

j) Pour les paramètres NAL HRD, BitRate[SchedSelIdx] <= 1200 * MaxBR et CpbSize[SchedSelIdx] <= 1200 * MaxCPB pour au moins une valeur de SchedSelIdx, où BitRate[SchedSelIdx] est donné par l'équation E-37 et CpbSize[SchedSelIdx] est donné par l'équation E-38 lorsque nal_hrd_parameters_present_flag est égal à 1. MaxBR et MaxCPB sont spécifiés au Tableau A-1 en unités de 1200 bit/s et 1200 bit, respectivement. Le flux binaire doit satisfaire à ces conditions pour au moins une valeur de SchedSelIdx dans la gamme de 0 à cpb_cnt_minus1.

k) La gamme de composante verticale de vecteurs cinétiques ne doit pas excéder, pour les vecteurs cinétiques luma, MaxVmvR en unités d'échantillons de trame luma, où MaxVmvR est spécifié au Tableau A-1.

NOTE 1 – Lorsque chroma_format_idc est égal à 1 et que le macrobloc actuel est un sous-macrobloc de trames, la gamme de composante de vecteurs cinétiques pour les vecteurs cinétiques chroma peut dépasser MaxVmvR en unités d'échantillons luma de trame, en raison de la méthode de calcul des vecteurs cinétiques chroma spécifiée dans le § 8.4.1.4.

l) La gamme de vecteurs cinétiques horizontaux ne doit pas excéder l'étendue de –2048 à 2047,75 inclus, en unités d'échantillons luma.

m) Le nombre de vecteurs cinétiques pour deux macroblocs consécutifs dans l'ordre de décodage (qui s'applique aussi au total depuis le dernier macrobloc d'une tranche et depuis le premier macrobloc de la tranche suivante dans l'ordre de décodage et qui s'applique aussi -en particulier- au total depuis le dernier macrobloc de la dernière tranche d'une image et depuis le premier macrobloc de la première tranche de l'image suivante, dans l'ordre du décodage) ne doit pas excéder MaxMvsPer2Mb, où MaxMvsPer2Mb est spécifié au Tableau A-1. Le nombre de vecteurs cinétiques pour chaque macrobloc correspond à la valeur de la variable MvCnt une fois achevé le processus d'intraprédiction ou d'interprédiction pour le macrobloc.

n) Le nombre de bits de données de macroblock_layer() pour tout macrobloc n'est pas supérieur à 3200. En fonction de entropy_coding_mode_flag, les bits de données de macroblock_layer() sont comptés comme suit.

- Si entropy_coding_mode_flag est égal à 0, le nombre de bits de données de macroblock_layer() est donné par le nombre de bits dans la structure syntaxique de macroblock_layer() pour un macrobloc.
- Sinon (si entropy_coding_mode_flag est égal à 1), le nombre de bits de données de macroblock_layer() pour un macrobloc est donné par le nombre de fois que read_bits(1) est appelé aux § 9.3.3.2.2 et 9.3.3.2.3 lors de l'analyse syntaxique de macroblock_layer() associé au macrobloc.

Le Tableau A-1 spécifie les limites pour chaque niveau. Les entrées marquées "–" dans le Tableau A-1 notent l'absence de limite correspondante. Aux fins de la comparaison des capacités en terme de niveau, un niveau doit être considéré comme étant inférieur (supérieur) à tout autre niveau s'il apparaît plus près de la rangée du haut (du bas) que l'autre niveau dans le Tableau A.1.

Un niveau auquel le flux binaire est conforme doit être indiqué par les éléments syntaxiques level_idc et constraint_set3_flag comme suit:

– Si level_idc est égal à 11 et si constraint_set3_flag est égal à 1, le niveau indiqué est le niveau 1b.

– Sinon (si level_idc n'est pas égal à 11 ou si constraint_set3_flag n'est pas égal à 1), level_idc doit être mis égal à une valeur de dix fois le numéro de niveau spécifié dans le Tableau A-1 et constraint_set3_flag doit être mis égal à 0.

Tableau A-1 – Limites de niveau

Numéro de niveau	Vitesse maximale de traitement de macrobloc MaxMBPS (MB/s)	Longueur max. de trame MaxFS (MB)	Taille max. de tampon d'image décodée MaxDPB (1024 octets pour 4:2:0)	Débit binaire vidéo maximal MaxBR (1000 bit/s, 1200 bit/s, cpbBrVclFact ou bit/s, ou cpbBrNalFact ou bit/s)	Taille maximale de CPB MaxCPB (1000 bit, 1200 bit, cpbBrVclFact ou bit, ou cpbBrNalFact ou bit)	Gamme de composant MV vertical MaxVmvR (échantillons de trame luma)	Taux minimal de compression MinCR	Nombre maximal de vecteurs cinétiques pour deux MB consécutifs MaxMvsPer2Mb
1	1 485	99	148.5	64	175	[-64,+63.75]	2	-
1b	1 485	99	148.5	128	350	[-64,+63.75]	2	-
1.1	3 000	396	337.5	192	500	[-128,+127.75]	2	-
1.2	6 000	396	891.0	384	1 000	[-128,+127.75]	2	-
1.3	11 880	396	891.0	768	2 000	[-128,+127.75]	2	-
2	11 880	396	891.0	2 000	2 000	[-128,+127.75]	2	-
2.1	19 800	792	1 782.0	4 000	4 000	[-256,+255.75]	2	-
2.2	20 250	1 620	3 037.5	4 000	4 000	[-256,+255.75]	2	-
3	40 500	1 620	3 037.5	10 000	10 000	[-256,+255.75]	2	32
3.1	108 000	3 600	6 750.0	14 000	14 000	[-512,+511.75]	4	16
3.2	216 000	5 120	7 680.0	20 000	20 000	[-512,+511.75]	4	16
4	245 760	8 192	12 288.0	20 000	25 000	[-512,+511.75]	4	16
4.1	245 760	8 192	12 288.0	50 000	62 500	[-512,+511.75]	2	16
4.2	522 240	8 704	13 056.0	50 000	62 500	[-512,+511.75]	2	16
5	589 824	22 080	41 400.0	135 000	135 000	[-512,+511.75]	2	16
5.1	983 040	36 864	69 120.0	240 000	240 000	[-512,+511.75]	2	16

Les niveaux avec des numéros de niveau non entiers dans le Tableau A-1 sont considérés comme "niveaux intermédiaires".

NOTE 2 – Tous les niveaux ont le même statut, mais certaines applications peuvent choisir de n'utiliser que les niveaux à numéro entier.

Pour information, le § A.3.4 montre les effets de ces limites sur les débits de tramedans plusieurs exemples de formats d'image.

A.3.2 Limites de niveau communes aux profils élevé, élevé 10, élevé 4:2:2 et élevé 4:4:4

Soit la variable fR à calculer comme suit:

– Si l'image n est une trame, fR est mis à 1 ÷ 172.

– Sinon (si l'image n est une sous-trame), fR est mis à 1 ÷ (172 * 2).

Les flux binaires conformes aux profils élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 à un niveau spécifié doivent respecter les contraintes suivantes:

a) le temps de retrait nominal de l'unité d'accès n (N > 0) du tampon CPB spécifié dans le § C.1.2, satisfait la contrainte que $t_{r,n}(n) – t_r(n – 1)$ est plus grand ou égal à Max(PicSizeInMbs ÷ MaxMBPS, fR), où MaxMBPS est la valeur spécifiée dans le Tableau A-1 qui s'applique à l'image n – 1, et où PicSizeInMbs est le nombre de macroblocs contenus dans l'image n – 1.

b) La différence entre instants consécutifs de sortie d'images du tampon DPB, spécifiée dans le § C.2.2, satisfait la contrainte que $\Delta t_{o,dpb}(n) >= Max(PicSizeInMbs ÷ MaxMBPS, fR)$, où MaxMBPS est la valeur spécifiée dans le Tableau A-1 pour l'image n, et où PicSizeInMbs est le nombre de macroblocs de l'image n, à condition que l'image n soit en sortie et qu'elle ne soit pas la dernière image du flux binaire en sortie.

c) PicWidthInMbs * FrameHeightInMbs <= MaxFS, où MaxFS est spécifié dans le Tableau A-1

d) PicWidthInMbs <= Sqrt(MaxFS * 8)

e) FrameHeightInMbs <= Sqrt(MaxFS * 8)

f) max_dec_Frame_buffering <= MaxDpbSize, où MaxDpbSize est égal à Min(1024 * MaxDPB / (PicWidthInMbs * FrameHeightInMbs * 384), 16) et où MaxDPB est spécifié dans le Tableau A-1.

g) L'étendue de la composante verticale des vecteurs cinétiques ne dépasse pas MaxVmvR en unités d'échantillons luma de trame, où MaxVmvR est spécifié dans le Tableau A-1.

h) L'étendue de la composante horizontale des vecteurs cinétiques ne dépasse pas l'étendue de -2048 à 2047.75, inclus, en unités d'échantillons luma.

i) Le nombre de vecteurs cinétiques par deux macroblocs consécutifs dans l'ordre du décodage (s'appliquant également au total depuis le dernier macrobloc d'une tranche et depuis le premier macrobloc de la prochaine tranche dans l'ordre du décodage) ne dépasse pas MaxMvsPer2Mb, où MaxMvsPer2Mb est spécifié dans le Tableau A-1. Le nombre de vecteurs cinétiques pour chaque macrobloc est la valeur de la variable MvCnt après l'achèvement du processus d'intraprédiction ou d'interprédiction pour le macrobloc.

j) Le nombre de bits des données macroblock_layer( ) pour un macrobloc quelconque n'est pas plus grand que 128 + RawMbBits. Selon le fanion entropy_coding_mode_flag, les bits des données macroblock_layer( ) sont décomptés comme suit:

   – Si entropy_coding_mode_flag est égal à 0, le nombre de bits de données macroblock_layer( ) est donné par le nombre de bits dans la structure syntaxique macroblock_layer( ) pour un macrobloc.

   – Sinon (si entropy_coding_mode_flag est égal à 1), le nombre de bits de données macroblock_layer() pour un macrobloc est donné par le nombre de fois où la structure read_bits(1) est appelée dans les § 9.3.3.2.2 et 9.3.3.2.3 lors d'une analyse syntaxique de la structure macroblock_layer( ) associée à ce macrobloc.

Le Tableau A-1 spécifie les limites pour chaque niveau. Les entrées marquées "-" dans le Tableau A-1 indiquent l'absence de limite correspondante.

Un niveau auquel le flux binaire est conforme doit être indiqué par l'élément syntaxique level_idc comme suit:

– Si level_idc est égal à 9, le niveau indiqué est le niveau 1b.

– Sinon (si level_idc n'est pas égal à 9), level_idc doit être mis égal à une valeur de dix fois le numéro de niveau spécifié dans le Tableau A-1.

A.3.3 Limites de niveau spécifiques d'un profil

a) Dans les flux binaires conformes aux profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4, le temps de retrait de l'unité d'accès 0 doit satisfaire à la contrainte que le nombre de tranches dans l'image 0 soit inférieur ou égal à (PicSizeInMbs + MaxMBPS * (t_r(0) – $t_{r,n}$(0))) ÷ SliceRate, où SliceRate est la valeur spécifiée au Tableau A-4 qui s'applique à l'image 0.

b) Dans les flux binaires conformes aux profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4, la différence entre le temps de retrait consécutif des unités d'accès n et n – 1 (avec n > 0) doit satisfaire à la contrainte que le nombre de tranches dans l'image n soit inférieur ou égal à MaxMBPS * (t_r(n) - t_r(n - 1)) ÷ SliceRate, où SliceRate est la valeur spécifiée au Tableau A-4 qui s'applique à l'image n.

c) Dans les flux binaires conformes aux profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4, les ensembles de paramètres de séquence doivent avoir le fanion direct_8x8_inference_flag égal à 1 pour les niveaux spécifiés au Tableau A-4.

NOTE 1 – Le fanion direct_8x8_inference_flag n'est pas pertinent pour le profil de base car il ne permet pas le type de tranche B (spécifié au § A.2.1), et direct_8x8_inference_flag est égal à 1 pour tous les niveaux du profil étendu (spécifié au § A.2.3).

d) Dans les flux binaires conformes au profil principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 ou aux profils étendus, les ensembles de paramètres de séquence doivent avoir frame_mbs_only_flag égal à 1 pour les niveaux spécifiés au Tableau A-4 pour les profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 et au Tableau A-5 pour le profil étendu.

NOTE 2 – Le fanion frame_mbs_only_flag est égal à 1 pour tous les niveaux du profil de base (spécifié au § A.2.1).

e) Dans les flux binaires conformes aux profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 ou aux profils étendus, la valeur de sub_mb_type dans les macroblocs B ne doit pas être égale à B_Bi_8x4, B_Bi_4x8, ou B_Bi_4x4 pour les niveaux dans lesquels MinLumaBiPredSize est indiqué comme 8x8 au Tableau A-4 pour les profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 et au Tableau A-5 pour le profil étendu.

f) Dans les flux binaires conformes aux profils: principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 ou aux profils étendus, ($xInt_{max} - xInt_{min} + 6$) * ($yInt_{max} - yInt_{min} + 6$) <= MaxSubMbRectSize dans les macroblocs codés avec mb_type égal à P_8x8, P_8x8ref0 ou B_8x8 pour toutes les invocations du processus spécifié au § 8.4.2.2.1 utilisé afin de produire le tableau d'échantillons luma prédits pour une seule liste d'images de référence (liste 0 ou liste 1 d'images de référence) pour chaque sous-macrobloc 8x8, où NumSubMbPart(sub_mb_type) > 1, où MaxSubMbRectSize est spécifié au Tableau A-3 pour le profil de base et au Tableau A-5 pour le profil étendu et

   – $xInt_{min}$ comme valeur minimale de $xInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc
   – $xInt_{max}$ comme valeur maximale de $xInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc
   – $yInt_{min}$ comme valeur minimale de $yInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc
   – $yInt_{max}$ comme valeur maximale de $yInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc.

g) Dans les flux binaires conformes au profil élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 pour les paramètres du décodeur HRD en couche VCL, BitRate[SchedSelIdx] <= cpbBrVclFactor * MaxBR et CpbSize[SchedSelIdx] <= cpbBrVclFactor * MaxCPB pour au moins une seule valeur de SchedSelIdx, où cpbBrVclFactor est spécifié dans le Tableau A-2, BitRate[SchedSelIdx] est spécifié par l'équation E-37 et CpbSize[SchedSelIdx] est spécifié par l'équation E-38 lorsque le fanion vcl_hrd_parameters_present_flag est égal à 1. MaxBR et MaxCPB sont respectivement spécifiés dans le Tableau A-1 en unités de cpbBrVclFactor x bit/s et cpbBrVclFactor x bit. Le flux binaire doit satisfaire ces conditions pour au moins une seule valeur de SchedSelIdx dans l'étendue de 0 à cpb_cnt_minus1, inclus.

h) Dans les flux binaires conformes au profil élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4 pour les paramètres du décodeur HRD en couche NAL, BitRate[SchedSelIdx] <= cpbBrNalFactor * MaxBR et CpbSize[SchedSelIdx] <= cpbBrNalFactor * MaxCPB pour au moins une seule valeur de SchedSelIdx, où cpbBrNalFactor est spécifié dans le Tableau A-2, BitRate[SchedSelIdx] est spécifié par l'équation E-37 et CpbSize[SchedSelIdx] est spécifié par l'équation E-38 lorsque le fanion nal_hrd_parameters_present_flag est égal à 1. MaxBR et MaxCPB sont respectivement spécifiés dans le Tableau A-1 en unités de cpbBrNalFactor x bit/s et cpbBrNalFactor x bit. Le flux binaire doit satisfaire ces conditions pour au moins une seule valeur de SchedSelIdx dans l'étendue de 0 à cpb_cnt_minus1.

Tableau A-2 – Spécification des facteurs cpbBrVclFactor et cpbBrNalFactor

Profil	cpbBrVclFactor	cpbBrNalFactor
Elevé	1 250	1 500
Elevé 10	3 000	3 600
Elevé 4:2:2	4 000	4 800
Elevé 4:4:4	4 000	4 800

A.3.3.1 Limites du profil de base

Le Tableau A-3 spécifie les limites pour chaque niveau qui est spécifique des flux binaires conformes au profil de base. Les entrées marquées "–" au Tableau A-3 notent l'absence de limite correspondante.

Tableau A-3 – Limites de niveau du profil de base

Numéro de niveau	MaxSubMbRectSize
1	576
1b	576
1.1	576
1.2	576
1.3	576
2	576
2.1	576
2.2	576
3	576
3.1	-
3.2	-
4	-
4.1	-
4.2	-
5	-
5.1	-

A.3.3.2 Limites de profil principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4

Le Tableau A-4 spécifie les limites pour chaque niveau qui est spécifique des flux binaires conformes au profil principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4. Les entrées marquées "–" au Tableau A-4 notent l'absence de limite correspondante.

Tableau A-4 – Limites de niveau du profil principal, élevé, élevé 10, élevé 4:2:2 ou élevé 4:4:4

Numéro de niveau	SliceRate	MinLumaBiPredSize	direct_8x8_inference_flag	frame_mbs_only_flag
1	-	-	-	1
1b	-	-	-	1
1.1	-	-	-	1
1.2	-	-	-	1
1.3	-	-	-	1
2	-	-	-	1
2.1	-	-	-	-
2.2	-	-	-	-
3	22	-	1	-
3.1	60	8x8	1	-
3.2	60	8x8	1	-
4	60	8x8	1	-
4.1	24	8x8	1	-
4.2	24	8x8	1	1
5	24	8x8	1	1
5.1	24	8x8	1	1

A.3.3.3 Limites de profil étendu

Le Tableau A-5 spécifie les limites pour chaque niveau qui est spécifique des flux binaires conformes au profil étendu. Les entrées marquées "–" dans le Tableau A-5 indiquent l'absence de limite correspondante.

Tableau A-5 – Limites de niveau du profil étendu

Numéro de niveau	MaxSubMbRectSize	MinLumaBiPredSize	frame_mbs_only_flag
1	576	-	1
1b	576	-	1
1.1	576	-	1
1.2	576	-	1
1.3	576	-	1
2	576	-	1
2.1	576	-	-
2.2	576	-	-
3	576	-	-
3.1	-	8x8	-
3.2	-	8x8	-
4	-	8x8	-
4.1	-	8x8	-
4.2	-	8x8	1
5	-	8x8	1
5.1	-	8x8	1

A.3.4 Effet des limites de niveau sur le débit de trame (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

Tableau A-6 – Débits de trame maximaux (trames par seconde) pour quelques exemples de longueurs de trame

Niveau:					**1**	**1b**	**1.1**	**1.2**	**1.3**	**2**	**2.1**
L. max. de trame (macroblocs):					**99**	**99**	**396**	**396**	**396**	**396**	**792**
Max (macroblocs):					**1 485**	**1 485**	**3 000**	**6 000**	**11 880**	**11 880**	**19 800**
Longueur max. de trame (éch.):					25 344	25 344	101 376	101 376	101 376	101 376	202 752
Max. échantillons/s:					380 160	380 160	768 000	1 536 000	3 041 280	3 041 280	5 068 800
Format	**Largeur luma**	**Hauteur luma**	**Total de MB**	**Echant. luma**							
SQCIF	**128**	**96**	48	12 288	30.9	30.9	62.5	125.0	172.0	172.0	172.0
QCIF	**176**	**144**	99	25 344	15.0	15.0	30.3	60.6	120.0	120.0	172.0
QVGA	**320**	**240**	300	76 800	-	-	10.0	20.0	39.6	39.6	66.0
525 SIF	**352**	**240**	330	84 480	-	-	9.1	18.2	36.0	36.0	60.0
CIF	**352**	**288**	396	101 376	-	-	7.6	15.2	30.0	30.0	50.0
525 HHR	**352**	**480**	660	168 960	-	-	-	-	-	-	30.0
625 HHR	**352**	**576**	792	202 752	-	-	-	-	-	-	25.0
VGA	**640**	**480**	1 200	307 200	-	-	-	-	-	-	-
525 4SIF	**704**	**480**	1 320	337 920	-	-	-	-	-	-	-
525 SD	**720**	**480**	1 350	345 600	-	-	-	-	-	-	-
4CIF	**704**	**576**	1 584	405 504	-	-	-	-	-	-	-
625 SD	**720**	**576**	1 620	414 720	-	-	-	-	-	-	-
SVGA	**800**	**600**	1 900	486 400	-	-	-	-	-	-	-
XGA	**1024**	**768**	3 072	786 432	-	-	-	-	-	-	-
720p HD	**1280**	**720**	3 600	921 600	-	-	-	-	-	-	-
4VGA	**1280**	**960**	4 800	1 228 800	-	-	-	-	-	-	-
SXGA	**1280**	**1024**	5 120	1 310 720	-	-	-	-	-	-	-
525 16SIF	**1408**	**960**	5 280	1 351 680	-	-	-	-	-	-	-
16CIF	**1408**	**1152**	6 336	1 622 016	-	-	-	-	-	-	-
4SVGA	**1600**	**1200**	7 500	1 920 000	-	-	-	-	-	-	-
1080 HD	**1920**	**1088**	8 160	2 088 960	-	-	-	-	-	-	-
2Kx1K	**2048**	**1024**	8 192	2 097 152	-	-	-	-	-	-	-
2Kx1080	**2048**	**1088**	8 704	2 228 224	-	-	-	-	-	-	-
4XGA	**2048**	**1536**	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	**2560**	**1920**	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	-	-	-	-	-	-

Tableau A-6 (suite) – Débits de trame maximaux (trames par seconde) pour quelques exemples de longueurs de trame

Niveau:					**2.2**	**3**	**3.1**	**3.2**	**4**	**4.1**	**4.2**
Longueur max. de trame (macroblocks):					**1 620**	**1 620**	**3 600**	**5 120**	**8 192**	**8 192**	**8 704**
Max macroblocks/s:					**20 250**	**40 500**	**108 000**	**216 000**	**245 760**	**245 760**	**522 240**
Longueur max. de trame (éch.):					414 720	414 720	921 600	1 310 720	2 097 152	2 097 152	2 228 224
Max. échantillon/s:					5 184 000	10 368 000	27 648 000	55 296 000	62 914 560	62 914 560	133 693 440
Format	**Largeur luma**	**Hauteur luma**	**Total de MBs**	**Echant. luma**							
SQCIF	**128**	**96**	48	12 288	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QCIF	**176**	**144**	99	25 344	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QVGA	**320**	**240**	300	76 800	67.5	135.0	172.0	172.0	172.0	172.0	172.0
525 SIF	**352**	**240**	330	84 480	61.4	122.7	172.0	172.0	172.0	172.0	172.0
CIF	**352**	**288**	396	101 376	51.1	102.3	172.0	172.0	172.0	172.0	172.0
525 HHR	**352**	**480**	660	168 960	30.7	61.4	163.6	172.0	172.0	172.0	172.0
625 HHR	**352**	**576**	792	202 752	25.6	51.1	136.4	172.0	172.0	172.0	172.0
VGA	**640**	**480**	1 200	307 200	16.9	33.8	90.0	172.0	172.0	172.0	172.0
525 4SIF	**704**	**480**	1 320	337 920	15.3	30.7	81.8	163.6	172.0	172.0	172.0
525 SD	**720**	**480**	1 350	345 600	15.0	30.0	80.0	160.0	172.0	172.0	172.0
4CIF	**704**	**576**	1 584	405 504	12.8	25.6	68.2	136.4	155.2	155.2	172.0
625 SD	**720**	**576**	1 620	414 720	12.5	25.0	66.7	133.3	151.7	151.7	172.0
SVGA	**800**	**600**	1 900	486 400	-	-	56.8	113.7	129.3	129.3	172.0
XGA	**1024**	**768**	3 072	786 432	-	-	35.2	70.3	80.0	80.0	172.0
720p HD	**1280**	**720**	3 600	921 600	-	-	30.0	60.0	68.3	68.3	145.1
4VGA	**1280**	**960**	4 800	1 228 800	-	-	-	45.0	51.2	51.2	108.8
SXGA	**1280**	**1024**	5 120	1 310 720	-	-	-	42.2	48.0	48.0	102.0
525 16SIF	**1408**	**960**	5 280	1 351 680	-	-	-	-	46.5	46.5	98.9
16CIF	**1408**	**1152**	6 336	1 622 016	-	-	-	-	38.8	38.8	82.4
4SVGA	**1600**	**1200**	7 500	1 920 000	-	-	-	-	32.8	32.8	69.6
1080 HD	**1920**	**1088**	8 160	2 088 960	-	-	-	-	30.1	30.1	64.0
2Kx1K	**2048**	**1024**	8 192	2 097 152	-	-	-	-	30.0	30.0	63.8
2Kx1080	**2048**	**1088**	8 704	2 228 224	-	-	-	-	-	-	60.0
4XGA	**2048**	**1536**	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	**2560**	**1920**	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	-	-	-	-	-	-

Tableau A-6 (fin) – Débits de trame maximaux (trames par seconde) pour quelques exemples de longueurs de trame

Niveau:					**5**	**5.1**
Longueur max. de trame (macroblocs):					**22 080**	**36 864**
Max macroblocs/s:					**589 824**	**983 040**
Longueur max. de trame (éch.):					5 652 480	9 437 184
Max. échantillon/s:					150 994 944	251 658 240
Format	**Largeur luma**	**Hauteur luma**	**Total de MB**	**Échant. luma**		
SQCIF	**128**	**96**	48	12 288	172.0	172.0
QCIF	**176**	**144**	99	25 344	172.0	172.0
QVGA	**320**	**240**	300	76 800	172.0	172.0
525 SIF	**352**	**240**	330	84 480	172.0	172.0
CIF	**352**	**288**	396	101 376	172.0	172.0
525 HHR	**352**	**480**	660	168 960	172.0	172.0
625 HHR	**352**	**576**	792	202 752	172.0	172.0
VGA	**640**	**480**	1 200	307 200	172.0	172.0
525 4SIF	**704**	**480**	1 320	337 920	172.0	172.0
525 SD	**720**	**480**	1 350	345 600	172.0	172.0
4CIF	**704**	**576**	1 584	405 504	172.0	172.0
625 SD	**720**	**576**	1 620	414 720	172.0	172.0
SVGA	**800**	**600**	1 900	486 400	172.0	172.0
XGA	**1024**	**768**	3 072	786 432	172.0	172.0
720p HD	**1280**	**720**	3 600	921 600	163.8	172.0
4VGA	**1280**	**960**	4 800	1 228 800	122.9	172.0
SXGA	**1280**	**1024**	5 120	1 310 720	115.2	172.0
525 16SIF	**1408**	**960**	5 280	1 351 680	111.7	172.0
16CIF	**1408**	**1152**	6 336	1 622 016	93.1	155.2
4SVGA	**1600**	**1200**	7 500	1 920 000	78.6	131.1
1080 HD	**1920**	**1088**	8 160	2 088 960	72.3	120.5
2Kx1K	**2048**	**1024**	8 192	2 097 152	72.0	120.0
2Kx1080	**2048**	**1088**	8 704	2 228 224	67.8	112.9
4XGA	**2048**	**1536**	12 288	3 145 728	48.0	80.0
16VGA	**2560**	**1920**	19 200	4 915 200	30.7	51.2
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176	27.2	45.3
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480	26.7	44.5
4Kx2K	**4096**	**2048**	32 768	8 388 608	-	30.0
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184	-	26.7

Il convient de noter ce qui suit:

– La présente Recommandation | Norme internationale est une spécification de trame à longueur variable. Les longueurs de trame spécifiques du Tableau A-6 sont uniquement données à titre d'exemple.

– Tel qu'utilisé dans le Tableau A-6, "525" se rapporte à une utilisation typique dans les environnements utilisant 525 lignes de balayage analogique (dont environ 480 lignes contiennent la région visible de l'image), et "625" se rapporte aux environnements utilisant 625 lignes de balayage analogique (dont environ 576 lignes contiennent la région visible de l'image).

– XGA est aussi connu sous le nom de (aka) XVGA, 4SVGA aka UXGA, 16XGA aka 4Kx3K, CIF aka 625 SIF, 625 HHR aka 2CIF aka demi 625 D-1, aka demi 625 UIT-R BT.601, 525 SD aka 525 D-1 aka 525 UIT-R BT.601, 625 SD aka 625 D-1 aka 625 UIT-R BT.601.

– Les débits de trame donnés sont corrects pour les modes de balayage progressifs. Les débits de trame sont aussi corrects pour le codage vidéo entrelacé pour les cas de hauteur de trame divisible par 32.

Annexe B

Format de flux d'octets

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie la syntaxe et la sémantique d'un format de flux d'octets spécifié afin d'être utilisé par des applications qui fournissent certains ou tous les flux d'unités NAL comme flux ordonnés d'octets ou de bits au sein desquels les localisations des frontières des unités NAL doivent être identifiables à partir des schémas inclus dans les données, tels que les systèmes de la Rec. UIT-T H.222.0 | ISO/CEI 13818-1 ou de la Rec. UIT-T H.320. Pour les applications orientées bit, l'ordre binaire pour le format de flux d'octet est spécifié comme commençant avec le bit de plus fort poids (MSB) du premier octet, puis en continuant jusqu'au bit de plus faible poids (LSB) du premier octet, suivi par le MSB du second octet, etc.

Le format de flux d'octets consiste en une séquence de structures syntaxiques d'unités NAL de flux d'octet. Chaque structure syntaxique d'unité NAL de flux d'octet contient un préfixe de code de déclenchement suivi d'une structure syntaxique nal_unit(NumBytesInNALunit). Il peut (et dans certaines circonstances, il doit) aussi contenir un élément syntaxique zero_byte supplémentaire. Il peut aussi contenir un ou plusieurs éléments syntaxiques trailing_zero_8bits supplémentaires. Lorsqu'il s'agit de la première unité NAL de flux d'octet du flux linéaire, il peut aussi contenir un ou plusieurs éléments syntaxiques leading_zero_8bits supplémentaires.

B.1 Syntaxe et sémantique d'unité NAL de flux d'octet

B.1.1 Syntaxe d'unité NAL de flux d'octet

byte_stream_nal_unit(NumBytesInNALunit) {	C	Descripteur
tant que(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
leading_zero_8bits /* égal à 0x00 */		f(8)
si(next_bits(24) != 0x000001)		
zero_byte /* equal to 0x00 */		f(8)
start_code_prefix_one_3bytes /* égal à 0x000001 */		f(24)
nal_unit(NumBytesInNALunit)		
tant que(more_data_in_byte_stream( ) && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
trailing_zero_8bits /* égal à 0x00 */		f(8)
}		

B.1.2 Sémantique d'unité NAL de flux d'octet

L'ordre des unités NAL de flux d'octet dans le flux d'octet doit suivre l'ordre de décodage des unités NAL contenues dans le flux d'octet des unités NAL (voir le § 7.4.1.2). Le contenu de chaque unité NAL de flux d'octet est associé à la même unité d'accès que l'unité NAL contenue dans l'unité NAL de flux d'octet (voir le § 7.4.1.2.3).

leading_zero_8bits est un octet égal à 0x00.

NOTE – L'élément syntaxique leading_zero_8bits ne peut être présent que dans la première unité NAL de flux d'octet du flux binaire, étant donné que (comme il ressort du diagramme de syntaxe du § B.1.1) les octets égaux à 0x00 qui suivent une structure syntaxique d'unité NAL et qui précèdent la séquence de quatre octets 0x00000001 (qui doit être interprétée comme un élément zero_byte suivi d'un élément start_code_prefix_one_3bytes) seront considérés comme des éléments syntaxiques trailing_zero_8bits qui font partie de l'unité NAL de flux d'octet précédente.

zero_byte est un seul octet égal à 0x00.

Lorsque l'une des conditions suivantes est remplie, les 'éléments syntaxiques zero_byte doivent être présents.

– le nal_unit_type au sein de nal_unit() est égal à 7 (ensemble de paramètres de séquence) ou 8 (ensemble de paramètres d'image);

– la structure syntaxique de l'unité NAL de flux d'octet contient la première unité NAL d'une unité d'accès dans l'ordre de décodage, comme spécifié au § 7.4.1.2.3.

start_code_prefix_one_3bytes est une séquence de valeur fixe de 3 octets égale à 0x000001. Cet élément syntaxique est appelé un préfixe de code de déclenchement.

trailing_zero_8bits est un octet égal à 0x00.

B.2 Processus de décodage d'unité NAL de flux d'octet

L'entrée dans ce processus consiste en un flux ordonné d'octets fait d'une séquence de structures syntaxiques d'unités NAL de flux d'octets. La sortie du processus est une réponse de structures syntaxiques d'unités NAL.

Au début du processus de décodage, le décodeur initialise sa position courante dans le flux d'octets au début de ce flux. Puis il extrait et supprime chaque élément syntaxique leading_zero_8bits (le cas échéant) en déplaçant la position courante dans le flux d'octets vers l'avant à raison d'un octet à la fois, jusqu'à ce que la position courante dans le flux d'octets soit telle que les quatre octets suivants du flux binaire forment la séquence de quatre octets 0x00000001.

Le décodeur effectue alors le processus par étapes de façon répétitive afin d'extraire et décoder chaque structure syntaxique d'unités NAL dans le flux d'octets jusqu'à ce qu'il arrive à la fin du flux d'octets (telle que déterminée par des moyens non spécifiés) et que la dernière unité NAL du flux binaire ait été décodée:

1. lorsque les quatre octets suivants dans le flux binaire forment la séquence de quatre octets 0x00000001, l'octet suivant dans le flux d'octets (qui est un élément syntaxique zero_byte) est extrait et supprimé et la position courante dans le flux d'octets est mis égal à la position de l'octet qui suit cet octet supprimé;
2. la prochaine séquence de trois octets dans le flux d'octets (qui est un élément syntaxique start_code_prefix_one_3bytes) est extraite et supprimée et la position courante dans le flux d'octet est mise égale à la position de l'octet suivant cette séquence de trois octets;
3. NumBytesInNALunit est mis égal au nombre d'octets commençant par l'octet à la position courante dans le flux d'octet, jusques et y compris le dernier octet qui précède la localisation de toute condition suivante:
   a. une séquence suivante de trois octets alignés sur les octets égale à 0x000000, ou
   b. une séquence suivante de trois octets alignés sur les octets égale à 0x000001, ou
   c. la fin du flux d'octets, telle que déterminée par des moyens non spécifiés;
4. NumBytesInNALunit octets sont retirés du flux binaire et la position en cours dans le flux d'octets est avancée de NumBytesInNALunit octets. Cette séquence d'octets est nal_unit(NumBytesInNALunit) et elle est décodée au moyen du processus de décodage d'unités NAL;
5. lorsque la position courante dans le flux d'octets ne correspond pas à la fin du flux d'octets (telle que déterminée par des moyens non spécifiés) et que les octets suivants dans le flux d'octets ne commencent pas par une séquence de trois octets égale à 0x000001 et que les octets suivants dans le flux d'octets ne commencent pas par une séquence de quatre octets égale à 0x00000001, le décodeur extrait et supprime chaque élément syntaxique trailing_zero_8bits, en déplaçant la position courante dans le flux d'octets à la fois, jusqu'à ce que la position courante dans le flux d'octets soit telle que les octets suivants dans le flux d'octets forment la séquence de quatre octets 0x00000001 où que l'on soit arrivé à la fin du flux d'octets (telle que déterminée par des moyens non spécifiés).

B.3 Récupération du verrouillage d'octet du décodeur (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

De nombreuses applications fournissent les données à un décodeur d'une façon qui est intrinsèquement alignée sur les octets, et donc il n'est aucun besoin de la procédure de détection du verrouillage d'octet orientée sur les bits décrite au présent paragraphe.

On dit d'un décodeur qu'il a un verrouillage d'octet avec un flux binaire lorsque ce décodeur est en mesure de déterminer si les positions des données dans le flux binaire sont ou non alignées sur les octets. Lorsqu'un décodeur n'a pas de verrouillage d'octet avec le flux d'octets du codeur, le décodeur peut examiner le flux binaire entrant afin de chercher le schéma binaire '00000000 00000000 00000000 00000001' (31 bit consécutifs égaux à 0 suivis d'un bit égal à 1). Le bit suivant immédiatement ce schéma est le premier bit d'un octet verrouillé suivant un préfixe de code de déclenchement. A la détection de ce schéma, le décodeur sera verrouillé sur le codeur et positionné au début d'une unité NAL dans le flux d'octet.

Une fois le verrouillage d'octet effectué avec le codeur, le décodeur peut examiner le flux d'octet entrant afin de trouver les séquences de trois octets 0x000001 et 0x000003 suivantes.

Lorsque la séquence de trois octets 0x000001 est détectée, il s'agit d'un préfixe de code de déclenchement.

Lorsque la séquence de trois octets 0x000003 est détectée, le troisième octet (0x03) est un emulation_prevention_three_byte à détruire comme spécifié au § 7.4.1.

Lorsqu'une erreur dans la syntaxe du flux binaire est détectée (p. ex., une valeur différente de zéro du fanion forbidden_zero_bit ou une des séquences à trois octets ou à quatre octets qui sont interdites dans le § 7.4.1), le décodeur peut consider la condition détectée comme une indication que le verrouillage en octets peut avoir été perdu et peut rejeter toutes les données de flux binaire jusqu'à la détection du verrouillage en octets à une position ultérieure dans le flux binaire comme décrit dans le présent paragraphe.

Annexe C

Décodeur fictif de référence

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie le décodeur fictif de référence (HRD) et son utilisation afin de vérifier la conformité du flux binaire et du décodeur.

Deux types de flux binaires font l'objet de la vérification de conformité du décodeur HRD pour la présente Recommandation | Norme internationale. Le premier de ces types de flux binaire, appelé flux binaire de Type I, est un flux d'unités NAL ne contenant que des unités NAL de couche VCL et des unités NAL de données de remplisssage pour toutes les unités d'accès dans le flux binaire. Le second type de flux binaire, appelé flux binaire de Type II, contient, en plus des unités NAL de couche VCL et des unités NAL de données de remplissage pour toutes les unités d'accès dans le flux binaire, au moins un des éléments suivants:

– des unités NAL non VCL supplémentaires autres que des unités NAL de données de remplissage;

– tous les éléments syntaxiques leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes et trailing_zero_8bits qui forment un flux d'octets à partir du flux d'unités NAL (comme spécifié à l'Annexe B).

La Figure C-1 montre les types de points de conformité de flux binaire vérifiés par le décodeur HRD.

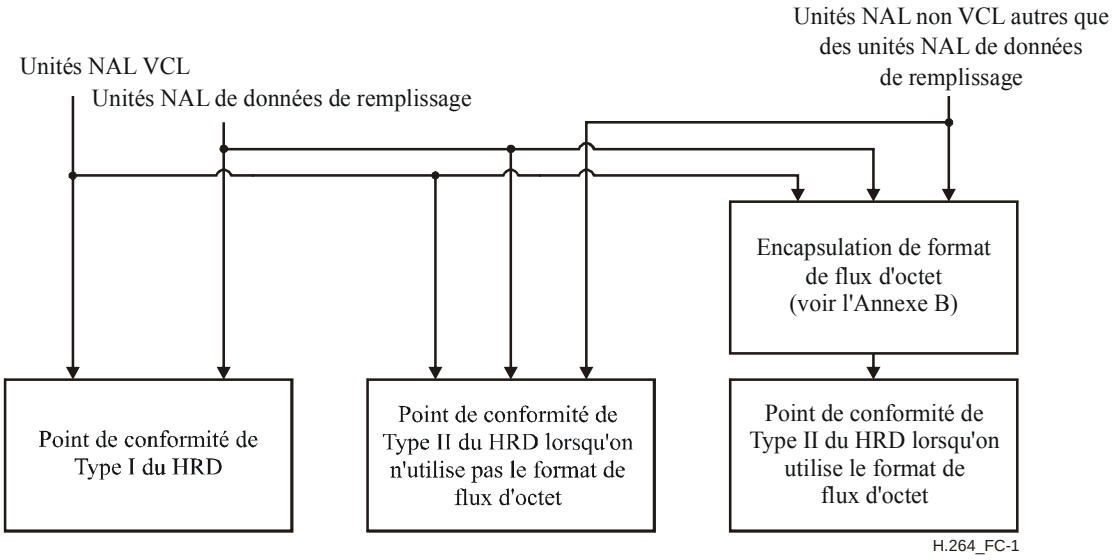


Figure C-1 – Structure des flux d'octets et des flux d'unités NAL pour les vérifications de conformité par le décodeur HRD

Les éléments syntaxiques des unités NAL non VCL (ou leurs valeurs par défaut pour certains éléments syntaxiques) requis pour le décodeur HRD sont spécifiés aux paragraphes de sémantique du § 7 et aux Annexes D et E.

Deux types d'ensembles de paramètres de décodeur HRD sont utilisés. Les ensembles de paramètres de décodeur HRD sont signalés par des informations de facilité d'utilisation vidéo, comme spécifié aux § E.1 et E.2, qui font partie de la structure syntaxique d'ensembles de paramètres de séquence.

Afin de vérifier la conformité d'un flux binaire au moyen du décodeur HRD, tous les ensembles de paramètres de séquences et les ensembles de paramètres d'image mentionnés dans les unités NAL de couche VCL, et les messages SEI correspondants de période de mise en mémoire tampon et de temporisation d'image doivent être convoyés au décodeur HRD en temps utile, soit dans le flux binaire (par des unités NAL non VCL), soit par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale.

Dans les Annexes C, D et E, la spécification pour "présence" des unités NAL non VCL est aussi satisfaite lorsque ces unités NAL (ou seulement certaines d'entre elles) sont convoyées aux décodeurs (ou au décodeur HRD) par d'autres moyens non spécifiés par la présente Recommandation | Norme internationale. Pour les besoins du comptage des bits, seuls les bits appropriés qui sont réellement présents dans le flux binaire sont comptés.

NOTE 1 – A titre d'exemple, la synchronisation d'une unité NAL non VCL, convoyée par des moyens autres que la présence dans le flux binaire, avec des unités NAL qui sont présentes dans le flux binaire, peut être réalisée en indiquant deux points dans le flux binaire, entre lesquels l'unité NAL non VCL aurait été présente dans le flux binaire, si le codeur avait décidé de la convoyer dans le flux binaire.

Lorsque le contenu d'une unité NAL non VCL est convoyé afin d'être appliqué par des moyens autres que la présence au sein du flux binaire, la représentation du contenu de l'unité NAL non VCL n'est pas nécessaire afin d'utiliser la même syntaxe que celle spécifiée dans la présente annexe.

NOTE 2 – Lorsque les informations du décodeur HRD sont contenues au sein du flux binaire, il est possible de vérifier la conformité d'un flux binaire aux exigences du présent paragraphe fondées uniquement sur les informations contenues dans le flux binaire. Lorsque les informations du décodeur HRD ne sont pas présentes dans le flux binaire, comme c'est le cas pour tous les flux binaires de Type I autonomes, la conformité ne peut être vérifiée que lorsque les données du décodeur HRD sont fournies par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale.

Le décodeur HRD contient une mémoire tampon d'image codée (CPB), un processus de décodage instantané, une mémoire tampon d'image décodée (DPB), et le recadrage du résultat comme indiqué à la Figure C-2.

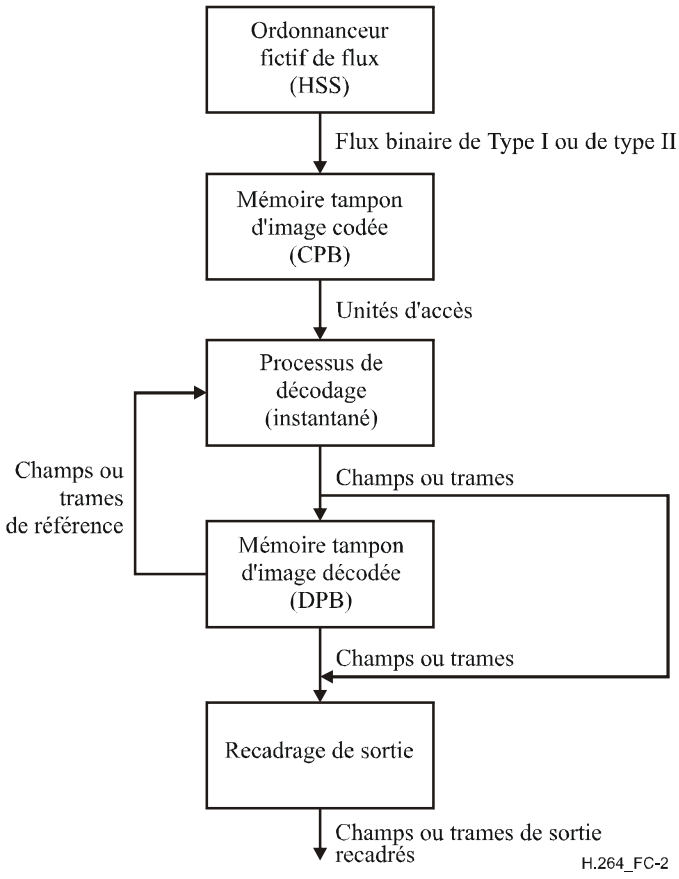


Figure C-2 – Modèle de mémoire tampon de décodeur HRD

La taille du tampon CPB (nombre de bits) est CpbSize[SchedSelIdx]. La taille du tampon DPB (nombre de tampons de trame) est Max (1, max_dec_frame_buffering).

Le décodeur HRD fonctionne comme suit. Les données associées aux unités d'accès qui s'écoulent dans le CPB conformément à un schéma d'arrivée spécifié sont livrées par l'ordonnanceur HSS. Les données associées à chaque unité d'accès sont retirées et décodées instantanément par le processus de décodage instantané aux instants de retrait du tampon CPB. Chaque image décodée est placée dans le tampon DPB à son moment de retrait à moins qu'elle ne soit sortie à son moment de retrait du tampon CPB et soit une image non référentielle. Lorsqu'une image est placée dans le tampon DPB, elle est retirée du tampon DPB au moment de son retrait du tampon DPB ou au moment où elle est marquée "inutilisée comme référence", selon le moment le plus tardif.

Le fonctionnement du tampon CPB est spécifié au § C.1. Le fonctionnement instantané du décodeur est spécifié aux § 8 et 9. Le fonctionnement du tampon DPB est spécifié au § C.2. Le recadrage du résultat est spécifié au § C.2.2.

Les informations d'ordonnanceur HSS et de décodeur HRD concernant le nombre de schémas de livraison énumérés et leurs débits binaires associés ainsi que les tailles de mémoires tampon sont spécifiées aux § E.1.1, E.1.2, E.2.1 et E.2.2.

le décodeur HRD est initialisé comme spécifié par le message SEI de période de mise en mémoire tampon comme spécifié aux § D.1.1 et D.2.1. Le moment de retrait des unités d'accès du tampon CPB et le moment de sortie du tampon DPB sont spécifiés au message SEI de temporisation d'image comme spécifié aux § D.1.2 et D.2.2. Toutes les informations de temporisation se rapportant à une unité d'accès spécifique doivent arriver avant le moment de retrait du tampon CPB de l'unité d'accès.

Le décodeur HRD est utilisé afin de vérifier la conformité des flux binaires et des décodeurs comme spécifié respectivement aux § C.3 et C.4.

NOTE 3 – Alors que la conformité est garantie dans l'hypothèse où tous les débits de trame et d'horloge utilisés afin de produire le flux binaire correspondent exactement aux valeurs signalées dans le flux binaire, dans un système réel, chacun d'eux peut différer de la valeur signalée ou spécifiée.

Toute l'arithmétique de la présente annexe est faite avec des valeurs réelles, de sorte qu'aucune erreur d'arrondi ne puisse se développer. Par exemple, le nombre de bits dans un CPB juste avant ou juste après le retrait d'une unité d'accès n'est pas nécessairement entier.

La variable t_c est déduite comme suit et est appelée un *tic-tac d'horloge*.

$$t_c = \text{num\_units\_in\_tick} \div \text{time\_scale} \tag{C-1}$$

Ce qui suit est spécifié afin d'exprimer les contraintes de la présente annexe.

– Soit l'unité d'accès n la $n^{\text{ième}}$ unité d'accès dans l'ordre de décodage, la première unité d'accès étant l'unité d'accès 0.

– Soit l'image n l'image codée primaire ou l'image primaire décodée de l'unité d'accès n.

C.1 Fonctionnement de la mémoire tampon d'image codée (CPB, *coded picture buffer*)

Les spécifications du présent paragraphe s'appliquent indépendamment à chaque ensemble de paramètres de CPB présent et aux points de conformité de Type I et de Type II représentés sur la Figure C-1.

C.1.1 Moment de l'arrivée du flux binaire

Le décodeur HRD peut être initialisé à tout message SEI de période de mise en mémoire tampon. Avant l'initialisation, le CPB est vide.

NOTE – Après l'initialisation, le décodeur HRD n'est pas réinitialisé par des messages SEI de période de mise en mémoire tampon ultérieurs.

Chaque unité d'accès est mentionnée comme unité d'accès n, où le numéro n identifie l'unité d'accès considérée. L'unité d'accès qui est associée au message SEI de période de mise en mémoire tampon qui initialise le CPB est mentionnée comme unité d'accès 0. La valeur de n est incrémentée de 1 pour chaque unité d'accès suivante dans l'ordre de décodage.

Le moment auquel le premier bit de l'unité d'accès n commence à entrer dans le CPB est désigné comme le moment d'arrivée initial $t_{ai}(n)$.

Le moment d'arrivée initial des unités d'accès est calculé comme suit:

– si l'unité d'accès est l'unité d'accès 0, $t_{ai}(0) = 0$,

– sinon (si l'unité d'accès est l'unité d'accès n avec $n > 0$), on applique ce qui suit:

  – si cbr_flag[SchedSelIdx] est égal à 1, le moment d'arrivée initial pour l'unité d'accès n est égal au moment d'arrivée final (qui est calculé ci-dessous) de l'unité d'accès n – 1, c'est-à-dire

$$t_{ai}(n) = t_{af}(n - 1) \tag{C-2}$$

  – sinon (si cbr_flag[SchedSelIdx] est égal à 0), le moment d'arrivée initial pour l'unité d'accès n est calculé par

$$t_{ai}(n) = \text{Max}(t_{af}(n - 1), t_{ai,earliest}(n)) \tag{C-3}$$

où $t_{ai,earliest}(n)$ est calculé comme suit

  – si l'unité d'accès n n'est pas la première unité d'accès d'une période de mise en mémoire tampon suivante, $t_{ai,earliest}(n)$ est calculé comme

$$t_{ai,earliest}(n) = t_{r,n}(n) - (\text{initial\_cpb\_removal\_delay}[\text{SchedSelIdx}] + \text{initial\_cpb\_removal\_delay\_offset}[\text{SchedSelIdx}]) \div 90000 \tag{C-4}$$

avec $t_{r,n}(n)$ qui est le moment de retrait nominal de l'unité d'accès n à partir du tampon CPB comme spécifié au § C.1.2 et initial_cpb_removal_delay[SchedSelIdx] et initial_cpb_removal_delay_offset[SchedSelIdx] qui sont spécifiés dans le message SEI de période de mise en mémoire tampon précédente.

– Sinon (si l'unité d'accès n suivante est la première unité d'accès d'une période de mise en mémoire tampon suivante), $t_{ai,earliest}(n)$ est calculé comme

$$t_{ai,earliest}(n) = t_{r,n}(n) – (\text{initial\_cpb\_removal\_delay}[\text{SchedSelIdx}] \div 90000) \quad \text{(C-5)}$$

avec initial_cpb_removal_delay[SchedSelIdx] qui est spécifié dans le message SEI de période de mise en mémoire tampon associé à l'unité d'accès n.

Le moment d'arrivée final pour l'unité d'accès n est calculé par

$$t_{af}(n) = t_{ai}(n) + b(n) \div \text{BitRate}[\text{SchedSelIdx}] \quad \text{(C-6)}$$

où b(n) est la taille en bits de l'unité d'accès n, en comptant les bits des unités NAL de couche VCL et des unités NAL de données de remplissage pour le point de conformité de Type I ou tous les bits de flux binaire de Type II pour le point de conformité de Type II, où les points de conformité de Type I et de Type II sont tels que représentés sur la Figure C-1.

Les valeurs de SchedSelIdx, BitRate[SchedSelIdx], et CpbSize[SchedSelIdx] sont soumises comme suit à des contraintes.

– Si l'unité d'accès n et l'unité d'accès n – 1 font partie de séquences vidéo codées différentes et que le contenu des ensembles actifs de paramètres de séquences des deux séquences vidéo codées diffère, l'ordonnanceur HSS choisit une valeur SchedSelIdx1 de SchedSelIdx parmi les valeurs de SchedSelIdx fournies pour la séquence vidéo codée qui contient l'unité d'accès n qui a pour résultat un BitRate[SchedSelIdx1] ou CpbSize[SchedSelIdx1] pour la seconde des deux séquences vidéo codées (qui contient l'unité d'accès n – 1) qui diffère de la valeur de BitRate[SchedSelIdx0] ou CpbSize[SchedSelIdx0] pour la valeur SchedSelIdx0 de SchedSelIdx qui était utilisée pour la séquence vidéo codée contenant l'unité d'accès –. La valeur de BitRate [SchedSelIdx11] ou CpbSize[SchedSelIdx1] peut différer de la valeur de BitRate[SchedSelIdx0] ou CpbSize[SchedSelIdx0] pour la valeur SchedSelIdx0 de SchedSelIdx qui était utilisée pour la séquence vidéo codée contenant l'unité d'accès n – 1.

– Sinon, l'ordonnanceur HSS continue de fonctionner avec les valeurs précédentes de SchedSelIdx, BitRate[SchedSelIdx] et CpbSize[SchedSelIdx].

Lorsque l'ordonnanceur HSS choisit des valeurs de BitRate[SchedSelIdx] ou CpbSize[SchedSelIdx] qui diffèrent de celles de l'unité d'accès précédente, on applique ce qui suit.

– la variable BitRate[SchedSelIdx] est mise en œuvre au temps $t_{ai}(n)$;

– la variable CpbSize[SchedSelIdx] est mise en œuvre comme suit.

  – Si la nouvelle valeur de CpbSize[SchedSelIdx] dépasse l'ancienne taille de CPB, elle est mise en œuvre au temps $t_{ai}(n)$.

  – Sinon, la nouvelle valeur de CpbSize[SchedSelIdx] est mise en œuvre au temps $t_r(n)$.

C.1.2 Moment du retrait de l'image codée

Pour l'unité d'accès 0, le moment de retrait nominal de l'unité d'accès du tampon CPB est spécifié par

$$t_{r,n}(0) = \text{initial\_cpb\_removal\_delay}[\text{SchedSelIdx}] \div 90000 \quad \text{(C-7)}$$

Pour la première unité d'accès d'une période de mise en mémoire tampon qui n'initialise pas le décodeur HRD, le moment de retrait nominal de l'unité d'accès du tampon CPB est spécifié par

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * \text{cpb\_removal\_delay}(n) \quad \text{(C-8)}$$

où $t_{r,n}(n_b)$ est le moment de retrait nominal de la première unité d'accès de la période précédente de mise en mémoire tampon et cpb_removal_delay(n) est la valeur de cpb_removal_delay spécifiée dans le message SEI de périodisation d'image associé à l'unité d'accès n.

Lorsqu'une unité d'accès n est la première unité d'accès d'une période de mise en mémoire tampon, n_b est mis égal à n au moment du retrait de l'unité d'accès n.

Le moment nominal de retrait $t_{r,n}(n)$ d'une unité d'accès n qui n'est pas la première unité d'accès d'une période de mise en mémoire tampon est donné par

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * \text{cpb\_removal\_delay}(n) \quad \text{(C-9)}$$

où $t_{r,n}(n_b)$ est le moment nominal de retrait de la première unité d'accès de la période de mise en mémoire tampon en cours et cpb_removal_delay(n) est la valeur de cpb_removal_delay spécifiée dans le message SEI de périodisation d'image associé à l'unité d'accès n.

Le moment de retrait d'une unité d'accès n est spécifié comme suit:

– Si low_delay_hrd_flag est égal à 0 ou $t_{r,n}(n) >= t_{af}(n)$, le moment de retrait de l'unité d'accès n est spécifié par

$$t_r(n) = t_{r,n}(n) \quad \text{(C-10)}$$

– Sinon (si low_delay_hrd_flag est égal à 1 et $t_{r,n}(n) < t_{af}(n)$), le moment de retrait de l'unité d'accès n est spécifié par

$$t_r(n) = t_{r,n}(n) + t_c * \text{Ceil}((t_{af}(n) - t_{r,n}(n)) \div t_c) \quad \text{(C-11)}$$

NOTE – Le dernier cas indique que la taille de l'unité d'accès n, b(n), est si grande qu'elle empêche le retrait au moment de retrait nominal.

C.2 Fonctionnement de la mémoire tampon de l'image décodée (DPB, *decoded picture buffer*)

La mémoire tampon de l'image décodée contient les mémoires tampon de trame. Chacune des mémoires tampon de trame peut contenir une trame décodée, une paire de sous-trames complémentaires décodées ou une seule (non appariée) sous-trame décodée qui sont marquées "utilisée comme référence" (images de référence) ou sont gardées pour une sortie future (images réordonnées ou retardées). Avant l'initialisation, le tampon DPB est vide (le remplissage du tampon DPB est mis à zéro). Les étapes suivantes des subdivisions du présent paragraphe surviennent toutes instantanément à $t_r(n)$ et selon la séquence indiquée.

C.2.1 Décodage des lacunes dans frame_num et mémorisation des trames "non existantes"

Le cas échéant, les lacunes dans frame_num sont détectées par le processus de décodage et les trames produites sont marquées et insérées dans le tampon DPB comme spécifié ci-dessous.

Les lacunes dans frame_num sont détectées par le processus de décodage et les trames produites sont marquées comme spécifié au § 8.2.5.2.

Après le marquage de chaque trame produite, chaque image m marquée par le processus de "fenêtre glissante" comme étant "inutilisée comme référence" est retirée du tampon DPB lorsqu'elle est aussi marquée "non existante" ou lorsque son moment de sortie du tampon DPB est inférieur ou égal au moment de retrait du tampon CPB de l'image en cours n; c'est-à-dire, $t_{o,dpb}(m) <= t_r(n)$. Lorsqu'une trame ou la dernière sous-trame d'une mémoire tampon de trame est retirée du tampon DPB, le remplissage du tampon DPB est décrémenté de un. La trame produite "non existante" est insérée dans le tampon DPB et le remplissage du tampon DPB est incrémenté de un.

C.2.2 Décodage et sortie d'image

L'image n est décodée et son temps de sortie du tampon DPB $t_{o,dpb}(n)$ est calculé par

$$t_{o,dpb}(n) = t_r(n) + t_c * \text{dpb\_output\_delay}(n) \quad \text{(C-12)}$$

La sortie de l'image en cours est spécifiée comme suit.

– Si $t_{o,dpb}(n) = t_r(n)$, l'image en cours est sortie.
NOTE – Lorsque l'image en cours est une image de référence, elle est mémorisée dans le tampon DPB.

– Sinon (si $t_{o,dpb}(n) > t_r(n)$), l'image en cours est sortie ultérieurement et sera mémorisée dans le tampon DPB (comme spécifié au § C.2.4) et sera sortie au moment $t_{o,dpb}(n)$ à moins qu'il ne soit indiqué de ne pas la sortir au décodage ou qu'il y ait une inférence de no_output_of_prior_pics_flag égal à 1 au moment qui précède $t_{o,dpb}(n)$.

L'image de sortie doit être recadrée, au moyen de la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence.

Lorsque l'image n est une image qui est sortie et n'est pas la dernière image du flux binaire qui est sorti, la valeur de $\Delta t_{o,dpb}(n)$ est définie comme:

$$\Delta t_{o,dpb}(n) = t_{o,dpb}(n_n) - t_{o,dpb}(n) \quad \text{(C-13)}$$

où n_n indique l'image qui suit après l'image n dans l'ordre de sortie.

L'image décodée est temporairement mémorisée (mais pas dans le tampon DPB).

C.2.3 Retrait des images du tampon DPB avant une possible insertion de l'image en cours

Le retrait des images du tampon DPB avant la possible insertion de l'image en cours s'effectue comme suit.

– Si l'image décodée est à rafraîchissement IDR, on applique ce qui suit.

  – Toutes les images de référence dans le tampon DPB sont marquées "inutilisée comme référence" comme spécifié au§ 8.2.5.1.

  – Lorsque l'image à rafraîchissement IDR n'est pas la premièreimage à rafraîchissement IDR décodée et que la valeur de PicWidthInMbs ou de FrameHeightInMbs ou de max_dec_frame_buffering déduite de l'ensemble actif de paramètres de séquence est différente respectivement de la valeur de PicWidthInMbs ou FrameHeightInMbs ou max_dec_frame_buffering déduite de l'ensemble de paramètres de séquence qui était actif pour la précédente séquence, no_output_of_prior_pics_flag est supposé être égal à 1 par le décodeur HRD, sans considération de la valeur réelle de no_output_of_prior_pics_flag.

    NOTE – Les mises en œuvre de décodeurs devraient essayer de traiter les changements de trame ou de taille du tampon DPB plus harmonieusement que le décodeur HRD par rapport aux changements dans PicWidthInMbs ou FrameHeightInMbs.

  – Lorsque no_output_of_prior_pics_flag est égal à 1 ou est supposé être égal à 1, toutes les mémoires tampon de trame contenues dans le tampon DPB sont vidées sans sortie des images qu'elles contiennent, et le remplissage du tampon DPB est mis à 0.

– Sinon (si l'image décodée n'est pas à rafraîchissement IDR), on applique ce qui suit.

  – Si l'en-tête de tranche de l'image en cours inclut memory_management_control_operation égal à 5, toutes les images de référence dans le tampon DPB sont marquées "inutilisée comme référence".

  – Sinon (si l'en-tête de tranche de l'image en cours n'inclut pas de memory_management_control_operation égal à 5), le processus de marquage d'image de référence décodée spécifié au § 8.2.5 est invoqué.

Toutes les images m contenues dans le tampon DPB, pour lesquelles toutes les conditions suivantes sont vraies, sont retirées du tampon DPB.

– L'image m est marquée "inutilisée comme référence" ou l'image m est une image non référentielle. Lorsqu'une image est une trame de référence, elle n'est considérée comme marquée "inutilisée comme référence" que lorsque ses deux sous-trames ont été marquées "inutilisée comme référence".

– L'image m est marquée "non existante" ou son moment de sortie du tampon DPB est inférieur ou égal au moment de retrait du tampon CPB de l'image en cours n; c'est-à-dire que $t_{o,dpb}(m) <= t_r(n)$.

Lorsqu'une trame ou la dernière sous-trame dans une mémoire tampon de trame est retirée du tampon DPB, le remplissage du tampon DPB est décrémenté de un.

C.2.4 Marquage et mémorisation de l'image décodée en cours

C.2.4.1 Marquage et mémorisation d'une image décodée de référence dans le tampon DPB

Lorsque l'image en cours est une image de référence, elle est mémorisée dans le tampon DPB comme suit.

– Si l'image décodée en cours est une seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames de référence complémentaires, et que la première sous-trame de la paire soit toujours dans le tampon DPB, l'image décodée en cours est mémorisée dans la même mémoire tampon de trame que la première sous-trame de la paire.

– Sinon, l'image décodée en cours est mémorisée dans une mémoire tampon de trame vide, et le remplissage du tampon DPB est incrémenté de un.

C.2.4.2 Mémorisation d'une image non référentielle dans le tampon DPB

Lorsque l'image en cours est une image non référentielle et que l'image n en cours a $t_{o,dpb}(n) > t_r(n)$, elle est mémorisée dans le tampon DPB comme suit.

– Si l'image décodée en cours est une seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames non référentielles complémentaires, et que la première sous-trame de la paire soit encore dans le tampon DPB, l'image décodée en cours est mémorisée dans la même mémoire tampon de trame que la première sous-trame de la paire.

– Sinon, l'image décodée en cours est mémorisée dans une mémoire tampon de trame vide, et le remplissage du tampon DPB est incrémenté de un.

C.3 Conformité de flux binaire

Un flux binaire de données codées conforme à la présente Recommandation | Norme internationale doit satisfaire aux exigences suivantes.

Le flux binaire est construit conformément à la syntaxe, à la sémantique, et aux contraintes spécifiées dans la présente Recommandation | Norme internationale hors de la présente annexe.

Le flux binaire est testé par le décodeur HRD comme spécifié ci-dessous:

pour les flux binaires de Type I, le nombre d'essais effectués est égal à cpb_cnt_minus1 + 1, où cpb_cnt_minus1 est l'élément syntaxique de hrd_parameters() suivant le vcl_hrd_parameters_present_flag ou est déterminé par l'application par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale. Un essai est effectué pour chaque combinaison de débit binaire et de taille de CPB spécifiée par les hrd_parameters( ) suivant le vcl_hrd_parameters_present_flag. Chacun de ces essais est effectué au niveau du point de conformité de Type I représenté sur la Figure C-1.

Pour les flux binaires de Type II, il y a deux ensembles d'essais. Le nombre d'essais du premier ensemble est égal à cpb_cnt_minus1 + 1 où cpb_cnt_minus1 est l'élément syntaxique de hrd_parameters( ) suivant le vcl_hrd_parameters_present_flag ou est déterminé par l'application par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale. Un essai est effectué pour chaque combinaison de débit binaire et de taille de CPB. Chacun de ces essais est effectué au niveau du point de conformité de Type I représenté sur la Figure C-1. Pour ces essais, seules les unités NAL de couche VCL et de données de remplissage sont comptées pour le débit binaire d'entrée et de mémorisation du tampon CPB.

Le nombre d'essais du second ensemble, pour les flux binaires de Type II, est égal à cpb_cnt_minus1 + 1, où cpb_cnt_minus1 est l'élément syntaxique de hrd_parameters() suivant le nal_hrd_parameters_present_flag ou est déterminé par l'application par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale. Un essai est effectué pour chaque combinaison de débit binaire et de taille de CPB spécifiée par les hrd_parameters() suivant le nal_hrd_parameters_present_flag. Chacun de ces essais est effectué au niveau du point de conformité de Type II représenté sur la Figure C-1. Pour ces essais, toutes les unités NAL (d'un flux d'unités NAL de Type II) ou tous les octets (d'un flux d'octets) sont comptés pour le débit binaire d'entrée et de mémorisation du tampon CPB.

NOTE 1 – Les paramètres NAL HRD établis par une valeur de SchedSelIDx pour les points de conformité de Type II représentés sur la Figure C-1 sont suffisants afin d'établir également la conformité VCL HRD pour le point de conformité de Type I représenté sur la Figure C-1 pour les mêmes valeurs de initial_cpb_removal_delay[SchedSelIdx], BitRate[SchedSelIdx] et CpbSize[SchedSelIdx] pour le cas VBR (cbr_flag[SchedSelIdx] égal à 0). Cela s'explique par le fait que le flux de données dans le point de conformité de Type I est un sous-ensemble du flux de données dans le point de conformité de Type II et par le fait que, pour le cas VBR, le CPB est autorisé à se vider et à rester vide jusqu'au moment où une nouvelle image devrait normalement commencer à arriver. Par exemple, lorsque les paramètres NAL HRD fournis pour le point de conformité de Type II entrent non seulement dans les limites de conformité de profil fixées pour ces paramètres au point j) du § A.3.1 ou au point h) du § A.3.3 (selon le profil utilisé), mais aussi dans les limites de conformité de profil fixées pour les paramètres VCL HRD au point i) du § A.3.1 ou au point h) du § A.3.3 (selon le profil utilisé), on peut également être assuré que les paramètres VCL HRD pour le point de conformité de Type I entreront dans les limites de conformité indiquées au point i) du § A.3.1.

Pour les flux binaires conformes, toutes les conditions suivantes doivent être satisfaites pour chacun des essais.

– Pour chaque unité d'accès n, avec n>0, associée à un message SEI de période de mise en mémoire tampon, avec $\Delta t_{g,90}(n)$ spécifié par

$$\Delta t_{g,90}(n) = 90000 * (t_{r,n}(n) - t_{af}(n - 1)) \quad \text{(C-14)}$$

la valeur de initial_cpb_removal_delay[SchedSelIdx] doit respecter les contraintes suivantes:

– Si cbr_flag[SchedSelIdx] est égal à 0,

$$\text{initial\_cpb\_removal\_delay[SchedSelIdx]} <= \text{Ceil}(\Delta t_{g,90}(n)) \quad \text{(C-15)}$$

– Sinon (si cbr_flag[SchedSelIdx] est égal à 1),

$$\text{Floor}(\Delta t_{g,90}(n)) <= \text{initial\_cpb\_removal\_delay[SchedSelIdx]} <= \text{Ceil}(\Delta t_{g,90}(n)) \quad \text{(C-16)}$$

NOTE 2 – Le nombre exact de bits dans le CPB au moment de retrait de chaque image peut dépendre du message SEI de période de mise en mémoire tampon choisi afin d'initialiser le décodeur HRD. Les codeurs doivent en tenir compte afin de faire en sorte que toutes les contraintes spécifiées soient respectées quel que soit le message SEI de période de mise en mémoire tampon choisi afin d'initialiser le décodeur HRD, car le décodeur HRD peut être initialisé à chacun des messages SEI de période de mise en mémoire tampon.

– Le débordement d'un CPB est spécifié comme l'état dans lequel le nombre total de bits dans le CPB est supérieur à la taille de celui-ci. Le CPB ne doit jamais déborder.

– Le sous-remplissage d'un CPB est spécifié comme l'état dans lequel tr,n(n) est inférieur à taf(n). Lorsque low_delay_hrd_flag est égal à 0, le CPB ne doit jamais être sous-rempli.

– Le moment nominal de retrait des images du tampon CPB (commençant à la seconde image dans l'ordre de décodage), doit satisfaire aux contraintes sur tr,n(n) et tr(n) exprimées aux § A.3.1 à A.3.3 pour le profil et le niveau spécifiés dans le flux binaire.

– Immédiatement après que toute image décodée est ajoutée au DPB, le remplissage du tampon DPB doit être inférieur ou égal à la taille DPB telle qu'elle est spécifiée par les Annexes A, D, et E pour le profil et le niveau spécifiés dans le flux binaire.

– Toutes les images de référence doivent être présentes dans le tampon DPB lorsqu'elles sont nécessaires pour la prédiction. Chaque image doit être présente dans le tampon DPB à son moment de sortie du tampon DPB à moins qu'elle ne soit pas mémorisée du tout dans le tampon DPB, ou ne soit retirée du tampon DPB avant son moment de sortie par un des processus spécifiés au § C.2.

– La valeur de $\Delta$to,dpb(n) telle que donnée par l'équation C-13, qui est la différence entre le moment de sortie d'une image et celui de l'image la suivant immédiatement dans l'ordre de sortie, doit satisfaire à la contrainte exprimée au § A.3.1 pour le profil et le niveau spécifiés dans le flux binaire.

C.4 Conformité du décodeur

Un décodeur conforme à la présente Recommandation | Norme internationale doit satisfaire aux exigences suivantes.

Un décodeur prétendant à la conformité à un profil et niveau spécifiques doit être capable de décoder avec succès tous les flux binaires conformes spécifiés pour la conformité du décodeur au § C.3, pourvu que les ensembles de paramètres de séquence, les ensembles de paramètres d'image mentionnés dans les unités NAL de couche VCL et les messages SEI appropriés de période de mise en mémoire tampon et de synchronisation d'image soient convoyés au décodeur, en temps utile, dans le flux binaire (par des unités NAL non-VCL), ou par des moyens externes non spécifiés par la présente Recommandation | Norme internationale.

Il y a deux types de conformité qui peuvent être revendiqués par un décodeur: une conformité de synchronisation de sortie et une conformité d'ordre de sortie.

Afin de vérifier la conformité d'un décodeur, les essais de flux binaires conformes au profil et niveau revendiqué, comme spécifié au § C.3 sont fournis par un ordonnanceur fictif de flux (HSS, *hypothetical stream scheduler*) à la fois au décodeur HRD et au décodeur soumis à l'essai (DUT, *decoder under test*). Toutes les images sorties par le décodeur HRD doivent aussi être sorties par le décodeur DUT et, pour chaque image sortie par le décodeur HRD, les valeurs de tous les échantillons qui sont sortis par le décodeur DUT pour l'image correspondante doivent être égales aux valeurs des échantillons sortis par le décodeur HRD.

Pour la conformité de synchronisation de sortie du décodeur, l'ordonnanceur HSS fonctionne comme décrit ci-dessus, avec des schémas de livraison choisis uniquement parmi le sous-ensemble de valeurs de SchedSelIdx pour lequel le débit binaire et la taille de CPB sont soumis à des restrictions spécifiées à l'Annexe A pour le profil et niveau spécifié, ou avec des schémas de livraison "interpolés" tels que spécifiés ci-dessous pour lesquels le débit binaire et la taille de CPB sont soumises à des restrictions spécifiées à l'Annexe A. Le même schéma de livraison est utilisé pour le décodeur HRD et le décodeur DUT.

Lorsque les paramètres de décodeur HRD et les messages SEI de période de mise en mémoire tampon sont présents avec cpb_cnt_minus1 supérieur à 0, le décodeur doit être capable de décoder le flux binaire tel que livré du HSS en fonctionnant à l'aide du schéma de livraison "interpolé" spécifié comme ayant un débit binaire de pointe r, une taille de CPB c(r), et un délai initial de retrait de CPB (f(r) ÷ r) comme suit

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]), \quad \text{(C-17)}$$

$$c(r) = \alpha * \text{CpbSize}[\text{SchedSelIdx}] + (1 - \alpha) * \text{CpbSize}[\text{SchedSelIdx}-1], \quad \text{(C-18)}$$

$$f(r) = \alpha * initial\_cpb\_removal\_delay[SchedSelIdx] * BitRate[SchedSelIdx] + (1 - \alpha) * initial\_cpb\_removal\_delay[SchedSelIdx - 1] * BitRate[SchedSelIdx - 1] \quad \text{(C-19)}$$

pour tout SchedSelIdx > 0 et r tels que BitRate[SchedSelIdx – 1] <= r <= BitRate[SchedSelIdx] tel que r et c(r) soient au sein des limites spécifiées à l'Annexe A pour le débit binaire et la taille de mémoire tampon maximal pour les profil et niveau spécifiés.

NOTE 1 – L'élément initial_cpb_removal_delay[SchedSelIdx] peut être différent d'une période de mise en mémoire tampon à une autre et peut devoir être recalculé.

Pour la conformité de synchronisation de sortie du décodeur, on utilise un décodeur HRD comme décrit ci-dessus et la synchronisation (par rapport au moment de livraison du premier bit) de la sortie d'image est la même pour le décodeur HRD et le décodeur DUT jusqu'à l'écoulement d'un délai fixé.

Pour la conformité de l'ordre de sortie du décodeur, l'ordonnanceur HSS fournit le flux binaire au DUT "à la demande" du décodeur DUT, ce qui signifie que l'ordonnanceur HSS ne délivre les bits (dans l'ordre de décodage) que lorsque le décodeur DUT a besoin de plus de bits afin de poursuivre son traitement.

NOTE 2 – Cela signifie que, pour cet essai, la mémoire tampon de l'image codée du décodeur DUT pourrait être aussi petite que la taille de la plus grande unité d'accès.

On utilise un décodeur HRD modifié tel que décrit ci-dessous, et l'ordonnanceur HSS délivre le flux binaire au décodeur HRD selon un des schémas spécifiés dans le flux binaire tel que le débit binaire et la taille du tampon CPB suivent les limites spécifiées à l'Annexe A. L'ordre de sortie des images doit être le même pour le décodeur HRD et le décodeur DUT.

Pour la conformité de l'ordre de sortie du décodeur, la taille du tampon CPB HRD est égale à CpbSize[SchedSelIdx] pour le schéma choisi et la taille du tampon DPB est égale à MaxDpbSize. Le moment de retrait du tampon CPB pour le décodeur HRD est égal au moment final d'arrivée de bit et le décodage est immédiat. Le fonctionnement du tampon DPB de ce décodeur HRD est décrit ci-dessous.

C.4.1 Fonctionnement du tampon DPB d'ordres de sortie

La mémoire tampon de l'image décodée contient des mémoires tampon de trame. Chacune des mémoires tampon de trame peut contenir une trame décodée, une paire de sous-trames complémentaires décodées ou une seule (non appariée) sous-trame décodée qui est marquée "utilisée comme référence" ou est gardée pour une sortie future (images réordonnées). A l'initialisation du décodeur HRD, le remplissage du tampon DPB, mesuré en trames, est mis à 0. Les étapes suivantes surviennent toutes instantanément lorsqu'une unité d'accès est retirée du tampon CPB, et dans l'ordre figurant sur la liste.

C.4.2 Décodage des lacunes dans frame_num et mémorisation d'images "non existantes"

Le cas échéant, les lacunes dans frame_num sont détectés par le processus de décodage et le nombre nécessaire de trames "non existante" sont déduites dans l'ordre spécifié par la production des valeurs de UnusedShortTermFrameNum dans l'équation 7-21 et marquées comme spécifié au § 8.2.5.2. Les tampons de trames contenant une trame ou une paire de sous-trames complémentaires ou une sous-trame non appariée qui est (sont) marquée(s) "non nécessaire pour sortie" et "inutilisée comme référence" sont vidés (sans sortie), et le remplissage du tampon DPB est décrémenté par le nombre de tampons de trames vidés.Chaque trame "non existante" est mémorisée dans le tampon DPB comme suit:

- Lorsqu'il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage du tampon DPB est égal à la taille du tampon DPB), le processus de "supplantation" spécifié au § C.4.5.3 est invoqué de façon répétée jusqu'à ce que l'on trouve une mémoire tampon de trame vide dans laquelle mémoriser la trame "non existante".
- La trame "non existante" est mémorisée dans une mémoire tampon de trame vide et est marquée "non nécessaire pour sortie" et le remplissage du tampon DPB est incrémenté de un.

C.4.3 Décodage d'image

L'image codée primaire n est décodée et est mémorisée temporairement (mais pas dans le tampon DPB).

C.4.4 Retrait d'images du tampon DPB avant l'insertion possible de l'image en cours

Le retrait des images du tampon DPB avant que ne soit possible l'insertion de l'image en cours s'effectue comme suit:

- Si l'image décodée est à rafraîchissement IDR, on applique ce qui suit.
  - Toutes les images de référence dans le tampon DPB sont marquées "inutilisée comme référence" comme spécifié au § 8.2.5.

– Lorsque l'image à rafraîchissement IDR n'est pas la premièreimage à rafraîchissement IDR décodée et que la valeur de PicWidthInMbs (*largeur d'image en macroblocs*) ou FrameHeightInMbs (*hauteur de trame en macroblocs*) ou max_dec_frame_buffering (*maximum de mémoire tampon de trame décodée*) déduite de l'ensemble actif de paramètres de séquence est différent de la valeur de, respectivement, PicWidthInMbs ou FrameHeightInMbs ou max_dec_frame_buffering, déduite de l'ensemble de paramètres de séquence qui était actif pour la séquence précédente, no_output_of_prior_pics_flag (*fanion pas de sortie d'images précédentes*) est supposé être égal à 1 par le décodeur HRD, sans considération de la valeur réelle de no_output_of_prior_pics_flag.

  NOTE – Les mises en œuvre de décodeurs devraient essayer de traiter plus harmonieusement que le décodeur HRD les changements de valeur de PicWidthInMbs ou FrameHeightInMbs ou max_dec_frame_buffering.

– Lorsque le fanion no_output_of_prior_pics_flag est égal à 1 ou est supposé être égal à 1, toutes les mémoires tampon de trame contenues dans le tampon DPB sont vidées sans sortie des images qu'elles contiennent, et le remplissage du tampon DPB est mis à 0.

– Sinon (si l'image décodée n'est pas à rafraîchissement IDR), le processus de marquage d'image de référence décodée est invoqué comme spécifié au § 8.2.5. Les mémoires tampon de trame contenant une trame ou une paire de sous-trames complémentaires ou une sous-trame non appariée marquées "non nécessaire pour sortie" et "inutilisée comme référence" sont vidées (sans sortie) et le remplissage du tampon DPB est décrémenté du nombre de mémoires tampon de trame vidées.

Lorsque l'image en cours contient un élément memory_management_control_operation égal à 5 ou est à rafraîchissement IDRpour laquelle le fanion no_output_of_prior_pics_flag n'est pas égal à 1 et n'est pas supposé être égal à 1,les deux étapes suivantes sont exécutées,

1. Les tampons de trames contenant une trame ou une paire de sous-trames complémentaires ou une sous-trame non appariée qui est (sont) marquée(s) "non nécessaire pour sortie" et "inutilisée comme référence" sont vidés (sans sortie), et le remplissage du tampon DPB est décrémenté par le nombre de tampons de trames vidés.

2. Toutes les mémoires tampon de trame qui ne sont pas vides dans le tampon DPB sont vidées en invoquant de façon répétée le processus de "supplantation" spécifié au § C.4.5.3, et le remplissage du tampon DPB est mis à 0.

C.4.5 Marquage et mémorisation de l'image décodée en cours

C.4.5.1 Mémorisation et marquage de l'image décodée de référence dans le tampon DPB

Lorsque l'image en cours est une image de référence, elle est mémorisée dans le tampon DPB comme suit.

– Si l'image décodée en cours est la seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames de référence complémentaires, et que la première sous-trame de la paire soit toujours dans le tampon DPB, l'image en cours est mémorisée dans la même mémoire tampon de trame que la première sous-trame de la paire et est marquée "nécessaire pour sortie".

– Sinon, les opérations suivantes sont effectuées:

  – Lorsqu'il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage du tampon DPB est égal à la taille du tampon DPB), le processus de "supplantation" spécifié au § C.4.5.3 est invoqué de façon répétée jusqu'à ce que l'on trouve une mémoire tampon de trame vide dans laquelle mémoriser l'image décodée en cours.

  – L'image décodée en cours est mémorisée par une mémoire tampon de trame vide et est marquée "nécessaire pour sortie", et le remplissage du tampon DPB est incrémenté de un.

C.4.5.2 Mémorisation et marquage d'une image décodée non référentielle dans le tampon DPB

Lorsque l'image en cours est une image non référentielle, les opérations suivantes sont effectuées.

– Si l'image décodée en cours est la seconde sous-trame (dans l'ordre de décodage) d'une paire de sous-trames non référentielles complémentaires, et que la première sous-trame de la paire soit toujours dans le tampon DPB, l'image en cours est mémorisée dans la même mémoire tampon de trame que la première sous-trame de la paire et est marquée "nécessaire pour sortie".

– Sinon, les opérations suivantes sont effectuées de façon répétée jusqu'à ce que l'image décodée en cours soit recadrée et sortie ou qu'elle soit mémorisée dans le tampon DPB:

  – s'il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage du tampon DPB est égal à la taille du tampon DPB), on applique ce qui suit.

- Si l'image en cours n'a pas une valeur de PicOrderCnt()inférieure à celle de toutes les images du tampon DPB qui sont marquées "nécessaire pour sortie", le processus de "supplantation" spécifié au § C.4.5.3 est mis en œuvre.
- Sinon (si l'image en cours a une valeur de PicOrderCnt()inférieure à celle de toutes les images du tampon DPB qui sont marquées "nécessaire pour sortie"), l'image en cours est recadrée, au moyen de la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence et l'image recadrée est sortie.

- Sinon (s'il y a une mémoire tampon de trame vide, c'est-à-dire que le remplissage du tampon DPB est inférieur à la taille du tampon DPB) l'image décodée en cours est mémorisée dans une mémoire tampon de trame vide et est marquée "nécessaire pour sortie", et le remplissage du tampon DPB est incrémenté de un.

C.4.5.3 Processus de "supplantation"

Le processus de "supplantation"est invoqué dans les cas suivants:

- Il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage du tampon DPB est égal à la taille du tampon DPB) et une mémoire tampon de trame vide est nécessaire afin de mémoriser une trame "non existante" déduite, comme spécifié au § C.4.2.
- L'image en cours est à rafraîchissement IDR et le fanion no_output_of_prior_pics_flag n'est pas égal à 1 et n'est pas supposé être égale à 1, comme indiqué au § C.4.4.
- L'image en cours a memory_management_control_operation égal à 5, comme spécifié au § C.4.4.
- Il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage du tampon DPB est égal à la taille du tampon DPB) et une mémoire tampon de trame vide est nécessaire afin de mémoriser une image de référence (non IDR) décodée, comme spécifié au § C.4.5.1.
- Il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage du tampon DPB est égal à la taille du tampon DPB) et l'image en cours est une image non référentielle qui n'est pas le second sous-trame d'une paire de sous-trames non référentielles complémentaires et il y a des images dans le tampon DPB qui sont marquées "nécessaire pour sortie" qui précèdent l'image non référentielle en cours dans l'ordre de sortie, comme spécifié au § C.4.5.2, ce qui rend nécessaire une mémoire tampon vide afin de mémoriser l'image en cours.

Le processus de "supplantation" consiste en ce qui suit:

- l'image ou la paire de sous-trames de référence complémentaires qui arrive en premier pour la sortie est sélectionnée comme suit.
  - On sélectionne la mémoire tampon de trame qui contient l'image ayant la plus faible valeur de PicOrderCnt() de toutes les images du tampon DPB marquées "nécessaire pour sortie".
  - Si cette mémoire tampon de trame contient une paire de sous-trames non référentielles complémentaires avec les deux sous-trames marquées "nécessaire pour sortie" et que les deux sous-trames aient le même PicOrderCnt(), la première des deux sous-trames dans l'ordre de décodage est considérée comme étant la première pour la sortie.
  - Sinon, si cette mémoire tampon de trame contient une paire de sous-trames de référence complémentaires avec les deux sous-trames marquées "nécessaire pour sortie" et que les deux sous-trames aient le même PicOrderCnt(), la paire de sous-trames de référence complémentaires entière est considérée comme la première pour la sortie.
  - Sinon, l'image dans cette mémoire tampon de trame qui a la plus faible valeur de PicOrderCnt() est considérée comme la première pour la sortie.
- Si une image unique est considérée comme la première pour la sortie, cette image est recadrée, au moyen de la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence, l'image recadrée est sortie, et l'image est marquée "non nécessaire pour sortie".
- Sinon (si une paire de sous-trames de référence complémentaires est considérée comme étant la première pour la sortie), les deux sous-trames de la paire de sous-trames de référence complémentaires sont recadrées, au moyen de la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence, les deux sous-trames de la paire de sous-trames de référence complémentaires sont sorties ensemble, et les deux sous-trames de la paire de sous-trames de référence complémentaires sont marquées "non nécessaire pour sortie".
- La mémoire tampon de trame qui incluait l'image ou la paire de sous-trames de référence complémentaires qui a été recadrée et sortie est vérifiée, et lorsque aucune des conditions suivantes n'est satisfaite, la mémoire tampon de trame est vidée et le remplissage du tampon DPB est décrémenté de 1.
  - La mémoire tampon de trame contient un sous-trame non référentielle non appariée.

– La mémoire tampon de trame contient une trame non référentielle.

– La mémoire tampon de trame contient une paire de sous-trames non référentielles complémentaires avec les deux sous-trames marquées "non nécessaire pour sortie".

– La mémoire tampon de trame contient une sous-trame de référence non appariée, marquée "inutilisée comme référence".

– La mémoire tampon de trame contient une trame de référence avec les deux sous-trames marquées "inutilisée comme référence".

– La mémoire tampon de trame contient une paire de sous-trames de référence complémentaires avec les deux sous-trames marquées "inutilisée comme référence" et "non nécessaire pour sortie".

Annexe D

Informations d'amélioration supplémentaires

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie la syntaxe et la sémantique pour les charges utiles de message SEI.

Les messages SEI aident aux processus se rapportant au décodage, à l'affichage et à d'autres fins. Cependant, les messages SEI ne sont pas nécessaires afin de construire les échantillons luma ou chroma par le processus de décodage. Les décodeurs conformes ne sont pas tenus de traiter ces informations pour la conformité de l'ordre de sortie à la présente Recommandation | Norme internationale (voir l'Annexe C pour la spécification de la conformité). Certaines informations de messages SEI sont nécessaires afin de vérifier la conformité du flux binaire et celle du délai de sortie du décodeur.

A l'Annexe D, les spécifications de présence des messages SEI sont aussi satisfaites lorsque ces messages (ou certains de leurs sous-ensembles) sont convoyés aux décodeurs (ou au décodeur HRD) par d'autres moyens non spécifiés par la présente Recommandation | Norme internationale. Lorsqu'ils sont présents dans le flux binaire, les messages SEI doivent respecter la syntaxe et la sémantique spécifiée aux § 7.3.2.3 et 7.4.2.3 et dans la présente annexe. Lorsque le contenu d'un message SEI est convoyé afin d'être appliqué par des moyens autres que la présence au sein du flux binaire, la représentation du contenu du message SEI n'est pas nécessaire pour l'utilisation de la même syntaxe que spécifiée dans la présente annexe. Pour les besoins du comptage des bits, seuls les bits appropriés qui sont réellement présents dans le flux binaire sont comptés.

D.1 Syntaxe de la charge utile SEI

sei_payload(payloadType, payloadSize) {	C	Descripteur
si(payloadType == 0)		
buffering_period(payloadSize)	5	
ou si(payloadType == 1)		
pic_timing(payloadSize)	5	
ou si(payloadType == 2)		
pan_scan_rect(payloadSize)	5	
ou si(payloadType == 3)		
filler_payload(payloadSize)	5	
ou si(payloadType == 4)		
user_data_registered_itu_t_t35(payloadSize)	5	
ou si(payloadType == 5)		
user_data_unregistered(payloadSize)	5	
ou si(payloadType == 6)		
recovery_point(payloadSize)	5	
ou si(payloadType == 7)		
dec_ref_pic_marking_repetition(payloadSize)	5	
ou si(payloadType == 8)		
spare_pic(payloadSize)	5	
ou si(payloadType == 9)		
scene_info(payloadSize)	5	
ou si(payloadType == 10)		
sub_seq_info(payloadSize)	5	
ou si(payloadType == 11)		
sub_seq_layer_characteristics(payloadSize)	5	
ou si(payloadType == 12)		
sub_seq_characteristics(payloadSize)	5	
ou si(payloadType == 13)		

full_frame_freeze(payloadSize)	5	
ou si(payloadType == 14)		
full_frame_freeze_release(payloadSize)	5	
ou si(payloadType == 15)		
full_frame_snapshot(payloadSize)	5	
ou si(payloadType == 16)		
progressive_refinement_segment_start(payloadSize)	5	
ou si(payloadType == 17)		
progressive_refinement_segment_end(payloadSize)	5	
ou si(payloadType == 18)		
motion_constrained_slice_group_set(payloadSize)	5	
ou si(payloadType == 19)		
film_grain_characteristics(payloadSize)	5	
ou si(payloadType == 20)		
deblocking_filter_display_preference(payloadSize)	5	
ou si(payloadType == 21)		
stereo_video_info(payloadSize)	5	
ou		
reserved_sei_message(payloadSize)	5	
si(!byte_aligned( )) {		
bit_equal_to_one /* equal to 1 */	5	f(1)
tant que(!byte_aligned( ))		
bit_equal_to_zero /* equal to 0 */	5	f(1)
}		
}		

D.1.1 Syntaxe de message SEI de période de mise en mémoire tampon

buffering_period(payloadSize) {	**C**	**Descripteur**
seq_parameter_set_id	5	ue(v)
si(NalHrdBpPresentFlag) {		
pour(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay[SchedSelIdx **]**	5	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx **]**	5	u(v)
}		
}		
si(VclHrdBpPresentFlag) {		
pour(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay[SchedSelIdx **]**	5	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx **]**	5	u(v)
}		
}		
}		

D.1.2 Syntaxe de message SEI de synchronisation d'image

pic_timing(payloadSize) {	C	Descripteur
si(CpbDpbDelaysPresentFlag) {		
cpb_removal_delay	5	u(v)
dpb_output_delay	5	u(v)
}		
si(pic_struct_present_flag) {		
pic_struct	5	u(4)
pour(i = 0; i < NumClockTS ; i++) {		
clock_timestamp_flag[i]	5	u(1)
si(clock_timestamp_flag[i]) {		
ct_type	5	u(2)
nuit_field_based_flag	5	u(1)
counting_type	5	u(5)
full_timestamp_flag	5	u(1)
discontinuity_flag	5	u(1)
cnt_dropped_flag	5	u(1)
n_frames	5	u(8)
si(full_timestamp_flag) {		
seconds_value /* 0..59 */	5	u(6)
minutes_value /* 0..59 */	5	u(6)
hours_value /* 0..23 */	5	u(5)
} ou {		
seconds_flag	5	u(1)
si(seconds_flag) {		
seconds_value /* gamme 0..59 */	5	u(6)
minutes_flag	5	u(1)
si(minutes_flag) {		
minutes_value /* 0..59 */	5	u(6)
hours_flag	5	u(1)
si(hours_flag)		
hours_value /* 0..23 */	5	u(5)
}		
}		
}		
si(time_offset_length > 0)		
time_offset	5	i(v)
}		
}		
}		
}		

D.1.3 Syntaxe de message SEI de rectangle de balayage complet

pan_scan_rect(payloadSize) {	C	Descripteur
pan_scan_rect_id	5	ue(v)
pan_scan_rect_cancel_flag	5	u(1)
si(!pan_scan_rect_cancel_flag) {		
pan_scan_cnt_minus1	5	ue(v)
pour(i = 0; i <= pan_scan_cnt_minus1; i++) {		
pan_scan_rect_left_offset[i **]**	5	se(v)
pan_scan_rect_right_offset[i **]**	5	se(v)
pan_scan_rect_top_offset[i **]**	5	se(v)
pan_scan_rect_bottom_offset[i **]**	5	se(v)
}		
pan_scan_rect_repetition_period	5	ue(v)
}		
}		

D.1.4 Syntaxe de message SEI de charge utile de remplissage

filler_payload(payloadSize) {	C	Descripteur
pour(k = 0; k < payloadSize; k++)		
ff_byte /* égal à to 0xFF */	5	f(8)
}		

D.1.5 Syntaxe de message SEI de données d'usager enregistrées par la Rec. UIT-T T.35

user_data_registered_itu_t_t35(payloadSize) {	C	Descripteur
itu_t_t35_country_code	5	b(8)
si(itu_t_t35_country_code != 0xFF)		
i = 1		
ou {		
itu_t_t35_country_code_extension_byte	5	b(8)
i = 2		
}		
faire {		
itu_t_t35_payload_byte	5	b(8)
i++		
} tant que(i < payloadSize)		
}		

D.1.6 Syntaxe de message SEI de données d'usager non enregistrées

user_data_unregistered(payloadSize) {	C	Descripteur
uuid_iso_iec_11578	**5**	u(128)
pour(i = 16; i < payloadSize; i++)		
user_data_payload_byte	5	b(8)
}		

D.1.7 Syntaxe de message SEI de point de récupération

recovery_point(payloadSize) {	C	Descripteur
recovery_frame_cnt	5	ue(v)
exact_match_flag	5	u(1)
broken_link_flag	5	u(1)
changing_slice_group_idc	5	u(2)
}		

D.1.8 Syntaxe de message SEI de marquage d'image de référence décodée

dec_ref_pic_marking_repetition(payloadSize) {	C	Descripteur
original_idr_flag	5	u(1)
original_frame_num	5	ue(v)
si(!frame_mbs_only_flag) {		
original_field_pic_flag	5	u(1)
si(original_field_pic_flag)		
original_bottom_field_flag	5	u(1)
}		
dec_ref_pic_marking( )	5	
}		

D.1.9 Syntaxe de message SEI d'image de rechange

spare_pic(payloadSize) {	C	Descripteur
target_frame_num	5	ue(v)
spare_field_flag	5	u(1)
si(spare_field_flag)		
target_bottom_field_flag	5	u(1)
num_spare_pics_minus1	5	ue(v)
pour(i = 0; i < num_spare_pics_minus1 + 1; i++) {		
delta_spare_frame_num[i **]**	5	ue(v)
si(spare_field_flag)		
spare_bottom_field_flag[i **]**	5	u(1)
spare_area_idc[i **]**	5	ue(v)
si(spare_area_idc[i] == 1)		
pour(j = 0; j < PicSizeInMapUnits; j++)		
spare_unit_flag[i **][** j **]**	5	u(1)
ou si(spare_area_idc[i] == 2) {		
mapUnitCnt = 0		
pour(j=0; mapUnitCnt < PicSizeInMapUnits; j++) {		
zero_run_length[i **][** j **]**	5	ue(v)
mapUnitCnt += zero_run_length[i][j] + 1		
}		
}		
}		
}		

D.1.10 Syntaxe de message SEI d'informations de scène

scene_info(payloadSize) {	C	Descripteur
scene_info_present_flag	5	u(1)
si(scene_info_present_flag) {		
scene_id	5	ue(v)
scene_transition_type	5	ue(v)
si(scene_transition_type > 3)		
second_scene_id	5	ue(v)
}		
}		

D.1.11 Syntaxe de message SEI d'informations de sous-séquence

sub_seq_info(payloadSize) {	C	Descripteur
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
first_ref_pic_flag	5	u(1)
leading_non_ref_pic_flag	5	u(1)
last_pic_flag	5	u(1)
sub_seq_frame_num_flag	5	u(1)
si(sub_seq_frame_num_flag)		
sub_seq_frame_num	5	ue(v)
}		

D.1.12 Syntaxe de message SEI de caractéristiques de couche de sous-séquence

sub_seq_layer_characteristics(payloadSize) {	C	Descripteur
num_sub_seq_layers_minus1	5	ue(v)
pour(layer = 0; layer <= num_sub_seq_layers_minus1; layer++) {		
accurate_statistics_flag	5	u(1)
average_bit_rate	5	u(16)
average_frame_rate	5	u(16)
}		
}		

D.1.13 Syntaxe de message SEI de caractéristiques de sous-séquence

sub_seq_characteristics(payloadSize) {	C	Descripteur
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
duration_flag	5	u(1)
si(duration_flag)		
sub_seq_duration	5	u(32)
average_rate_flag	5	u(1)
si(average_rate_flag) {		
accurate_statistics_flag	5	u(1)
average_bit_rate	5	u(16)
average_frame_rate	5	u(16)
}		
num_referenced_subseqs	5	ue(v)
pour(n = 0; n < num_referenced_subseqs; n++) {		
ref_sub_seq_layer_num	5	ue(v)
ref_sub_seq_id	5	ue(v)
ref_sub_seq_direction	5	u(1)
}		
}		

D.1.14 Syntaxe de message SEI de gel de trame entière

full_frame_freeze(payloadSize) {	C	Descripteur
full_frame_freeze_repetition_period	5	ue(v)
}		

D.1.15 Syntaxe de message SEI de libération de gel de trame entière

full_frame_freeze_release(payloadSize) {	C	Descripteur
}		

D.1.16 Syntaxe de message SEI de photographie de trame entière

full_frame_snapshot(payloadSize) {	C	Descripteur
snapshot_id	5	ue(v)
}		

D.1.17 Syntaxe de message SEI de début de segment de raffinement progressif

progressive_refinement_segment_start(payloadSize) {	C	Descripteur
progressive_refinement_id	5	ue(v)
num_refinement_steps_minus1	5	ue(v)
}		

D.1.18 Syntaxe de message SEI de fin de segment de raffinement progressif

progressive_refinement_segment_end(payloadSize) {	C	Descripteur
progressive_refinement_id	5	ue(v)
}		

D.1.19 Syntaxe de message SEI d'ensemble de groupes de tranches à mouvement restreint

motion_constrained_slice_group_set(payloadSize) {	C	Descripteur
num_slice_groups_in_set_minus1	5	ue(v)
pour(i = 0; i <= num_slice_groups_in_set_minus1; i++)		
slice_group_id[i **]**	5	u(v)
exact_sample_value_match_flag	5	u(1)
pan_scan_rect_flag	5	u(1)
si(pan_scan_rect_flag)		
pan_scan_rect_id	5	ue(v)
}		

D.1.20 Syntaxe de message SEI de caractéristiques de grain d'émulsion

film_grain_characteristics(payloadSize) {	C	Descripteur
film_grain_characteristics_cancel_flag	5	u(1)
si(!film_grain_**characteristics_**cancel_flag) {		
model_id	5	u(2)
separate_colour_description_present_flag	5	u(1)
si(separate_colour_description_present_flag) {		
film_grain_bit_depth_luma_minus8	5	u(3)
film_grain_bit_depth_chroma_minus8	5	u(3)
film_grain_full_range_flag	5	u(1)
film_grain_colour_primaries	5	u(8)
film_grain_transfer_characteristics	5	u(8)
film_grain_matrix_coefficients	5	u(8)
}		
blending_mode_id	5	u(2)
log2_scale_factor	5	u(4)
pour(c = 0; c < 3; c++)		
comp_model_present_flag[c]	5	u(1)
pour(c = 0; c < 3; c++)		
si(comp_model_present_flag[c]) {		
num_intensity_intervals_minus1[c **]**	5	u(8)
num_model_values_minus1[c **]**	5	u(3)
pour(i = 0; i <= num_intensity_intervals_minus1[c]; i++) {		
intensity_interval_lower_bound[c **][** i **]**	5	u(8)
intensity_interval_upper_bound[c **][** i **]**	5	u(8)
pour(j = 0; j <= num_model_values_minus1[c]; j++)		
comp_model_value[c **][** i **][** j **]**	5	se(v)
}		
}		
film_grain_characteristics_repetition_period	5	ue(v)
}		
}		

D.1.21 Syntaxe de message SEI de préférence d'affichage du filtre de démontage de bloc

deblocking_filter_display_preference(payloadSize) {	C	Descripteur
deblocking_display_preference_cancel_flag	5	u(1)
si(!deblocking_display_preference_cancel_flag) {		
display_prior_to_deblocking_preferred_flag	5	u(1)
dec_frame_buffering_constraint_flag	5	u(1)
deblocking_display_preference_repetition_period	5	ue(v)
}		
}		

D.1.22 Syntaxe de message SEI d'informations stéréoscopiques

stereo_video_info(payloadSize) {	C	Descripteur
field_views_flag	5	u(1)
si(field_views_flag)		
top_field_is_left_view_flag	5	u(1)
sinon {		
current_frame_is_left_view_flag	5	u(1)
next_frame_is_second_view_flag	5	u(1)
}		
left_view_self_contained_flag	5	u(1)
right_view_self_contained_flag	5	u(1)
}		

D.1.23 Syntaxe de message SEI réservé

reserved_sei_message(payloadSize) {	C	Descripteur
pour(i = 0; i < payloadSize; i++)		
reserved_sei_message_payload_byte	5	b(8)
}		

D.2 Sémantique de charge utile SEI

D.2.1 Sémantique de message SEI de période de mise en mémoire tampon

Lorsque NalHrdBpPresentFlag ou VclHrdBpPresentFlag sont égaux à 1, un message SEI de période de mise en mémoire tampon peut être associé à toute unité d'accès dans le flux binaire, et un message SEI de période de mise en mémoire tampon doit être associé à chaque unité d'accès IDR et avec chaque unité d'accès associée à un message SEI de point de récupération.

NOTE – Pour certaines applications, la présence fréquente d'un message SEI de période de mise en mémoire tampon peut être souhaitable.

Un message SEI de période de mise en mémoire tampon est spécifié comme ensemble des unités d'accès entre deux instances du message SEI de période de mise en mémoire tampon dans l'ordre de décodage.

seq_parameter_set_id spécifie l'ensemble de paramètres de séquence qui contient la séquence d'attributs HRD. La valeur de seq_parameter_set_id doit être égale à la valeur de seq_parameter_set_id dans l'ensemble de paramètres d'image mentionné par l'image codée primaire associée au message SEI de période de mise en mémoire tampon. La valeur de seq_parameter_set_id doit être dans la gamme de 0 à 31 inclus.

initial_cpb_removal_delay[SchedSelIdx] spécifie le délai pour le SchedSelIdx^{ième} CPB entre le moment d'arrivée dans le CPB du premier bit des données codées associées à l'unité d'accès associée au message SEI de période de mise en mémoire tampon et le moment de retrait du tampon CPB des données codées associées à la même unité d'accès, pour la première période de mise en mémoire tampon après l'initialisation du décodeur HRD. L'élément syntaxique a une longueur en bits donnée par initial_cpb_removal_delay_length_minus1 + 1. Elle est en unités d'horloge à 90 kHz. initial_cpb_removal_delay[SchedSelIdx] ne doit pas être égal à 0 et ne doit pas excéder 90000 * (CpbSize[SchedSelIdx] ÷ BitRate[SchedSelIdx]), l'équivalent en temps de la taille du tampon CPB en unités d'horloge à 90 kHz.

initial_cpb_removal_delay_offset[SchedSelIdx] est utilisé pour le SchedSelIdx^{ième} CPB en combinaison avec le cpb_removal_delay afin de spécifier le moment initial de livraison des unités d'accès codées au CPB. Le initial_cpb_removal_delay_offset[SchedSelIdx] est en unités d'horloge à 90 kHz. L'élément syntaxique initial_cpb_removal_delay_offset[SchedSelIdx] est un code de longueur fixe dont la longueur en bits est donnée par initial_cpb_removal_delay_length_minus 1 + 1. Cet élément syntaxique n'est pas utilisé par les décodeurs et n'est nécessaire que pour le mesureur de flux fictsi (HSS) spécifié à l'Annexe C.

La somme de initial_cpb_removal_delay[SchedSelIdx] et initial_cpb_removal_delay_offset [SchedSelIdx] doit être constante pour chaque valeur de SchedSelIdx sur la séquence vidéo codée tout entière.

D.2.2 Sémantique de message SEI de synchronisation d'image

La présence du message SEI de synchronisation d'image dans le flux binaire est spécifiée comme suit:

– Si CpbDpbDelaysPresentFlag est égal à 1 ou si pic_struct_present_flag est égal à 1, un seul message SEI de synchronisation d'image doit être présent dans chaque unité d'accès de la séquence vidéo codée.

– Sinon (si CpbDpbDelaysPresentFlag est égal à 0 et si pic_struct_present_flag est égal à 0), aucun message SEI de synchronisation d'images ne doit être présent dans une quelconque unité d'accès de la séquence vidéo codée.

cpb_removal_delay spécifie combien de tic-tac d'horloge (voir le § E.2.1) on doit attendre après le retrait du tampon CPB de l'unité d'accès associée au plus récent message SEI de période de mise en mémoire tampon avant de retirer de la mémoire tampon les données de l'unité d'accès associées au message SEI de synchronisation d'image. Cette valeur est aussi utilisée afin de calculer le moment le plus tôt possible d'arrivée des données d'unité d'accès dans le CPB pour l'ordonnanceur HSS, comme spécifié à l'Annexe C. L'élément syntaxique est un code de longueur fixe dont la longueur en bits est donnée par cpb_removal_delay_length_minus1 + 1. Le cpb_removal_delay est le reste d'un compteur $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$.

La valeur de cpb_removal_delay pour la première image dans le flux binaire doit être égale à 0.

dpb_output_delay est utilisé afin de calculer le temps de sortie de l'image du tampon DPB. Il spécifie combien de tic-tac d'horloge on doit attendre avant le retrait d'une unité d'accès du tampon CPB avant que l'image décodée puisse être sortie du tampon DPB (voir le § C.2).

NOTE 1 – Une image n'est pas retirée du tampon DPB à son moment de sortie lorsqu'elle est toujours marquée "utilisée comme référence à court terme" ou "utilisée comme référence à long terme".

NOTE 2 – Un seul élément dpb_output_delay est spécifié pour une image décodée.

La taille de l'élément syntaxique dpb_output_delay est donnée en bits par dpb_output_delay_ length_minus1 + 1. Lorsque max_dec_frame_buffering est égal à 0, dpb_output_delay doit être égal à 0.

Le moment de sortie déduit de l'élément dpb_output_delay de toute image qui est sortie d'un décodeur conforme à la synchronisation de sortie, comme spécifié au § C.2, doit précéder le moment de sortie déduit de l'élément dpb_output_delay de toutes les images de toute séquence vidéo codée suivante dans l'ordre de décodage.

Le moment de sortie déduit de l'élément dpb_output_delay de la seconde sous-trame, dans l'ordre de décodage, d'une paire de sous-trames non référentielles complémentaires doit excéder le moment de sortie déduit de l'élément dpb_output_delay de la première sous-trame de la même paire de sous-trames non référentielles complémentaires.

L'ordre de sortie d'image établi par les valeurs de cet élément syntaxique doit être le même ordre que celui établit par les valeurs de PicOrderCnt() comme spécifié par les § C.4.1 à C.4.5, excepté que lorsque les deux sous-trames d'une paire de sous-trames de référence complémentaires ont la même valeur de PicOrderCnt(), les deux sous-trames ont des moments de sortie différents.

Pour les images qui ne sont pas sorties par le processus de "supplantation" du § C.4.5 parce qu'elles précèdent, dans l'ordre de décodage, une image à rafraîchissement IDR avec no_output_of_prior_pics_flag égal à 1 ou supposé être égal à 1, les moments de sortie déduits de l'élément dpb_output_delay doivent être croissants avec des valeurs croissantes de PicOrderCnt() par rapport à toutes les images au sein de la même séquence vidéo codée à la suite de toute image ayant un élément memory_management_control_operation égal à 5.

pic_struct indique si une image devrait être affichée comme une trame ou comme une ou plusieurs sous-trames, conformément au Tableau D-1. Le doublage de trame (pic_struct égal à 7) indique que la trame devrait être affichée deux fois consécutives, et le triplage de trame (pic_struct égal à 8) indique que la trame devrait être affichée trois fois consécutivement.

NOTE 3 – Le doublage de trame peut faciliter l'affichage, par exemple, de 25p vidéo sur un affichage de 50p et 29,97p vidéo sur un affichage de 59,94p. L'utilisation du doublage de trame et du triplage de trame en combinaison sur chaque autre trame peut faciliter l'affichage de 23,98p vidéo sur un affichage de 59,94p.

Tableau D-1 – Interprétation de pic_struct

Valeur	Affichage d'image indiqué	Restrictions	NumClockTS
0	Trame	field_pic_flag doit être 0	1
1	Sous-trame supérieure	field_pic_flag doit être 1, bottom_field_flag doit être 0	1
2	Sous-trame inférieure	field_pic_flag doit être 1, bottom_field_flag doit être 1	1
3	Sous-trame supérieure, sous-trame inférieure, dans cet ordre	field_pic_flag doit être 0	2
4	Sous-trame inférieure, sous-trame supérieure, dans cet ordre	field_pic_flag doit être 0	2
5	Sous-trame supérieure, sous-trame inférieure, sous-trame supérieure répétées dans cet ordre	field_pic_flag doit être 0	3
6	Sous-trame inférieure, sous-trame supérieure, sous-trame inférieure répétées dans cet ordre	field_pic_flag doit être 0	3
7	Doublage de trame	field_pic_flag doit être 0 fixed_frame_rate_flag doit être 1	2
8	Triplage de trame	field_pic_flag doit être 0 fixed_frame_rate_flag doit être 1	3
9..15	Réservé		

NumClockTS est déterminé par pic_struct comme spécifié au Tableau D-1. Il y a jusqu'à NumClockTS ensembles d'informations d'horodatage pour une image, comme spécifié par clock_timestamp_flag[i] pour chaque ensemble. Les ensembles d'informations d'horodatage s'appliquent à la(aux) sous-trame(s) ou à la (aux) trame(s) associée(s) à l'image par pic_struct.

Le contenu des éléments syntaxiques d'horodatage indique un moment d'origine, de capture, ou d'affichage idéal de remplacement. Ce moment indiqué est calculé somme suit:

$$\begin{aligned} clockTimestamp = {} & ((hH * 60 + mM) * 60 + sS) * time\_scale + \\ & nFrames * (num\_units\_in\_tick * (1 + nuit\_field\_based\_flag)) + tOffset, \end{aligned} \quad \text{(D-1)}$$

en unités de tic-tac d'horloge d'une horloge avec une fréquence d'horloge égale à time_scale Hz, par rapport à un point non spécifié dans le temps pour lequel clockTimestamp est égal à 0. L'ordre de sortie et la synchronisation de sortie du tampon DPB ne sont pas affectés par la valeur de clockTimestamp. Lorsque deux, ou plus, trames avec pic_struct égal à 0 sont consécutives dans l'ordre de sortie et ont d'égales valeurs de clockTimestamp, l'indication est que les trames représentent le même contenu et que la dernière de ces trames dans l'ordre de sortie est la représentation préférée.

NOTE 4 – Les indications de temps de clockTimestamp peuvent aider à l'affichage sur des appareils qui ont des taux de rafraîchissement autres que ceux qui sont en pleine correspondance avec les temps de sortie du tampon DPB.

clock_timestamp_flag[i **]** égal à 1 indique qu'un certain nombre d'éléments syntaxiques d'horodatage sont présents et suivent immédiatement. clock_timestamp_flag[i] égal à 0 indique que les éléments syntaxiques d'horodatage associés ne sont pas présents. Lorsque NumClockTS est supérieur à 1 et que clock_timestamp_flag[i] est égal à 1 pour plus d'une valeur de i, la valeur de clockTimestamp doit être non décroissante avec une valeur de i croissante.

ct_type indique le type de balayage (entrelacé ou progressif) du matériau source comme spécifié dans le Tableau D-2.

Deux sous-trames d'une trame codée peuvent avoir différentes valeurs de ct_type.

Lorsque clockTimestamp est égal pour deux sous-trames de parité opposée qui sont consécutives dans l'ordre de sortie, toutes deux avec ct_type égal à 0 (balayage progressif) ou ct_type égal à 2 (inconnu), les deux sous-trames sont indiquées comme provenant de la même trame progressive d'origine. Deux sous-trames consécutives dans l'ordre de sortie doivent avoir des valeurs de clockTimestamp différentes lorsque la valeur de ct_type pour l'une ou l'autre sous-trame est 1 (balayage entrelacé).

Tableau D-2 – Mappage de ct_type avec le balayage de l'image source

Valeur	Balayage de l'image d'origine
0	Progressif
1	Entrelacé
2	Inconnu
3	Réservé

nuit_field_based_flag est utilisé pour le calcul de clockTimestamp, comme spécifié à l'équation D-1.

counting_type spécifie la méthode de réduction des valeurs de n_frames comme spécifié au Tableau D-3.

Tableau D-3 – Définition des valeurs de counting_type

Valeur	Interprétation
0	Pas de réduction des valeurs de comptage de n_frames et pas d'utilisation de time_offset
1	Pas de réduction des valeurs de comptage de n_frames
2	Réduction des valeurs zéro individuelles du comptage de n_frames
3	Réduction des valeurs individuelles MaxFPS-1 du comptage de n_frames
4	Réduction des deux comptes les plus faibles (valeur 0 et 1) de n_frames lorsque seconds_value est égal à 0 et que minutes_value n'est pas un multiple entier de 10
5	Réduction des valeurs individuelles non spécifiées de comptage de n_frames
6	Réduction des numéros non spécifiés des valeurs de comptage non spécifié de n_frames
7..31	Réservé

full_timestamp_flag égal à 1 spécifie que l'élément syntaxique n_frames est suivi par seconds_value, minutes_value, et hours_value. full_timestamp_flag égal à 0 spécifie que l'élément syntaxique n_frames est suivi par seconds_flag.

discontinuity_flag égal à 0 indique que la différence entre la valeur en cours de clockTimestamp et la valeur de clockTimestamp calculée d'après l'horodatage précédent dans l'ordre de sortie peut s'interpréter comme la différence de temps entre le moment d'origine ou de capture des trames ou sous-trames associées. Un fanion discontinuity_flag égal à 1 indique que la différence entre la valeur en cours de clockTimestamp et la valeur de clockTimestamp calculée d'après l'horodatage précédent dans l'ordre de sortie ne devrait pas être interprétée comme la différence de temps entre le moment d'origine ou de capture des trames ou sous-trames associées. Lorsque discontinuity_flag est égal à 0, la valeur de clockTimestamp doit être supérieure ou égale à toutes les valeurs de clockTimestamp présentes pour la précédente image dans l'ordre de sortie du tampon DPB.

cnt_dropped_flag spécifie qu'on saute une ou plusieurs valeurs de n_frames au moyen de la méthode de comptage spécifiée par counting_type.

n_frames spécifie la valeur de nFrames utilisée afin de calculer clockTimestamp. n_frames doit être inférieur à

$$MaxFPS = Ceil(time\_scale \div num\_units\_in\_tick) \quad (D\text{-}2)$$

NOTE 5 – n_frames est un comptage fondé sur les trames. Pour les indications de synchronisation spécifiques des sous-trames, time_offset devrait être utilisé afin d'indiquer un clockTimestamp distinct pour chaque sous-trame.

Lorsque counting_type est égal à 2 et que cnt_dropped_flag est égal à 1, n_frames doit être égal à 1 et la valeur de n_frames pour l'image précédente dans l'ordre de sortie ne doit pas être égale à 0 à moins que discontinuity_flag ne soit égal à 1.

NOTE 6 – Lorsque counting_type est égal à 2, on peut éviter d'être obligé d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers (par exemple, 12,5 trames par seconde avec time_scale égal à 25 et num_units_in_tick égal à 2 et nuit_field_based_flag égal à 0) en sautant occasionnellement la valeur n_frames égale à 0 lors du comptage (par exemple, en comptant n_frames de 0 à 12, puis en incrémentant seconds_value et en comptant n_frames de 1 à 12, puis en incrémentant seconds_value et en comptant n_frames de 0 à 12, etc.).

Lorsque counting_type est égal à 3 et que cnt_dropped_flag est égal à 1, n_frames doit être égal à 0 et la valeur de n_frames pour l'image précédente dans l'ordre de sortie ne doit pas être égale à MaxFPS – 1 à moins que discontinuity_flag ne soit égal à 1.

NOTE 7 – Lorsque counting_type est égal à 3, on peut éviter d'être obligé d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers (par exemple, 12,5 trames par seconde avec time_scale égal à 25 et num_units_in_tick égal à 2 et nuit_field_based_flag égal à 0) en sautant occasionnellement la valeur n_frames égale à MaxFPS lors du comptage (par exemple, en comptant n_frames de 0 à 12, puis en incrémentant seconds_value et en comptant n_frames de 1 à 11, puis en incrémentant seconds_value et en comptant n_frames de 0 à 12, etc.).

Lorsque counting_type est égal à 4 et cnt_dropped_flag est égal à 1, n_frames doit être égal à 2 et la valeur spécifiée de sS doit être zéro et la valeur spécifiée de mM ne doit pas être un multiple entier de dix et n_frames pour l'image précédente dans l'ordre de sortie ne doit pas être égal à 0 ou 1 à moins que discontinuity_flag ne soit égal à 1.

NOTE 8 – Lorsque counting_type est égal à 4, le besoin d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers (par exemple, 30000÷1001 trames par seconde avec time_scale égal à 60000 et num_units_in_tick égal à 1001 et nuit_field_based_flag égal à 1) peut être réduit en sautant occasionnellement la valeur n_frames égale à MaxFPS lors du comptage (par exemple, en comptant n_frames de 0 à 29, puis en incrémentant seconds_value et en comptant n_frames de 0 à 29, etc., jusqu'à ce que seconds_value soit zéro et que minutes_value ne soit pas un multiple entier de dix, puis en comptant n_frames de 2 à 29, puis en incrémentant seconds_value et en comptant n_frames de 0 à 29, etc.). Cette méthode de comptage est bien connue dans l'industrie et est souvent mentionnée comme le comptage "réduction de trame NTSC".

Lorsque counting_type est égal à 5 ou 6 et que cnt_dropped_flag est égal à 1, n_frames ne doit pas être égal à 1 plus la valeur de n_frames pour l'image précédente dans l'ordre de sortie modulo MaxFPS à moins que discontinuity_flag ne soit égal à 1.

NOTE 9 – Lorsque counting_type est égal à 5 ou 6, on peut éviter d'être obligé d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers en sautant occasionnellement certaines valeurs de n_frames lors du comptage. Les valeurs spécifiques de n_frames qui sont sautées ne sont pas spécifiées lorsque counting_type est égal à 5 ou 6.

seconds_flag égal à 1 spécifie que seconds_value et minutes_flag sont présents lorsque full_timestamp_flag est égal à 0. seconds_flag égal à 0 spécifie que seconds_value et minutes_flag ne sont pas présents.

seconds_value spécifie la valeur de sS utilisée afin de calculer clockTimestamp. La valeur de seconds_value doit être dans la gamme de 0 à 59 inclus. Lorsque seconds_value n'est pas présent, le seconds_value précédant dans l'ordre de décodage doit être utilisé comme sS afin de calculer clockTimestamp.

minutes_flag égal à 1 spécifie que minutes_value et hours_flag sont présents lorsque full_timestamp_flag est égal à 0 et seconds_flag est égal à 1. minutes_flag égal à 0 spécifie que minutes_value et hours_flag ne sont pas présents.

minutes_value spécifie la valeur de mM utilisée afin de calculer clockTimestamp. La valeur de minutes_value doit être dans la gamme de 0 à 59 inclus. Lorsque minutes_value n'est pas présent, le minutes_value précédant dans l'ordre de décodage doit être utilisé comme mM afin de calculer clockTimestamp.

hours_flag égal à 1 spécifie que hours_value est présent lorsque full_timestamp_flag est égal à 0 et que seconds_flag est égal à 1 et minutes_flag est égal à 1.

hours_value spécifie la valeur de hH utilisée afin de calculer clockTimestamp. La valeur de hours_value doit être dans la gamme de 0 à 23 inclus. Lorsque hours_value n'est pas présent, le hours_value précédant dans l'ordre de décodage doit être utilisé comme hH afin de calculer clockTimestamp.

time_offset spécifie la valeur de tOffset utilisée afin de calculer clockTimestamp. Le nombre de bits utilisé afin de représenter time_offset doit être égal à time_offset_length. Lorsque time_offset n'est pas présent, la valeur 0 doit être utilisée comme tOffset afin de calculer clockTimestamp.

D.2.3 Sémantique de message SEI de rectangle de balayage complet

Les éléments syntaxiques de message SEI de rectangle de balayage complet spécifient les coordonnées d'un rectangle par rapport à la mire de l'ensemble de paramètres de séquence. Chaque coordonnée de ce rectangle est spécifiée en unités d'espacement d'un seizième d'échantillon par rapport à la grille d'échantillonnage luma.

pan_scan_rect_id contient un numéro d'identification qui peut être utilisé afin d'identifier l'objet du rectangle de balayage complet (par exemple, afin d'identifier le rectangle comme la zone à montrer d'un dispositif d'affichage particulier ou comme la zone qui contient un acteur particulier dans la scène). La valeur de pan_scan_rect_id doit être dans la gamme de 0 à $2^{32} – 1$ inclus.

Les valeurs de pan_scan_rect_id de 0 à 255 et de 512 à $2^{31} –1$ peuvent être utilisées comme déterminé par l'application. Les valeurs de pan_scan_rect_id de 256 à 511 et de 2^{31} à $2^{32} –1$ sont réservées afin d'être utilisées à l'avenir par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de pan_scan_rect_id dans la gamme de 256 à 511 ou dans la gamme de 2^{31} à $2^{32} – 1$ doivent l'ignorer (la retirer du flux binaire et la détruire).

pan_scan_rect_cancel_flag égal à 1 indique que le message SEI annule le maintien de tout message SEI précédant de rectangle de balayage complet, dans l'ordre de sortie. pan_scan_rect_cancel_flag égal à 0 indique que suivent des informations de rectangle de balayage complet.

pan_scan_cnt_minus1 spécifie le nombre de rectangles de balayage complet qui sont présents dans le message SEI. pan_scan_cnt_minus1 doit être dans la gamme de 0 à 2 inclus. pan_scan_cnt_minus1 égal à 0 indique qu'un seul rectangle de balayage complet est présent et qu'il s'applique à toutes les sous-trames de l'image décodée. pan_scan_cnt_minus1 doit être égal à 0 lorsque l'image en cours est une sous-trame. pan_scan_cnt_minus1 égal à 1 indique que deux rectangles de balayage complet sont présents, dont le premier s'applique au premier sous-trame de l'image dans l'ordre de sortie et dont le second s'applique à la seconde sous-trame de l'image dans l'ordre de sortie. pan_scan_cnt_minus1 égal à 2 indique que trois rectangles de balayage complet sont présents, dont le premier s'applique à la première sous-trame de l'image dans l'ordre de sortie, le second s'applique à la seconde sous-trame de l'image dans l'ordre de sortie, et le troisième s'applique à une répétition de la première sous-trame comme troisième sous-trame dans l'ordre de sortie.

pan_scan_rect_left_offset[i **]**, **pan_scan_rect_right_offset[** i **]**, **pan_scan_rect_top_offset[** i **]**, et **pan_scan_rect_bottom_offset[** i **]** spécifient, comme quantités algébriques entières en unités d'espacement d'un seizième d'échantillon par rapport à la grille d'échantillonnage luma, la localisation du rectangle de balayage complet. Les valeurs de chacun de ces quatre éléments syntaxiques doivent être dans la gamme de -2^{31} à $2^{31} - 1$, inclus.

Le rectangle de balayage complet est spécifié, en unités d'espacement d'un seizième d'échantillon par rapport à une grille d'échantillonnage luma, comme étant la région ayant des coordonnées horizontales de trame égales à 16*CropUnitX * frame_crop_left_offset + pan_scan_rect_left_offset[i] à 16 * (16 * PicWidthInMbs – CropUnitX * frame_crop_ right_offset) + pan_scan_rect_right_offset[i] – 1 et ayant des coordonnées verticales égales à 16 *CropUnitY * frame_crop_top_offset + pan_scan_rect_top_offset[i] à 16 * (16 * PicHeightInMbs – CropUnitY * frame_crop_bottom_offset) + pan_scan_ rect_bottom_offset[i] – 1, inclus. La valeur de 16 * CropUnitX * frame_crop_left_offset + pan_scan_rect_left_offset[i] doit être inférieure ou égale à 16 * (16 * PicWidthInMbs – CropUnitX * frame_crop_right_offset) + pan_scan_rect_right_offset[i] – 1; et la valeur de 16 CropUnitY * frame_crop_top_offset + pan_scan_rect_top_offset[i] doit être inférieure ou égale à 16 * (16 * PicHeightInMbs – CropUnitY * frame_crop_ bottom_offset) + pan_scan_rect_bottom_offset[i] – 1.

Lorsque la zone du rectangle de balayage complet inclut des échantillons en dehors de la mire, la région en dehors de la mire peut être remplie avec un contenu synthétisé (comme un contenu vidéo noir ou gris neutre) pour l'affichage.

pan_scan_rect_repetition_period spécifie la persistance du message SEI de rectangle de balayage complet et peut spécifier un intervalle de compte d'ordre d'image au sein duquel doit être présent un autre message SEI de rectangle de balayage complet ayant la même valeur de pan_scan_rect_id ou contenant la fin de la séquence vidéo codée dans le flux binaire. La valeur de pan_scan_rect_repetition_period doit être dans la gamme de 0 à 16 384 inclus. Lorsque pan_scan_cnt_minus1 est supérieur à 0, pan_scan_rect_repetition_period ne doit pas être supérieur à 1.

pan_scan_rect_repetition_period égal à 0 spécifie que les informations de rectangle de balayage complet s'appliquent uniquement à l'image décodée en cours.

pan_scan_rect_repetition_period égal à 1 spécifie que les informations de rectangle de balayage complet persistent dans l'ordre de sortie tant que l'une des conditions suivantes est vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id est sortie avec PicOrderCnt() plus grand que PicOrderCnt(CurrPic).

pan_scan_rect_repetition_period égal à 0 ou égal à 1 indique qu'un autre message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id peut être ou n'être pas présente.

pan_scan_rect_repetition_period supérieur à 1 spécifie que les informations de rectangle de balayage complet persistent tant que l'une des conditions suivantes est vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id est sortie et elle a PicOrderCnt() supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + pan_scan_rect_repetition_period.

pan_scan_rect_repetition_period supérieur à 1 indique qu'un autre message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id doit être présent pour une image dans une unité d'accès qui est sortie en ayant PicOrderCnt() supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + pan_scan_rect_repetition_period; à moins que le flux binaire ne se termine ou qu'une nouvelle séquence vidéo codée ne commence sans sortie d'une telle image.

D.2.4 Sémantique de message SEI de charge utile de remplissage

Ce message contient une série d'octets de payloadSize d'une valeur de 0xFF, qui peut être ignorée.

ff_byte doit être un octet ayant la valeur 0xFF.

D.2.5 Sémantique de message SEI de données d'usager enregistrées par la Rec. UIT-T T.35

Ce message contient des données d'usager enregistrées comme spécifié par la Rec. UIT-T T.35, dont le contenu n'est pas spécifié par la présente Recommandation | Norme internationale.

itu_t_t35_country_code doit être un octet d'une valeur spécifiée comme code de pays par la Rec. UIT-T T.35, Annexe A.

itu_t_t35_country_code_extension_byte doit être un octet d'une valeur spécifiée comme code de pays par la Rec. UIT-T T.35, Annexe B.

itu_t_t35_payload_byte doit être un octet contenant des données enregistrées comme spécifié par la Rec. UIT-T T.35.

Le code de fournisseur de terminal et le code orienté fournisseur de terminal UIT-T T.35 doivent être contenus dans le ou les premiers octets de itu_t_t35_payload_byte, dans le format spécifié par l'Administration qui a fourni le code de fournisseur de terminal. Toutes les données restantes de itu_t_t35_payload_byte doivent être des données qui ont la syntaxe et la sémantique spécifiées par l'entité identifiée par le code de pays et les codes de fournisseur de terminal UIT-T T.35.

D.2.6 Sémantique de message SEI de données d'usager non enregistrées

Ce message contient des données d'usager non enregistrées identifiées par un identificateur UUID, dont le contenu n'est pas spécifié par la présente Recommandation | Norme internationale.

uuid_iso_iec_11578 doit avoir une valeur spécifiée comme un identificateur UUID conformément aux procédures de l'ISO/CEI 11578:1996, Annexe A.

user_data_payload_byte doit être un octet contenant des données ayant la syntaxe et la sémantique spécifiées par le générateur d'identificateurs UUID.

D.2.7 Sémantique de message SEI de point de récupération

Le message SEI de point de récupération aide un décodeur à déterminer lorsque le processus de décodage va produire des images acceptables pour l'affichage après que le décodeur a initialisé un accès aléatoire, ou après que le codeur a indiqué une rupture de liaison dans la séquence. Lorsque le processus de décodage débute avec l'unité d'accès dans l'ordre de décodage associée au message SEI de point de récupération, toutes les images décodées au point de récupération ou à sa suite dans l'ordre de sortie spécifié dans ce message SEI sont indiquées comme étant correctes ou approximativement correctes quant à leur contenu. Les images décodées produites par accès aléatoire au moment de l'image associée au message SEI de point de récupération, ou avant elle, n'ont pas besoin d'être correctes quant à leur contenu jusqu'au point de récupération indiqué, et l'opération du processus de décodage commençant à l'image associée au message SEI de point de récupération peut contenir des références aux images indisponibles dans la mémoire tampon d'image décodée.

De plus, au moyen du fanion broken_link_flag, le message SEI de point de récupération peut indiquer au décodeur la localisation de certaines images dans le flux binaire qui peuvent produire à l'affichage de sérieuses distorsions visuelles, même lorsque le processus de décodage avait commencé à la localisation d'une unité d'accès IDR précédente dans l'ordre de décodage.

NOTE 1 – Le fanion broken_link_flag peut être utilisé par les codeurs afin d'indiquer la localisation d'un point après lequel le processus de décodage pour le décodage de certaines images peut provoquer la référence à des images qui, bien que disponibles pour une utilisation dans le processus de décodage, ne sont pas les images qui étaient utilisées pour référence lorsque le flux binaire a été codé à l'origine (par exemple, à cause d'une opération de collage effectuée pendant la production du flux binaire).

Le point de récupération est spécifié comme un compte en unités d'incréments frame_num suivant le frame_num de l'unité d'accès en cours à la position du message SEI.

NOTE 2 – Lorsque des informations HRD sont présentes dans le flux binaire, un message SEI de période de mise en mémoire tampon devrait être associé à l'unité d'accès associée au message SEI de point de récupération afin d'établir l'initialisation du modèle de mémoire tampon HRD après un accès aléatoire.

recovery_frame_cnt spécifie le point de récupération des images sorties dans l'ordre de sortie. Toutes les images décodées dans l'ordre de sortie sont indiquées comme correctes ou approximativement correctes quant à leur contenu en commençant à la position d'ordre de sortie de l'image de référence qui a le frame_num égal au frame_num des unités NAL de couche VCL pour l'unité d'accès en cours incrémentée de recovery_frame_cnt en arithmétique modulo MaxFrameNum. recovery_frame_cnt doit être dans la gamme de 0 à MaxFrameNum – 1, inclus.

exact_match_flag indique si les images décodées au point de récupération et à sa suite dans l'ordre de sortie, calculé en commençant le processus de décodage à l'unité d'accès associée au message SEI de point de récupération, doivent correspondre exactement aux images qui seraient produites en commençant le processus de décodage à la localisation d'une unité d'accès IDR précédente dans le flux d'unités NAL. La valeur 0 indique que la correspondance peut n'être pas exacte et la valeur 1 indique que la correspondance doit être exacte.

Lorsque le décodage débute à la localisation du message SEI de point de récupération, toutes les références à des images de référence indisponibles doivent être comprises comme des références à des images ne contenant que des macroblocs codés utilisant les modes d'intraprédiction de macrobloc et ayant des valeurs d'échantillon données par Y échantillons égaux à 128, Cb échantillons égaux à 128, et Cr échantillons égaux à 128 (gris moyen) pour les besoins de la détermination de la conformité de la valeur de exact_match_flag.

NOTE 3 – Lorsqu'on effectue un accès aléatoire, les décodeurs devraient comprendre que toutes les références à des images de référence indisponibles sont des références à des images ne contenant que des macroblocs intra et qui ont des valeurs d'échantillon données par Y égal à 128, Cb égal à 128, et Cr égal à 128 (gris moyen), indépendamment de la valeur de exact_match_flag.

Lorsque exact_match_flag est égal à 0, la qualité de l'approximation au point de récupération est choisie par le processus de codage et n'est pas spécifiée par la présente Recommandation | Norme internationale.

broken_link_flag indique la présence ou l'absence d'une rupture de liaison dans le flux d'unité NAL à la localisation du message SEI de point de récupération et a la sémantique suivante:

– si broken_link_flag est égal à 1, les images produites en commençant le processus de décodage à la localisation d'une unité d'accès IDR précédente peuvent contenir des distorsions visuelles indésirables à tel point que les images décodées à l'unité d'accès associée au message SEI de point de récupération et à sa suite dans l'ordre de décodage ne devraient pas être affichées jusqu'au point de récupération spécifié dans l'ordre de sortie;

– sinon (si broken_link_flag est égal à 0), aucune indication n'est donnée concernant la présence potentielle de distorsions visuelles.

Indépendamment de la valeur du fanion broken_link_flag, les images à la suite du point de récupération spécifié dans l'ordre de sortie sont spécifiées comme correctes ou approximativement correctes quant à leur contenu.

NOTE 4 – Lorsqu'un message SEI d'information de sous-séquence est présent en conjonction avec un message SEI de point de récupération dans lequel broken_link_flag est égal à 1 et lorsque sub_seq_layer_num est égal à 0, sub_seq_id devrait être différent du dernier sub_seq_id pour sub_seq_layer_num égal à 0 qui a été décodé avant la localisation du message SEI de point de récupération. Lorsque broken_link_flag est égal à 0, le sub_seq_id dans la couche 0 de sous-séquence devrait rester inchangé.

changing_slice_group_idc égal à 0 indique que les images décodées sont correctes ou approximativement correctes quant à leur contenu au point de récupération et à sa suite dans l'ordre de sortie lorsque tous les macroblocs des images codées primaires sont décodés au sein de la période de groupe de tranches changeant, c'est-à-dire, la période entre l'unité d'accès associée au message SEI de point de récupération (inclus) et le point de récupération spécifié (inclus) dans l'ordre de décodage. changing_slice_group_idc doit être égal à 0 lorsque num_slice_groups_minus1 est égal à 0 dans toute image codée primaire au sein de la période de groupe de tranches changeant.

Lorsque changing_slice_group_idc est égal à 1 ou 2, num_slice_groups_minus1 doit être égal à 1 et le type de mappage 3, 4 ou 5 de macrobloc à groupe de tranche doit être appliqué dans chaque image codée primaire dans la période de groupe de tranches changeant.

changing_slice_group_idc égal à 1 indique qu'au sein de la période de groupe de tranches changeant, aucune valeur d'échantillon en dehors des macroblocs décodés couverts par le groupe de tranches 0 n'est utilisée pour l'interprédiction de tout macrobloc au sein du groupe de tranches 0. De plus, changing_slice_group_idc égal à 1 indique que lorsque tous les macroblocs dans le groupe de tranches 0 au sein de la période de groupe de tranches changeant sont décodés, les images décodées seront correctes ou approximativement correctes quant à leur contenu au point de récupération spécifié et à sa suite dans l'ordre de sortie indépendamment du décodage éventuel d'un macrobloc dans le groupe de tranches 1 au sein de la période de groupe de tranches changeant.

changing_slice_group_idc égal à 2 indique qu'au sein de la période de groupe de tranches changeant, aucune valeur d'échantillon en dehors des macroblocs décodés couverts par le groupe de tranches 1 n'est utilisée pour l'interprédiction de tout macrobloc au sein du groupe de tranches 1. De plus, changing_slice_group_idc égal à 2 indique que lorsque tous les macroblocs dans le groupe de tranches 1 au sein de la période de groupe de tranches changeant sont décodés, les images décodées seront correctes ou approximativement correctes quant à leur contenu au point de récupération spécifié et à sa suite dans l'ordre de sortie indépendamment du décodage éventuel d'un macrobloc dans le groupe de tranches 0 au sein de la période de groupe de tranches changeant.

changing_slice_group_idc doit être dans la gamme de 0 à 2, inclus.

D.2.8 Sémantique de message SEI de répétition de marquage d'image de référence décodée

Le message SEI de répétition de marquage d'image de référence décodée est utilisé afin de répéter la structure syntaxique de marquage d'image de référence décodée qui était localisée dans l'en-tête de tranche d'une image précédente dans la séquence dans l'ordre de décodage.

original_idr_flag doit être égal à 1 lorsque la structure syntaxique de marquage d'image de référence décodée survenait à l'origine dans uneimage à rafraîchissement IDR. original_idr_flag doit être égal à 0 lorsque la structure syntaxique de marquage d'image de référence décodée répétée ne survenait pas à l'origine dans uneimage à rafraîchissement IDR.

original_frame_num doit être égal au frame_num de l'image où survenait à l'origine la structure syntaxique de marquage d'image de référence décodée répétée. L'image indiquée par original_frame_num est l'image codée précédente qui a la valeur de frame_num spécifiée. La valeur de original_frame_num utilisée afin de faire référence à une image ayant un élément memory_management_control_operation égal à 5 doit être 0.

original_field_pic_flag doit être égal au fanion field_pic_flag de l'image où survenait à l'origine la structure syntaxique de marquage d'image de référence décodée répétée.

original_bottom_field_flag doit être égal au fanion bottom_field_flag de l'image où survenait à l'origine la structure syntaxique de marquage d'image de référence décodée répétée.

dec_ref_pic_marking( ) doit contenir une copie de la structure syntaxique de marquage d'image de référence décodée de l'image dont frame_num était original_frame_num. Le nal_unit_type utilisé pour la spécification de la structure syntaxique dec_ref_pic_marking( ) répétée doit être le nal_unit_type de ou des en-têtes de tranche de l'image dont frame_num était original_frame_num (c'est-à-dire que nal_unit_type comme utilisé au § 7.3.3.3 doit être considéré comme égal à 5 lorsque original_idr_flag est égal à 1 et ne doit pas être considéré comme égal à 5 lorsque original_idr_flag est égal à 0).

D.2.9 Sémantique de message SEI d'image de rechange

Ce message SEI indique que certaines unités de mappage de groupe de tranches, appelées *unités de rechange de mappage de groupe de tranches*, dans une ou plusieurs images de référence décodées, ressemblent aux unités de mappage de groupe de tranches colocalisées dans une image décodée spécifiée appelée l'*image cible.* Une unité de rechange de mappage de groupe de tranches peut être utilisée afin de remplacer une unité de mappage de groupe de tranches colocalisées incorrectement décodée dans l'image cible. Une image décodée contenant des unités de rechange de mappage de groupe de tranches est appelée *image de rechange*.

Pour toutes les images de rechange identifiées dans un message SEI d'image de rechange, la valeur du fanion frame_mbs_only_flag doit être égale à la valeur du fanion frame_mbs_only_flag de l'image cible dans le même message SEI. Les images de rechange contenues dans le message SEI sont soumises aux contraintes suivantes.

– Si l'image cible est une sous-trame décodée, toutes les images de rechange identifiées dans le même message SEI doivent être des sous-trames décodées.

– Sinon (si l'image cible est une trame décodée), toutes les images de rechange identifiées dans le même message SEI doivent être des trames décodées.

Pour toute image de rechange identifiée dans un message SEI d'image de rechange, les valeurs de pic_width_in_mbs_minus1 et pic_height_in_map_units_minus1 doivent être respectivement égales aux valeurs de pic_width_in_mbs_minus1 et pic_height_in_map_units_minus1 de l'image cible dans le même message SEI. L'image associée (comme spécifié au § 7.4.1.2.3) à ce message doit apparaître après l'image cible, dans l'ordre de décodage.

target_frame_num indique le frame_num de l'image cible.

spare_field_flag égal à 0 indique que l'image cible et les images de rechange sont des trames décodées. spare_field_flag égal à 1 indique que l'image cible et les images de rechange sont des sous-trames décodées.

target_bottom_field_flag égal à 0 indique que l'image cible est une sous-trame supérieure. target_bottom_field_flag égal à 1 indique que l'image cible est une sous-trame inférieure.

Une image cible est une image décodée de référence dont l'image codée primaire correspondante précède l'image en cours, dans l'ordre de décodage, et dans laquelle les valeurs de frame_num, field_pic_flag (lorsqu'il est présent) et bottom_field_flag (lorsqu'il est présent) sont respectivement égales à target_frame_num, spare_field_flag et target_bottom_field_flag.

num_spare_pics_minus1 indique le nombre d'images de rechange pour l'image cible spécifiée. Le nombre d'images de rechange est égal à num_spare_pics_minus1 + 1. La valeur de num_spare_pics_minus1 doit être dans la gamme de 0 à 15, inclus.

delta_spare_frame_num[i **]** est utilisé afin d'identifier l'image de rechange qui contient le $i^{ème}$ ensemble d'unités de mappage de groupe de tranches de rechange, ci-après appelée la $i^{ème}$ image de rechange, comme spécifié ci-dessous. La valeur de delta_spare_frame_num[i] doit être dans la gamme de 0 à MaxFrameNum – 1 – !spare_field_flag, inclus.

Le frame_num de la $i^{ème}$ image de rechange, spareFrameNum[i], est déduit comme suit pour toutes les valeurs de i de 0 à num_spare_pics_minus1, inclus:

```
candidateSpareFrameNum = target_frame_num - !spare_field_flag
pour( i = 0; i <= num_spare_pics_minus1; i++ ) {
    si( candidateSpareFrameNum < 0 )
        candidateSpareFrameNum = MaxFrameNum – 1
    spareFrameNum[ i ] = candidateSpareFrameNum – delta_spare_frame_num[ i ]          (D-3)
    si( spareFrameNum[ i ] < 0 )
        spareFrameNum[ i ] = MaxFrameNum + spareFrameNum[ i ]
    candidateSpareFrameNum = spareFrameNum[ i ] - !spare_field_flag
}
```

spare_bottom_field_flag[i **]** égal à 0 indique que la $i^{ème}$ image de rechange est une sous-trame supérieure. spare_bottom_field_flag[i] égal à 1 indique que la $i^{ème}$ image de rechange est une sous-trame inférieure.

L'image de rechange 0 est une image décodée de référence dont l'image codée primaire correspondante précède l'image cible, dans l'ordre de décodage, et dans laquelle les valeurs de frame_num, field_pic_flag (lorsqu'il est présent) et bottom_field_flag (lorsqu'il est présent) sont respectivement égales à spareFrameNum[0], spare_field_flag et spare_bottom_field_flag[0]. La $i^{ème}$ image de rechange est une image décodée de référence dont l'image codée primaire correspondante précède la $(i – 1)^{ème}$ image de rechange, dans l'ordre de décodage, et dans laquelle les valeurs de frame_num, field_pic_flag (lorsqu'il est présent) et bottom_field_flag (lorsqu'il est présent) sont respectivement égales à spareFrameNum[i], spare_field_flag et spare_bottom_field_flag[i].

spare_area_idc[i **]** indique la méthode utilisée afin d'identifier les unités de mappage de groupe de tranches de rechange dans la $i^{ème}$ image de rechange. spare_area_idc[i] doit être dans la gamme de 0 à 2, inclus. spare_area_idc[i] égal à 0 indique que toutes les unités de mappage de groupe de tranches dans la $i^{ème}$ image de rechange sont des unités de rechange. spare_area_idc [i] égal à 1 indique que la valeur de l'élément syntaxique spare_unit_flag[i][j] est utilisée afin d'identifier les unités de mappage de groupe de tranches de rechange. spare_area_idc[i] égal à 2 indique que l'élément syntaxique zero_run_length[i][j] est utilisé afin de calculer les valeurs de spareUnitFlagInBoxOutOrder[i][j], comme décrit ci-dessous.

spare_unit_flag[i **][** j **]** égal à 0 indique que la $j^{ème}$ unité de mappage de groupe de tranches dans l'ordre de balayage matriciel dans la $i^{ème}$ image de rechange est une unité de rechange. spare_unit_flag[i][j] égal à 1 indique que la $j^{ème}$ unité de mappage de groupe de tranches dans l'ordre de balayage matriciel dans la $i^{ème}$ image de rechange n'est pas une unité de rechange.

zero_run_length[i **][** j **]** est utilisé afin de calculer les valeurs de spareUnitFlagInBoxOutOrder[i][j] lorsque spare_area_idc[i] est égal à 2. Dans ce cas, les unités de mappage de groupe de tranches de rechange identifiées dans spareUnitFlagInBoxOutOrder[i][j] apparaissent dans l'ordre de sortie de boîte contraire des aiguilles d'une montre, comme spécifié au § 8.2.2.4, pour chaque image de rechange. spareUnitFlagInBoxOutOrder[i][j] égal à 0 indique que la $j^{ème}$ unité de mappage de groupe de tranches dans l'ordre de sortie de boîte contraire des aiguilles d'une montre dans la $i^{ème}$ image de rechange est une unité de rechange. spareUnitFlagInBoxOutOrder[i][j] égal à 1 indique que la $j^{ème}$ unité de mappage de groupe de tranches dans l'ordre de sortie de boîte contraire des aiguilles d'une montre dans la $i^{ème}$ image de rechange n'est pas une unité de rechange.

Lorsque spare_area_idc[0] est égal à 2, spareUnitFlagInBoxOutOrder[0][j] est calculé comme suit:

```
pour( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
    pour( k = 0; k < zero_run_length[ 0 ][ loop ]; k++ )
        spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 0                    (D-4)
    spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 1
}
```

Lorsque spare_area_idc[i] est égal à 2 et que la valeur de i est supérieure à 0, spareUnitFlagInBoxOutOrder[i][j] est calculé comme suit:

```
pour( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
    pour( k = 0; k < zero_run_length[ i ][ loop ]; k++ )
        spareUnitFlagInBoxOutOrder[ i ][ j ] = spareUnitFlagInBoxOutOrder[ i - 1 ][ j++ ]    (D-5)
```

```
        spareUnitFlagInBoxOutOrder[ i ][ j ] = !spareUnitFlagInBoxOutOrder[ i - 1 ][ j++ ]
    }
```

D.2.10 Sémantique de message SEI d'informations de scène

Une scène et une transition de scène sont définies ci-après comme un ensemble d'images consécutives dans l'ordre de sortie.

NOTE 1 – Les images décodées au sein d'une seule scène ont généralement un contenu similaire. Le message SEI d'informations de scène est utilisé afin d'étiqueter les images avec des identifiants de scène et afin d'indiquer les changements de scène. Le message spécifie comment ont été créées les images sources pour les images étiquetées. Le décodeur peut utiliser les informations afin de choisir un algorithme approprié afin de dissimuler les erreurs de transmission. Par exemple, un algorithme spécifique peut être utilisé afin de dissimuler les erreurs de transmission qui sont survenues dans des images appartenant à une transition graduelle de scène. De plus, le message SEI d'informations de scène peut être utilisé d'une façon déterminée par l'application, comme l'indexation des scènes d'une séquence codée.

Un message SEI d'informations de scène étiquette toutes les images, dans l'ordre de décodage, depuis l'image codée primaire à laquelle le message est associé (incluse), comme spécifié au § 7.4.1.2.3, jusqu'à l'image codée primaire à laquelle est associé (exclu) le prochain (s'il est présent) message SEI d'informations de scène dans l'ordre de décodage ou (sinon) jusqu'à la dernière unité d'accès dans le flux binaire (incluse). Ces images sont désignées ici comme les images cibles.

scene_info_present_flag égal à 0 indique que la scène ou transition de scène à laquelle appartiennent les images cibles n'est pas spécifiée. scene_info_present_flag égal à 1 indique que les images cibles appartiennent à la même scène ou transition de scène.

scene_id identifie la scène à laquelle appartiennent les images cibles. Lorsque la valeur de scene_transition_type des images cibles est inférieure à 4, l'image précédente dans l'ordre de sortie est marquée avec une valeur de scene_transition_type inférieure à 4, et la valeur de scene_id est la même que la valeur de scene_id de l'image précédente dans l'ordre de sortie, ce qui indique que la scène source pour les images cibles et la scène pour l'image précédente (dans l'ordre de sortie) sont considérées par le codeur comme étant de la même scène. Lorsque la valeur de scene_transition_type des images cibles est supérieure à 3, et que l'image précédente dans l'ordre de sortie est marquée avec une valeur de scene_transition_type inférieure à 4, et que la valeur de scene_id est la même que la valeur de scene_id de l'image précédente dans l'ordre de sortie, cela indique qu'une des scènes sources pour les images cibles et la scène source pour l'image précédente (dans l'ordre de sortie) sont considérées par le codeur comme étant de la même scène. Lorsque la valeur de scene_id n'est pas égale à la valeur de scene_id de l'image précédente dans l'ordre de sortie, cela indique que les images cibles et l'image précédente (dans l'ordre de sortie) sont considérées par le codeur comme étant de différentes scènes sources.

La valeur de scene_id doit être dans la gamme de 0 à 2^{32} –1 inclus. Les valeurs de scene_id dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à 2^{31} – 1 inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de scene_id dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à 2^{32} – 1 inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de scene_id dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à 2^{32} – 1 inclus, doivent l'ignorer (la retirer du flux binaire et la détruire).

scene_transition_type spécifie dans quel type de transition de scène (le cas échéant) sont impliquées les images cibles. Les valeurs valides de scene_transition_type sont spécifiées au Tableau D-4.

Tableau D-4 – Valeurs de scene_transition_type

Valeur	Description
0	Pas de transition
1	Fondu vers le noir
2	Fondu à partir du noir
3	Transition non spécifiée à partir d'une ou vers une couleur constante
4	Dissoudre
5	Effacer
6	Mélange non spécifié de deux scènes

Lorsque scene_transition_type est supérieur à 3, les images cibles incluent à la fois des contenus venant de la scène étiquetée par son scene_id et la scène suivante, dans l'ordre de sortie, qui est étiquetée par second_scene_id (voir ci-dessous). Le terme "la scène en cours" est utilisé afin d'indiquer la scène étiquetée par scene_id. Le terme "la scène

suivante" est utilisé afin d'indiquer la scène étiquetée par second_scene_id. Il n'est pas nécessaire pour toute image suivante, dans l'ordre de sortie, d'être étiquetée avec scene_id égal à second_scene_id du message SEI en cours.

Les types de transition de scène sont spécifiés comme suit.

"Pas de transition" spécifie que les images cibles ne sont pas impliquées dans une transition graduelle de scène.

NOTE 2 – Lorsque deux images consécutives dans l'ordre de sortie ont scene_transition_type égal à 0 et des valeurs différentes de scene_id, une coupure de scène est intervenue entre les deux images.

"Fondu vers le noir" indique que les images cibles font partie d'une séquence d'images, dans l'ordre de sortie, impliquées dans une transition de scène en fondu vers le noir, c'est-à-dire que les échantillons luma de la scène s'approchent graduellement de zéro et que les échantillons chroma de la scène s'approchent graduellement de 128.

NOTE 3 – Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur scene_transition_type est "Fondu vers le noir", la dernière, dans l'ordre de sortie, est plus sombre que la précédente.

"Fondu à partir du noir" indique que les images cibles font partie d'une séquence d'images, dans l'ordre de sortie, impliquées dans un fondu à partir d'une transition de scène noire, c'est-à-dire que les échantillons luma de la scène divergent graduellement de zéro et que les échantillons chroma de la scène peuvent diverger graduellement de 128.

NOTE 4 – Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur scene_transition_type est "Fondu à partir du noir", la dernière, dans l'ordre de sortie, est plus claire que la précédente.

"Dissoudre" indique que les valeurs d'échantillon de chaque image cible (avant le codage) ont été produites en calculant une somme de valeurs pondérées d'échantillons colocalisés d'une image de la scène en cours et d'une image de la scène suivante. La pondération de la scène en cours décroît graduellement d'une prise en compte complète vers zéro, tandis que la pondération de la scène suivante s'accroît graduellement de zéro à une prise en compte complète. Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur scene_transition_type est "Dissoudre", la pondération de leur scène en cours par rapport à la dernière, dans l'ordre de sortie, est inférieure à la pondération de leur scène en cours pour la précédente, et la pondération de la scène suivante par rapport à la dernière, dans l'ordre de sortie, est supérieure à la pondération de la prochaine scène par rapport à la précédente.

"Effacer" indique que certaines des valeurs d'échantillon de chaque image cible (avant codage) ont été produites en copiant les valeurs d'échantillons colocalisés d'une image dans la scène en cours et que les valeurs d'échantillon restantes de chaque image cible (avant codage) ont été produites en copiant les valeurs d'échantillons colocalisés d'une image dans la scène suivante. Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur scene_transition_type est "Effacer", le nombre d'échantillons copiés de la scène suivante sur la dernière image dans l'ordre de sortie est supérieur au nombre d'échantillons copiés de la prochaine scène sur l'image précédente.

second_scene_id identifie la prochaine scène dans la transition de scène graduelle dans laquelle sont impliquées les images cibles. La valeur de second_scene_id ne doit pas être égale à la valeur de scene_id. La valeur de second_scene_id ne doit pas être égale à la valeur de scene_id dans l'image précédente dans l'ordre de sortie. Lorsque la prochaine image dans l'ordre de sortie est marquée avec une valeur de scene_transition_type inférieure à 4, et que la valeur de second_scene_id est la même que la valeur de scene_id de l'image suivante dans l'ordre de sortie, cela indique que le codeur considère que l'une des scènes sources pour les images cibles et la scène source pour l'image suivante (dans l'ordre de sortie) constituent la même scène. Lorsque la valeur de second_scene_id n'est pas égale à la valeur de scene_id ou de second_scene_id (s'il est présent) de l'image suivante dans l'ordre de sortie, cela indique que le codeur considère que les images cibles et l'image suivante (dans l'ordre de sortie) proviennent de scènes sources différentes.

Lorsque la valeur de scene_id d'une image est égale à la valeur de scene_id de l'image suivante dans l'ordre de sortie et que la valeur de scene_transition_type dans ces deux images est inférieure à 4, cela indique que le codeur considère que les deux images proviennent de la même scène source. Lorsque les valeurs de scene_id, scene_transition_type et second_scene_id (s'il est présent) d'une image sont (respectivement) égales aux valeurs de scene_id, scene_transition_type et second_scene_id de l'image suivante dans l'ordre de sortie et que la valeur de scene_transition_type est supérieure à 0, cela indique que le codeur considère que les deux images proviennent de la même transition graduelle de scène source.

La valeur de second_scene_id doit être dans la gamme de 0 à $2^{32}-1$ inclus. Les valeurs de second_scene_id dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31}-1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de second_scene_id dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32}-1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de second_scene_id dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32}-1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

D.2.11 Sémantique de message SEI d'informations de sous-séquence

Le message SEI d'informations de sous-séquence est utilisé afin d'indiquer la position d'une image dans une hiérarchie de dépendance des données qui consiste en couches de sous-séquence et sous-séquences.

Une couche de sous-séquence contient un sous-ensemble des images codées dans une séquence. Les couches de sous-séquence sont numérotées avec des entiers non négatifs. Une couche ayant un plus grand numéro de couche est une couche plus élevée qu'une couche ayant un plus petit numéro de couche. Les couches sont ordonnées hiérarchiquement sur la base de leur dépendance les unes par rapport aux autres de sorte que toute image dans une couche ne doit pas être prédite à partir d'une image sur une couche supérieure.

NOTE 1 – En d'autres termes, toute image de la couche 0 ne doit pas être prédite à partir d'une image quelconque de la couche 1 ou au-dessus, les images de la couche 1 peuvent être prédites à partir de la couche 0, les images de la couche 2 peuvent être prédites à partir des couches 0 et 1, etc.

NOTE 2 – La qualité subjective est supposée croître avec le numéro des couches décodées.

Une sous-séquence est un ensemble d'images codées au sein d'une couche de sous-séquence. Une image doit résider dans une couche de sous-séquence et une seule. Aucune image dans une sous-séquence ne doit être prédite à partir d'une image dans une autre sous-séquence dans la même couche de sous-séquence ou d'une couche de sous-séquence supérieure. Une sous-séquence dans la couche 0 peut être décodée indépendamment de toute image qui n'appartient pas à la sous-séquence.

Le message SEI d'informations de sous-séquence concerne l'unité d'accès en cours. L'image codée primaire dans l'unité d'accès est désignée ici comme l'image en cours.

Le message SEI d'informations de sous-séquence ne doit pas être présent à moins que gaps_in_frame_num_value_allowed_flag dans l'ensemble de paramètres de séquence mentionné par l'image associée au message SEI de sous-séquence ne soit égal à 1.

sub_seq_layer_num spécifie le numéro de couche de sous-séquence de l'image en cours. Lorsque sub_seq_layer_num est supérieur à 0, les opérations de commande de gestion de mémoire ne doivent pas être utilisées dans un en-tête de tranche de l'image en cours. Lorsque l'image en cours réside dans une sous-séquence dont la première image dans l'ordre de décodage est à rafraîchissement IDR, la valeur de sub_seq_layer_num doit être égale à 0. Pour une sous-trame de référence non appariée, la valeur de sub_seq_layer_num doit être égale à 0. sub_seq_layer_num doit être dans la gamme de 0 à 255 inclus.

sub_seq_id identifie la sous-séquence au sein d'une couche. Lorsque l'image en cours réside dans une sous-séquence dont la première image dans l'ordre de décodage est à rafraîchissement IDR, la valeur de sub_seq_id doit être la même que la valeur de idr_pic_id de l'image à rafraîchissement IDR. sub_seq_id doit être dans la gamme de 0 à 65535 inclus.

first_ref_pic_flag égal à 1 spécifie que l'image en cours est la première image de référence de la sous-séquence dans l'ordre de décodage. Lorsque l'image en cours n'est pas la première image de la sous-séquence dans l'ordre de décodage, le fanion first_ref_pic_flag doit être égal à 0.

leading_non_ref_pic_flag égal à 1 spécifie que l'image en cours est une image non référentielle précédant une image de référence dans l'ordre de décodage au sein de la sous-séquence ou que la sous-séquence ne contient pas d'images de référence. Lorsque l'image en cours est une image de référence ou que l'image en cours est une image non référentielle succédant au moins à une image de référence dans l'ordre de décodage au sein de la sous-séquence, le fanion leading_non_ref_pic_flag doit être égal à 0.

last_pic_flag égal à 1 indique que l'image en cours est la dernière image de la sous-séquence (dans l'ordre de décodage), y compris toutes les images de référence et non référentielles de la sous-séquence. Lorsque l'image en cours n'est pas la dernière image de la sous-séquence (dans l'ordre de décodage), last_pic_flag doit être égal à 0.

L'image en cours est allouée comme suit à une sous-séquence.

- Si une ou plusieurs des conditions suivantes sont vraies, l'image en cours est la première image d'une sous-séquence dans l'ordre de décodage.
  - aucune image précédente dans l'ordre de décodage n'est étiquetée avec les mêmes valeurs de sub_seq_id et sub_seq_layer_num que l'image en cours;
  - la valeur de leading_non_ref_pic_flag est égale à 1 et la valeur de leading_non_ref_pic_flag est égale à 0 dans l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de sub_seq_id et sub_seq_layer_num que l'image en cours;
  - la valeur de first_ref_pic_flag est égale à 1 et la valeur de leading_non_ref_pic_flag est égale à 0 dans l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de sub_seq_id et sub_seq_layer_num que l'image en cours;
  - la valeur de last_pic_flag est égale à 1 dans l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de sub_seq_id et sub_seq_layer_num que l'image en cours.
- Sinon, l'image en cours appartient à la même sous-séquence que l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de sub_seq_id et sub_seq_layer_num que l'image en cours.

sub_seq_frame_num_flag égal à 0 spécifie que sub_seq_frame_num n'est pas présent. sub_seq_frame_num_flag égal à 1 spécifie que sub_seq_frame_num est présent.

sub_seq_frame_num doit être égal à 0 pour la première image de référence de la sous-séquence et pour toute image non référentielle précédant la première image de référence de la sous-séquence dans l'ordre de décodage. sub_seq_frame_num est de plus soumis aux contraintes suivantes.

– Si l'image en cours n'est pas la seconde sous-trame d'une paire de sous-trames complémentaires, sub_seq_frame_num doit être incrémenté de 1, en opération modulo MaxFrameNum, par rapport à l'image de référence précédente, dans l'ordre de décodage, qui appartient à la sous-séquence.

– Sinon (si l'image en cours est la seconde sous-trame d'une paire de sous-trames complémentaires), la valeur de sub_seq_frame_num doit être la même que la valeur de sub_seq_frame_num pour la première sous-trame de la paire de sous-trames complémentaires.

sub_seq_frame_num doit être dans la gamme de 0 à MaxFrameNum – 1, inclus.

Lorsque l'image en cours est à rafraîchissement IDR, elle doit débuter par une nouvelle sous-séquence dans la couche de sous-séquence 0. Et donc, le sub_seq_layer_num doit être 0, le sub_seq_id doit être différent de la sous-séquence précédente dans la couche de sous-séquence 0, first_ref_pic_flag doit être 1, et leading_non_ref_pic_flag doit être égal à 0.

Lorsque le message SEI d'informations de sous-séquence est présent pour les deux sous-trames codées d'une paire de sous-trames complémentaires, les valeurs de sub_seq_layer_num, sub_seq_id, leading_non_ref_pic_flag et sub_seq_frame_num, lorsqu'ils sont présents, doivent être identiques pour ces deux images.

Lorsque le message SEI d'informations de sous-séquence n'est présent que pour une seule des sous-trames codées d'une paire de sous-trames complémentaires, les valeurs de sub_seq_layer_num, sub_seq_id, leading_non_ref_pic_flag et sub_seq_frame_num, lorsqu'ils sont présents, sont aussi applicables à l'autre sous-trame codée de la paire de sous-trames complémentaires.

D.2.12 Sémantique de message SEI de caractéristiques de couche de sous-séquence

Le message SEI de caractéristiques de couche de sous-séquence spécifie les caractéristiques des couches de sous-séquence.

num_sub_seq_layers_minus1 plus 1 spécifie le nombre de couches de sous-séquence dans la séquence. num_sub_seq_layers_minus1 doit être dans la gamme de 0 à 255 inclus.

Une paire de average_bit_rate et average_frame_rate caractérise chaque couche de sous-séquence. La première paire de average_bit_rate et average_frame_rate spécifie les caractéristiques d'une couche de sous-séquence 0. Lorsqu'elle est présente, la seconde paire spécifie les caractéristiques des couches de sous-séquence 0 et 1 conjointement. Chaque paire dans l'ordre de décodage spécifie les caractéristiques pour une gamme de couches de sous-séquence depuis la couche numéro 0 jusqu'au numéro de couche spécifié par le compteur de boucles de couches. Les valeurs sont effectives depuis le point où elles sont décodées jusqu'à ce que soit décodée une mise à jour des valeurs.

accurate_statistics_flag égal à 1 indique que les valeurs de average_bit_rate et average_frame_rate sont arrondies à partir de valeurs correctes statistiquement. Le fanion accurate_statistics_flag égal à 0 indique que le average_bit_rate et le average_frame_rate sont estimés et peuvent dévier un peu des valeurs correctes.

Lorsque accurate_statistics_flag est égal à 0, la qualité de l'approximation utilisée dans le calcul des valeurs de average_bit_rate et de average_frame_rate est choisie par le processus de codage et n'est pas spécifié par la présente Recommandation | Norme internationale.

average_bit_rate indique le débit binaire moyen en unités de 1000 bit par seconde. Toutes les unités NAL dans la gamme de couches de sous-séquence spécifiées ci-dessus sont prises en compte dans le calcul. Le débit binaire moyen est calculé en fonction du temps de retrait de l'unité d'accès spécifié à l'Annexe C de la Recommandation | Norme internationale. Dans ce qui suit, bTotal est le nombre de bits dans toutes les unités NAL succédant à un message SEI de caractéristiques de couche de sous-séquence (y compris les bits des unités NAL de l'unité d'accès en cours) et précédant l'unité d'accès suivante (dans l'ordre de décodage) y compris un message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement). t_1 est le temps de retrait (en secondes) de l'unité d'accès en cours, et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès (dans l'ordre de décodage) avant le prochain message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement).

Lorsque accurate_statistics_flag est égal à 1, les conditions suivantes doivent être respectées comme suit.

– Si t_1 n'est pas égal à t_2, la condition suivante doit être vraie

$$\text{average\_bit\_rate} == \text{Round}(\text{bTotal} \div ((t_2 - t_1) * 1000))) \quad \text{(D-6)}$$

– Sinon (si t_1 est égal à t_2), la condition suivante doit être vraie

$$average\_bit\_rate == 0 \quad \text{(D-7)}$$

average_frame_rate indique le débit de trame moyen en unités de trames/(256 secondes). Toutes les unités NAL dans la gamme de couches de sous-séquences spécifiée ci-dessus sont prises en compte dans le calcul. Dans ce qui suit, fTotal est le nombre de trames, de paires de sous-trames complémentaires et de sous-trames non appariées entre l'image en cours (incluse) et le prochain message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement). t_1 est le temps de retrait (en secondes) de l'unité d'accès en cours, et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès (dans l'ordre de décodage) avant le prochain message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement).

Lorsque accurate_statistics_flag est égal à 1, les conditions suivantes doivent être respectées comme suit.

– Si t_1 n'est pas égal à t_2, la condition suivante doit être vraie

$$average\_frame\_rate == Round(fTotal * 256 \div (t_2 - t_1)) \quad \text{(D-8)}$$

– Sinon (si t_1 est égal à t_2), la condition suivante doit être vraie

$$average\_frame\_rate == 0 \quad \text{(D-9)}$$

D.2.13 Sémantique de message SEI de caractéristiques de sous-séquence

Le message SEI de caractéristiques de sous-séquence indique les caractéristiques d'une sous-séquence. Il indique aussi la dépendance de l'interprédiction entre les sous-séquences. Ce message doit être contenu dans la première unité d'accès dans l'ordre de décodage de la sous-séquence à laquelle s'applique le message SEI de caractéristiques de sous-séquence. Cette sous-séquence est appelée ici la sous-séquence cible.

sub_seq_layer_num identifie le numéro de couche de sous-séquence de la sous-séquence cible. sub_seq_layer_num doit être dans la gamme de 0 à 255 inclus.

sub_seq_id identifie la sous-séquence cible. sub_seq_id doit être dans la gamme de 0 à 65535 inclus.

duration_flag égal à 0 indique que la durée de la sous-séquence cible n'est pas spécifiée.

sub_seq_duration spécifie la durée de la sous-séquence cible en tic-tac d'une horloge à 90 kHz.

average_rate_flag égal à 0 indique que le débit binaire moyen et le débit de trame moyen de la sous-séquence cible ne sont pas spécifiés.

accurate_statistics_flag indique quel est le degré de confiance des valeurs de average_bit_rate et average_frame_rate. accurate_statistics_flag égal à 1, indique que average_bit_rate et average_frame_rate sont arrondis à partir de valeurs statistiquement correctes. accurate_statistics_flag égal à 0 indique que average_bit_rate et average_frame_rate sont estimés et peuvent dévier des valeurs statistiquement correctes.

average_bit_rate indique le débit binaire moyen en (1000 bit)/s de la sous-séquence cible. Toutes les unités NAL de la sous-séquence cible sont prises en compte dans le calcul. Le débit binaire moyen est calculé en fonction du temps de retrait de l'unité d'accès spécifié au § C.1.2. Dans ce qui suit, nB est le nombre de bits dans toutes les unités NAL dans la sous-séquence. t_1 est le temps de retrait (en secondes) de la première unité d'accès de la sous-séquence (dans l'ordre de décodage), et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès de la sous-séquence (dans l'ordre de décodage).

Lorsque accurate_statistics_flag est égal à 1, les conditions suivantes doivent être satisfaites comme suit.

– Si t_1 n'est pas égal à t_2, la condition suivante doit être vraie

$$average\_bit\_rate == Round(nB \div ((t_2 - t_1) * 1000)) \quad \text{(D-10)}$$

– Sinon (si t_1 est égal à t_2), la condition suivante doit être vraie

$$average\_bit\_rate == 0 \quad \text{(D-11)}$$

average_frame_rate indique le débit de trame moyen en unités de trames/(256 secondes) de la sous-séquence cible. Toutes les unités NAL de la sous-séquence cible sont prises en compte dans le calcul. Le débit de trame moyen est

calculé en fonction du temps de retrait d'unité d'accès spécifié au § C.1.2. Dans ce qui suit, fC est le nombre de trames, de paires de sous-trames complémentaires et de sous-trames non appariées dans la sous-séquence. t_1 est le temps de retrait (en secondes) de la première unité d'accès de la sous-séquence (dans l'ordre de décodage), et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès de la sous-séquence (dans l'ordre de décodage).

Lorsque accurate_statistics_flag est égal à 1, les conditions suivantes doivent être satisfaites comme suit.

– Si t_1 n'est pas égal à t_2, la condition suivante doit être vraie

$$\text{average\_frame\_rate} == \text{Round}(fC * 256 \div (t_2 - t_1)) \quad \text{(D-12)}$$

– Sinon (si t_1 est égal à t_2), la condition suivante doit être vraie

$$\text{average\_frame\_rate} == 0 \quad \text{(D-13)}$$

num_referenced_subseqs spécifie le nombre de sous-séquences qui contiennent des images qui sont utilisées comme images de référence pour l'interprédiction dans les images de la sous-séquence cible. num_referenced_subseqs doit être dans la gamme de 0 à 255 inclus.

ref_sub_seq_layer_num, ref_sub_seq_id, et **ref_sub_seq_direction** identifie la sous-séquence qui contient les images qui sont utilisées comme images de référence pour l'interprédiction dans les images de la sous-séquence cible. En fonction de ref_sub_seq_direction, on applique ce qui suit.

– Si ref_sub_seq_direction est égal à 0, un ensemble de sous-séquences candidates se compose des sous-séquences dont sub_seq_id est égal à ref_sub_seq_id, qui résident dans la couche de sous-séquence qui a sub_seq_layer_num égal à ref_sub_seq_layer_num, et dont la première image dans l'ordre de décodage précède la première image de la sous-séquence cible dans l'ordre de décodage.

– Sinon (si ref_sub_seq_direction est égal à 1), un ensemble de sous-séquences candidates se compose des sous-séquences dont sub_seq_id est égal à ref_sub_seq_id, qui résident dans la couche de sous-séquence qui a sub_seq_layer_num égal à ref_sub_seq_layer_num, et dont la première image dans l'ordre de décodage succède à la première image de la sous-séquence cible dans l'ordre de décodage.

La sous-séquence utilisée comme référence pour la sous-séquence cible est la sous-séquence parmi l'ensemble de sous-séquences candidates dont la première image est la plus proche de la première image de la sous-séquence cible dans l'ordre de décodage.

D.2.14 Sémantique de message SEI de gel de trame entière

Le message SEI de gel de trame entière indique que l'image actuelle et toutes les images subséquentes qui répondent aux conditions spécifiées dans l'ordre de sortie ne devraient pas affecter le contenu de l'affichage. Un seul message SEI au plus de gel de trame entière doit être présent dans une unité d'accès quelconque.

full_frame_freeze_repetition_period spécifie la persistance du message SEI de gel de trame entière et peut spécifier un intervalle de compte d'ordre d'image au sein duquel un autre message SEI de gel de trame entière ou un message SEI de libération de gel de trame entière ou la fin de la séquence vidéo codée doit être présent dans le flux binaire. La valeur de full_frame_freeze_repetition_period doit être dans la gamme de 0 à 16 384 inclus.

full_frame_freeze_repetition_period égal à 0 spécifie que le message SEI de gel de trame entière ne s'applique qu'à l'image décodée en cours.

full_frame_freeze_repetition_period égal à 1 spécifie que le message SEI de gel de trame entière persiste dans l'ordre de sortie tant qu'une des conditions suivantes est vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de gel de trame entière ou un message SEI de libération de gel de trame entière est sortie avec PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic).

full_frame_freeze_repetition_period supérieur à 1 spécifie que le message SEI de gel de trame entière persiste tant qu'une des conditions suivantes est vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de de gel de trame entière ou un message SEI de libération de gel de trame entière est sortie avec PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period.

full_frame_freeze_repetition_period supérieur à 1 indique qu'un autre message SEI de gel de trame entière ou un message SEI de libération de gel de trame entière doit être présent pour une image dans une unité d'accès qui est sortie avec PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period; à moins que le flux binaire ne se termine ou qu'une nouvelle séquence vidéo codée ne commence sans sortie d'une telle image.

D.2.15 Sémantique de message SEI de libération de gel de trame entière

Le message SEI de libération de gel de trame entière annule l'effet de tous les messages SEI de gel de trame entière envoyés avec des images qui précèdent l'image actuelle dans l'ordre de sortie. Le message SEI de libération de gel de trame entière indique que l'image actuelle et les images subséquentes dans l'ordre de sortie devraient affecter le contenu de l'affichage.

Pas plus d'un seul message SEI de libération de gel de trame entière doit être présent dans une quelconque unité d'accès. un message SEI de libération de gel de trame entière ne doit pas être présent dans une unité d'accès contenant un message SEI de gel de trame entière. Lorsqu'un message SEI de gel de trame entière est présent dans une unité d'accès contenant une sous-trame d'un paire de sous-trames complémentaires dans lesquelles les valeurs de PicOrderCnt(CurrPic) pour les deux sous-trames de la paire de sous-trames complémentaires sont égales l'une à l'autre, un message SEI de libération de gel de trame entière ne doit pas être présent dans l'une ou l'autre unité d'accès.

D.2.16 Sémantique de message SEI de photographie de trame entière

Le message SEI de photographie de trame entière indique que la trame en cours est étiquetée pour une utilisation déterminée par l'application en tant que photographie d'image fixe du contenu de la vidéo.

snapshot_id spécifie un numéro d'identification de photographie. snapshot_id doit être dans la gamme de 0 à $2^{32} – 1$ inclus.

Les valeurs de snapshot_id dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31}–1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de snapshot_id dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32}–1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de snapshot_id dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32}–1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

D.2.17 Sémantique de message SEI de début de segment de raffinement progressif

Le message SEI de début de segment de raffinement progressif spécifie le début d'un ensemble d'images codées consécutives qui est étiqueté comme étant l'image en cours suivie par une séquence d'une ou plusieurs images de raffinement de la qualité de l'image en cours, plutôt qu'une représentation d'une scène en mouvement continuel.

L'ensemble étiqueté d'images codées consécutives doit continuer jusqu'à ce qu'une des conditions suivantes soit vraie. Lorsqu'une des conditions ci-dessous devient vraie, la prochaine tranche à décoder n'appartient pas à l'ensemble étiqueté d'images codées consécutives.

1. La prochaine tranche à décoder appartient à uneimage à rafraîchissement IDR.
2. num_refinement_steps_minus1 est supérieur à 0 et le frame_num de la prochaine tranche à décoder est (currFrameNum + num_refinement_steps_minus1 + 1) % MaxFrameNum, où currFrameNum est la valeur de frame_num de l'image dans l'unité d'accès contenant le message SEI.
3. num_refinement_steps_minus1 est à 0 et un message SEI de fin de segment de raffinement progressif avec le même progressive_refinement_id que celui qui est dans ce message SEI est décodé.

L'ordre de décodage d'image au sein de l'ensemble étiqueté d'images consécutives devrait être le même que leur ordre de sortie. **progressive_refinement_id** spécifie un numéro d'identification pour l'opération de raffinement progressif. progressive_refinement_id doit être dans la gamme de 0 à $2^{32} – 1$ inclus.

Les valeurs de progressive_refinement_id dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31} – 1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de progressive_refinement_id dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32}–1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de progressive_refinement_id dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32} – 1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

num_refinement_steps_minus1 spécifie le nombre de trames de référence dans l'ensemble étiqueté d'images codées consécutives comme suit.

– Si num_refinement_steps_minus1 est égal à 0, le nombre de trame de référence dans l'ensemble étiqueté d'images codées consécutives est inconnu.

– Sinon, le nombre de trame de référence dans l'ensemble étiqueté d'images codées consécutives est égal à num_refinement_steps_minus1 + 1.

num_refinement_steps_minus1 doit être dans la gamme de 0 à MaxFrameNum – 1, inclus.

D.2.18 Sémantique de message SEI de fin de segment de raffinement progressif

Le message SEI de fin de segment de raffinement progressif spécifie la fin d'un ensemble d'images codées consécutives qui a été étiqueté au moyen d'un message SEI de début de segment de raffinement progressif comme image initiale suivi d'une séquence d'une ou plusieurs images de raffinement de la qualité de l'image initiale, et se terminant avec l'image en cours.

progressive_refinement_id spécifie un numéro d'identification pour l'opération de raffinement progressif. progressive_refinement_id doit être dans la gamme de 0 à $2^{32} – 1$ inclus.

Le message SEI de fin de segment de raffinement progressif spécifie la fin de tout segment de raffinement progressif ayant commencé antérieurement au moyen d'un message SEI de début de segment de raffinement progressif avec la même valeur de progressive_refinement_id.

Les valeurs de progressive_refinement_id dans la gamme de 0 à 255, inclus, et dans la gamme de 512 à $2^{31} – 1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de progressive_refinement_id dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32} – 1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de progressive_refinement_id dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32} – 1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

D.2.19 Sémantique de message SEI d'ensemble de groupes de tranches à mouvement restreint

Ce message SEI indique que l'interprédiction sur les frontières d'un groupe de tranches est soumise aux contraintes spécifiées ci-dessous. Lorsqu'il est présent, le message ne doit apparaître que lorsqu'il est associé, comme spécifié au § 7.4.1.2.3, avec une unité d'accès IDR.

L'ensemble d'image cible pour ce message SEI contient toutes les images codées primaires consécutives dans l'ordre de décodage débutant avec l'image à rafraîchissement IDR codée primaire associée (incluse) et se terminant avec l'image à rafraîchissement IDR codée primaire suivante (exclue) ou avec la toute dernière image codée primaire dans le flux binaire (inclus) dans l'ordre de décodage lorsqu'il n'y a pas d'image à rafraîchissement IDR codée primaire à la suite. L'ensemble de groupe de tranches est une collection d'un ou plusieurs groupes de tranches, identifiés par l'élément syntaxique slice_group_id[i].

Ce message SEI indique que, pour chaque image dans l'ensemble d'image cible, le processus d'interprédiction est soumis aux contraintes suivantes: pour l'interprédiction de tout échantillon au sein de l'ensemble de groupe de tranches n'est utilisée aucune valeur d'échantillon en dehors de l'ensemble de groupe de tranches, et aucune valeur d'échantillon à une position d'échantillon fractionnaire qui soit calculée au moyen d'une ou plusieurs valeurs d'échantillon en dehors de l'ensemble de groupe de tranches.

num_slice_groups_in_set_minus1 + 1 spécifie le nombre de groupe de tranches dans l'ensemble de groupe de tranches. La gamme admise de num_slice_groups_in_set_minus1 est de 0 à num_slice_groups_minus1, inclus. La gamme admise de num_slice_groups_minus1 est spécifiée à l'Annexe A.

slice_group_id[i **]** identifie le ou les groupes de tranches contenus au sein de l'ensemble de groupe de tranches. La gamme admise est de 0 à num_slice_groups_in_set_minus1, inclus. La taille de l'élément syntaxique slice_group_id[i] est de Ceil(Log2(num_slice_groups_minus1 + 1)) bits.

exact_sample_value_match_flag égal à 0 indique que, au sein de l'ensemble d'image cible, lorsque les macroblocs qui n'appartiennent pas à l'ensemble de groupe de tranches ne sont pas décodés, la valeur de chaque échantillon dans l'ensemble de groupe de tranches ne doit pas nécessairement être exactement la même que la valeur du même échantillon lorsque tous les macroblocs sont décodés. exact_sample_value_match_flag égal à 1 indique que, au sein de l'ensemble d'image cible, lorsque les macroblocs qui n'appartiennent pas à l'ensemble de groupe de tranches ne sont pas décodés, la valeur de chaque échantillon dans l'ensemble de groupe de tranches doit être exactement la même que la valeur du même échantillon lorsque sont décodés tous les macroblocs dans l'ensemble d'image cible.

NOTE 1 – Lorsque disable_deblocking_filter_idc est égal à 2 dans toutes les tranches dans l'ensemble d'image cible, exact_sample_value_match_flag devrait être 1.

pan_scan_rect_flag égal à 0 spécifie que pan_scan_rect_id n'est pas présent. pan_scan_rect_flag égal à 1 spécifie que pan_scan_rect_id est présent.

pan_scan_rect_id indique que l'ensemble de groupe de tranches spécifié couvre au moins le rectangle de balayage complet identifié par pan_scan_rect_id au sein de l'ensemble d'image cible.

NOTE 2 – Plusieurs messages SEI motion_constrained_slice_group_set peuvent être associés à la même image à rafraîchissement IDR. Par conséquent, plus d'un ensemble de groupe de tranches peut être actif au sein d'un ensemble d'image cible.

NOTE 3 – La taille, forme et localisation des groupes de tranches dans l'ensemble de groupe de tranches peut changer au sein de l'ensemble d'image cible.

D.2.20 Sémantique de message SEI de caractéristiques de grain d'émulsion

Ce message SEI fournit au décodeur avec un modèle paramétré pour la synthèse des grains d'émulsion. Par exemple, un codeur peut utiliser le message SEI de caractéristiques de grain d'émulsion afin de caractériser le grain d'émulsion qui était présent dans le matériel original de la source vidéo et qui a été supprimé par des techniques de filtrage lors d'un prétraitement. La synthèse de grain simulé d'émulsion sur les images décodées pour le processus d'affichage est facultative et n'a pas d'incidence sur le processus de décodage spécifié dans la présente Recommandation | Norme internationale. Si la synthèse de grain simulé d'émulsion sur les images décodées pour le processus d'affichage est effectuée, il n'y a aucune exigence que la méthode par laquelle la synthèse est effectuée soit la même que le modèle paramétré pour le grain d'émulsion tel que fourni dans le message SEI de caractéristiques de grain d'émulsion.

NOTE 1 – Le processus d'affichage n'est pas spécifié dans la présente Recommandation | Norme internationale.

film_grain_characteristics_cancel_flag égal à 1 indique que le message SEI annule la persistance de tout précédent message SEI de caractéristiques de grain d'émulsion dans l'ordre de sortie. Un fanion film_grain_characteristics_cancel_flag égal à 0 indique que des informations de modélisation de grain d'émulsion font suite.

model_id identifie le modèle de simulation de grain d'émulsion spécifié dans le Tableau D-5. La valeur de model_id doit être dans l'étendue de 0 à 1, inclus.

Tableau D-5 – Valeurs de model_id

Valeur	Description
0	filtrage de fréquence
1	autorégression
2	réservé
3	réservé

separate_colour_description_present_flag égal à 1 indique qu'une description distincte d'espace chromatique pour les caractéristiques de grain d'émulsion spécifiées dans le message SEI est présente dans la syntaxe de message SEI de caractéristiques de grain d'émulsion. Un fanion separate_colour_description_present_flag égal à 0 indique que la description d'espace chromatique pour les caractéristiques de grain d'émulsion spécifiées dans le message SEI est la même que pour la séquence vidéo codée spécifiée dans le § E.2.1.

NOTE 2 – Lorsque le fanion separate_colour_description_present_flag est égal à 1, l'espace chromatique spécifié pour les caractéristiques de grain d'émulsion spécifiées dans le message SEI peut différer de l'espace chromatique spécifié pour la vidéo codée spécifiée dans le § E.2.1.

film_grain_bit_depth_luma_minus8 plus 8 spécifie la profondeur de bits utilisée pour la composante luma des caractéristiques de grain d'émulsion spécifiées dans le message SEI. Lorsque l'élément film_grain_bit_depth_luma_minus8 n'est pas présent dans le message SEI de caractéristiques de grain d'émulsion, la valeur de film_grain_bit_depth_luma_minus8 doit être présumée égale à bit_depth_luma_minus8.

La valeur de filmGrainBitDepth[0] est calculée comme suit:

$$filmGrainBitDepth[\ 0\ ] = film\_grain\_bit\_depth\_luma\_minus8 + 8 \quad \text{(D-14)}$$

film_grain_bit_depth_chroma_minus8 plus 8 spécifie la profondeur de bits utilisée pour les composantes Cb et Cr des caractéristiques de grain d'émulsion spécifiées dans le message SEI. Lorsque film_grain_bit_depth_chroma_minus8 n'est pas présent dans le message SEI de caractéristiques de grain d'émulsion, la valeur de film_grain_bit_depth_chroma_minus8 doit être présumée égale à bit_depth_chroma_minus8.

La valeur de filmGrainBitDepth[c] pour c = 1 et 2 est calculée comme suit:

$$filmGrainBitDepth[\ c\ ] = film\_grain\_bit\_depth\_chroma\_minus8 + 8 \quad \text{with } c = 1, 2 \quad \text{(D-15)}$$

film_grain_full_range_flag a la même sémantique que spécifié dans le § E.2.1 pour l'élément syntaxique video_full_range_flag, à l'exception de ce qui suit:

– film_grain_full_range_flag spécifie l'espace chromatique des caractéristiques de grain d'émulsion spécifiées dans le message SEI, plutôt que l'espace chromatique utilisé pour la séquence vidéo codée.

– Lorsque film_grain_full_range_flag n'est pas présent dans le message SEI de caractéristiques de grain d'émulsion, la valeur de film_grain_full_range_flag doit être présumée égale à video_full_range_flag.

film_grain_colour_primaries a la même sémantique que spécifié dans le § E.2.1 pour l'élément syntaxique colour_primaries, à l'exception de ce qui suit:

– film_grain_colour_primaries spécifie l'espace chromatique des caractéristiques de grain d'émulsion spécifiées dans le message SEI, plutôt que l'espace chromatique utilisé pour la séquence vidéo codée.

– Lorsque film_grain_colour_primaries n'est pas présent dans le message SEI de caractéristiques de grain d'émulsion, la valeur de film_grain_colour_primaries doit être présumée égale à colour_primaries.

film_grain_transfer_characteristics a la même sémantique que spécifié dans le § E.2.1 pour l'élément syntaxique transfer_characteristics, à l'exception de ce qui suit:

– film_grain_transfer_characteristics spécifie l'espace chromatique des caractéristiques de grain d'émulsion spécifiées dans le message SEI, plutôt que l'espace chromatique utilisé pour la séquence vidéo codée.

– Lorsque film_grain_transfer_characteristics n'est pas présent dans le message SEI de caractéristiques de grain d'émulsion, la valeur de film_grain_transfer_characteristics doit être présumée égale à transfer_characteristics.

film_grain_matrix_coefficients a la même sémantique que spécifié dans le § E.2.1 pour l'élément syntaxique matrix_coefficients, à l'exception de ce qui suit:

– film_grain_matrix_coefficients spécifie l'espace chromatique des caractéristiques de grain d'émulsion spécifiées dans le message SEI, plutôt que l'espace chromatique utilisé pour la séquence vidéo codée.

– Lorsque film_grain_matrix_coefficients n'est pas présent dans le message SEI de caractéristiques de grain d'émulsion, la valeur de film_grain_matrix_coefficients doit être présumée égale à matrix_coefficients.

– Les valeurs autorisées pour film_grain_matrix_coefficients ne sont pas contraintes par la valeur de chroma_format_idc.

L'élément chroma_format_idc des caractéristiques de grain d'émulsion spécifiées dans le message SEI de caractéristiques de grain d'émulsion doit être présumé égale à 3 (4:4:4).

NOTE 3 – Etant donné que l'utilisation d'une méthode spécifique n'est pas requise pour l'exécution de la fonction de production de grains d'émulsion utilisée par le processus d'affichage, un décodeur peut, au besoin, sous-échantillonner les informations de modélisation pour les échantillons chroma afin de simuler un grain d'émulsion pour d'autres formats chromatiques (4:2:0 ou 4:2:2) plutôt que suréchantillonner la vidéo décodée (au moyen d'une méthode non spécifiée par la présente Recommandation | Norme internationale) avant l'exécution de la production de grains d'émulsion.

blending_mode_id identifie le mode de fusion servant à superposer le grain simulé d'émulsion aux images décodées comme spécifié dans le Tableau D-6. La valeur de l'élément blending_mode_id doit être dans l'étendue de 0 à 1, inclus.

Tableau D-6 – Valeur de blending_mode_id

Valeur	Description
0	additif
1	multiplicatif
2	réservé
3	réservé

Selon blending_mode_id, le mode de fusion est spécifié comme suit:

– Si blending_mode_id est égal à 0 le mode de fusion est additif, comme spécifié par:

$$I_{grain}[x, y, c] = Clip3(0, (1 << filmGrainBitDepth[c]) – 1, I_{decoded}[x, y, c] + G[x, y, c]) \quad \text{(D-16)}$$

– Sinon (si blending_mode_id est égal à 1), le mode de fusion est multiplicatif, comme spécifié par:

$$I_{grain}[x, y, c] = Clip3(0, (1 << filmGrainBitDepth[c]) – 1, I_{decoded}[x, y, c] * (1 + G[x, y, c])) \quad \text{(D-17)}$$

où $I_{decoded}[x, y, c]$ représente la valeur d'échantillon à coordonnées x, y de la composante chromatique c de l'image décodée $I_{decoded}$, G[x, y, c] est la valeur du grain simulé d'émulsion à la même position et à la même composante

chromatique, et filmGrainBitDepth[c] est le nombre de bits utilisés pour chaque échantillon dans une représentation binaire non signée de longueur fixe de la matrice I_{grain}[x, y, c].

log2_scale_factor spécifie un facteur de normalisation utilisé dans les équations de caractérisation du grain d'émulsion.

comp_model_present_flag[c] égal à 0 indique que le grain d'émulsion n'est pas modélisé sur la c-ième composante chromatique, où c égal à 0 se rapporte à la composante luma, c égal à 1 se rapporte à la composante Cb, et c égal à 2 se rapporte à la composante Cr. Le fanion comp_model_present_flag[c] égal à 1 indique que les éléments syntaxiques spécifiant la modélisation du grain d'émulsion sur la composante chromatique c sont présents dans le message SEI.

num_intensity_intervals_minus1[c] plus 1 spécifie le nombre d'intervalles d'intensité pour lequel un ensemble spécifique de valeurs de modèle a été estimé.

NOTE 4 – Ls intervalles d'intensité peuvent se chevaucher afin de simuler une production multiple de grains d'émulsion.

num_model_values_minus1[c] plus 1 spécifie le nombre de valeurs de modèle présentes pour chaque intervalle d'intensité dans lequel le grain d'émulsion a été modélisé. La valeur de num_model_values_minus1[c] doit être dans l'étendue de 0 à 5, inclus.

intensity_interval_lower_bound[c][i] spécifie la limite inférieure de l'intervalle i entre niveaux d'intensité auquel l'ensemble de valeurs de modèle est applicable.

intensity_interval_upper_bound[c][i] spécifie la limite supérieure de l'intervalle i entre niveaux d'intensité auquel l'ensemble de valeurs de modèle est applicable.

Selon model_id, la sélection des ensembles de valeurs de modèle est spécifiée comme suit:

– Si model_id est égal à 0, la valeur moyenne de chaque bloc b de 16x16 échantillons dans $I_{decoded}$, désignée par b_{avg}, sert à sélectionner les ensembles de valeurs de modèle ayant l'indice s[j] qui s'appliquent à tous les échantillons contenus dans le bloc:

```
pour( i = 0, j = 0; i <= num_intensity_intervals_minus1; i++ )
   si( b_avg >= intensity_interval_lower_bound[ c ][ i ] && b_avg <= intensity_interval_upper_bound[ c ][ i ] ) {
    s[ j ] = i                                                                                    (D-18)
    j++
   }
```

– Sinon (si model_id est égal à 1), les ensembles de valeurs de modèle utilisés afin de produire le grain d'émulsion sont sélectionnés pour chaque valeur d'échantillon contenue dans $I_{decoded}$ comme suit:

```
pour( i = 0, j = 0; i <= num_intensity_intervals_minus1; i++ )
   si( I_decoded[ x, y, c ] >= intensity_interval_lower_bound[ c ][ i ] &&
       I_decoded[ x, y, c ] <= intensity_interval_upper_bound[ c ][ i ] ) {                       (D-19)
    s[ j ] = i
    j++
   }
```

Les échantillons qui ne tombent pas dans tous les intervalles définis ne sont pas modifiés par la fonction de production de grains d'émulsion. Les échantillons qui tombent dans plus d'un seul intervalle deviendront des grains multiproduits: ceux-ci proviennent de l'adjonction de grains calculés indépendamment pour chaque intervalle d'intensité.

comp_model_value[c][i][j] représente chacune des valeurs de modèle présentes pour la composante chromatique c et pour l'intervalle d'intensité i. L'ensemble des valeurs de modèle a une signification différente selon la valeur de model_id. La valeur de comp_model_value[c][i][j] doit être contrainte comme suit et peut faire l'objet d'une contrainte supplémentaire comme spécifié ailleurs dans le présent paragraphe.

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][j] doit être dans l'étendue de 0 à $2^{filmGrainBitDepth[\ c\ ]} - 1$, inclus.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][j] doit être dans l'étendue de $-2^{(\ filmGrainBitDepth[\ c\ ] - 1\ )}$ à $2^{(\ filmGrainBitDepth[\ c\ ] - 1\ )} - 1$, inclus.

Selon model_id, la synthèse du grain d'émulsion est modélisée comme suit:

– Si model_id est égal à 0, un modèle à filtrage par fréquence permet de simuler le grain original d'émulsion pour c = 0..2, x = 0..$PicWidthInSamples_L$, et y = 0..$PicHeightInSamples_L$ comme spécifié par:

```
G[ x, y, c ] = ( comp_model_value[ c ][ s ][ 0 ] * Q[ x, y, c ] + comp_model_value[ c ][ s ][ 5 ] *
G[ x, y, c-1 ] ) >> log2_scale_factor                                                    (D-20)
```

où Q[c] est un processus aléatoire bidimensionnel qui est produit par filtrage de blocs 16x16 gaussRv dont les éléments de valeur aléatoire $gaussRv_{ij}$ sont produits selon une répartition gaussienne normalisée (d'échantillons de variable aléatoire indépendants et à répartition gaussienne uniforme avec moyenne nulle et variance égale à l'unité) et où la valeur d'un élément G[x, y, c-1], utilisée à droite de l'équation, est présumée égale à 0 lorsque c-1 est inférieur à 0.

NOTE 5 – Une valeur aléatoire gaussienne normalisée peut être produite par deux valeurs aléatoires indépendantes, réparties uniformément dans l'intervalle de 0 à 1 (et non égales à 0), désignées par uRv_0 et uRv_1, au moyen de la transformation de Box-Muller spécifiée par:

$$gaussRv_{ij} = \sqrt{-2 * Ln(uRv_0)} * Cos(2 * \pi * uRv_1) \qquad \text{(D-21)}$$

où Ln(x) est le logarithme naturel de x (le logarithme de base e, où e est la constante de base 2,718 281 828...du logarithme naturel), Cos(x) est la fonction trigonométrique cosinusoïdale opérant sur un argument x exprimé en unités de radians, et où π est la constante d'Archimède 3,141 592 653....

Le filtrage passe-bande des blocs gaussRv peut être effectué comme suit dans le domaine de la transformation en cosinus discrets (DCT):

```
pour( y = 0; y < 16; y++ )
    pour( x = 0; x < 16; x++ )
        si( ( x < comp_model_value[ c ][ s ][ 3 ]  &&  y < comp_model_value[ c ][ s ][ 4 ] )  ||        (D-22)
          x > comp_model_value[ c ][ s ][ 1 ]  ||  y > comp_model_value[ c ][ s ][ 2 ] )
            gaussRv[ x, y ] = 0
filteredRv = IDCT16x16( gaussRv )
```

où IDCT16x16(z) se rapporte à une transformation inverse en cosinus discrets (IDCT) unitaire opérant sur un argument matriciel 16x16 z comme spécifié par:

$$IDCT16x16(z) = r * z * r^T \qquad \text{(D-23)}$$

où l'indice supérieur T indique une transposition de matrice et où r est la matrice 16x16 avec les éléments r_{ij} spécifiés par:

$$r_{ij} = \frac{((i == 0)\ ?\ 1 : \sqrt{2})}{4} Cos\left(\frac{i*(2*j+1)*\pi}{32}\right) \qquad \text{(D-24)}$$

où Cos(x) est le fonction trigonométrique cosinusoïdale opérant sur un argument x exprimé en unités de radians et où π est La constante d'Archimède 3,141 592 653.

Q[c] est formé par le blocs filtrés en fréquence filteredRv.

NOTE 6 – Les valeurs de modèle codées sont sur la base de blocs de 16x16, mais une mise en œuvre de décodeur peut utiliser d'autres tailles de blocs. Par exemple, les décodeurs mettant en œuvre la transformation IDCT sur des blocs de 8x8 échantillons, devraient sous-échantillonner d'un facteur de deux l'ensemble des valeurs de modèle codées comp_model_value[c][s][i] pour i égal à 1..4.

NOTE 7 – Afin de réduire le degré de visibilité des blocs qui peut résulter de la mise en mosaïque des blocs filtrés en fréquence filteredRv, les décodeurs peuvent appliquer un filtre passe-bas aux frontières entre blocs filtrés en fréquence.

– Sinon (si model_id est égal à 1), un modèle d'autorégression permet de simuler le grain original d'émulsion pour c = 0..2, x = 0..$PicWidthInSamples_L$, et y = 0..$PicHeightInSamples_L$ comme spécifié par:

```
G[ x, y, c ] = ( comp_model_value[ c ][ s ][ 0 ] * n[ x, y, c ] +
    comp_model_value[ c ][ s ][ 1 ] * ( G[ x-1, y, c ] + ( ( comp_model_value[ c ][ s ][ 4 ] * G[ x, y-1, c ] )  >>
        log2_scale_factor ) ) +
    comp_model_value[ c ][ s ][ 3 ] * ( ( ( comp_model_value[ c ][ s ][ 4 ] * G[ x-1, y-1, c ] ) >>
        log2_scale_factor ) + G[ x+1, y-1, c ] ) +
    comp_model_value[ c ][ s ][ 5 ] * ( G[ x-2, y, c ] +
        ( ( comp_model_value[ c ][ s ][ 4 ] * comp_model_value[ c ][ s ][ 4 ] * G[ x, y-2, c ] ) >>
            ( 2 * log2_scale_factor ) ) ) +
    comp_model_value[ c ][ s ][ 2 ] * G[ x, y, c-1 ] ) >> log2_scale_factor                  (D-25)
```

où n[x, y, c] est une valeur aléatoire à répartition normale gaussienne (d'échantillons indépendants et uniformément répartis de variables aléatoires gaussiennes pour chaque valeur de x, y, et c) et où la valeur d'un élément G[x, y, c] utilisée du côté droit de l'équation est présumée égale à 0 lorsque la totalité des conditions suivantes soient vraies:

– x est inférieur à 0,

– y est inférieur à 0,

– x est supérieur ou égal à $PicWidthInSamples_L$,

– c est inférieur à 0.

comp_model_value[c][i][0] fournit la première valeur pour le modèle spécifié par model_id. comp_model_value[c][i][0] correspond à l'écart type du terme de bruit blanc gaussien dans les fonctions de production spécifiées dans les équations D-20 à D-23.

comp_model_value[c][i][1] fournit la seconde valeur pour le modèle spécifié par model_id. comp_model_value[c][i][1] doit être supérieur ou égal à 0 et inférieur à 16.

Si elle est absente dans le message SEI de caractéristiques de grain d'émulsion, la valeur de comp_model_value[c][i][1] doit être présumée comme suit.

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][1] doit être présumée égale à 8.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][1] doit être présumée égale à 0.

comp_model_value[c][i][1] est interprété comme suit.

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][1] indique la fréquence supérieure de coupure horizontale à utiliser afin de filtrer la transformée DCT d'un bloc de 16x16 valeurs aléatoires.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][1] indique la corrélatin spatiale du premier ordre pour échantillons voisins (x-1, y) et (x, y-1).

comp_model_value[c][i][2] fournit le troisième valeur pour le modèle spécifié par model_id. comp_model_value[c][i][2] doit être supérieur ou égal à 0 et inférieur à 16.

Si elle est absente dans le message SEI de caractéristiques de grain d'émulsion, la valeur de comp_model_value[c][i][2] doit être présumée comme suit:

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][2] doit être présumée égale à comp_model_value[c][i][1]

– Sinon (si model_id est égal à 0), la valeur de comp_model_value[c][i][2] doit être présumée égale à 0.

comp_model_value[c][i][2] est interprété comme suit:

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][2] indique la fréquence supérieure de coupure verticale à utiliser afin de filtrer la transformée DCT d'un bloc de 16x16 valeurs aléatoires.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][2] indique la corrélation chromatique entre composantes chromatiques consécutives.

comp_model_value[c][i][3] fournit la quatrième valeur pour le modèle spécifié par model_id. comp_model_value[c][i][3] doit être supérieur ou égal à 0 et inférieur ou égal à comp_model_value[c][i][1].

Si elle est absente dans le message SEI de caractéristiques de grain d'émulsion, la valeur de comp_model_value[c][i][3] doit être présumée égale à 0.

comp_model_value[c][i][3] est interprété comme suit.

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][3] indique la fréquence inférieure de coupure horizontale à utiliser afin de filtrer la transformée DCT d'un bloc de 16x16 valeurs aléatoires.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][3] indique la corrélation spatiale du premier ordre pour échantillons voisins (x-1, y-1) et (x+1, y-1).

comp_model_value[c][i][4] fournit la cinquième valeur pour le modèle spécifié par model_id. comp_model_value[c][i][4] doit être supérieur ou égal à 0 et inférieur ou égal à comp_model_value[c][i][2].

Si elle est absente dans le message SEI de caractéristiques de grain d'émulsion, la valeur de comp_model_value[c][i][4] doit être présumée égale à model_id.

comp_model_value[c][i][4] est interprété comme suit.

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][4] indique la fréquence inférieure de coupure verticale à utiliser afin de filtrer la transformée DCT d'un bloc de 16x16 valeurs aléatoires.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][4] indique le format du grain modélisé.

comp_model_value[c][i][5] fournit la sixième valeur pour le modèle spécifié par model_id. Si elle est absente dans le message SEI de caractéristiques de grain d'émulsion, la valeur de comp_model_value[c][i][5] doit être présumée égale à 0.

comp_model_value[c][i][5] est interprété comme suit:

– Si model_id est égal à 0, la valeur de comp_model_value[c][i][5] indique la corrélation chromatique entre composantes chromatiques consécutives.

– Sinon (si model_id est égal à 1), la valeur de comp_model_value[c][i][5] indique la corrélation spatiale du second ordre pour échantillons voisins (x, y-2) et (x-2, y).

film_grain_characteristics_repetition_period spécifie la persistance du message SEI de caractéristiques de grain d'émulsion et peut spécifier un intervalle de comptage ordonné d'image à l'intérieur duquel un autre message SEI de caractéristiques de grain d'émulsion ou la fin de la séquence vidéo codée doit être présent dans le flux binaire. La valeur de film_grain_characteristics_repetition_period doit être dans l'étendue 0 à 16 384, inclus.

film_grain_characteristics_repetition_period égal à 0 spécifie que le message SEI de caractéristiques de grain d'émulsion s'applique seulement à l'image décodée actuelle.

film_grain_characteristics_repetition_period égal à 1 spécifie que le message SEI de caractéristiques de grain d'émulsion persiste dans l'ordre de sortie jusqu'à ce que la totalité des conditions suivantes soient vraies.

– Une nouvelle séquence vidéo codée commence, ou

– Une image dans une unité d'accès contenant un message SEI de caractéristiques de grain d'émulsion en sortie qui utilise PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic).

film_grain_characteristics_repetition_period supérieur à 1 spécifie que le message SEI de caractéristiques de grain d'émulsion persiste jusqu'à ce que la totalité des conditions suivantes soient vraies.

– Une nouvelle séquence vidéo codée commence, ou

– Une image dans une unité d'accès contenant un message SEI de caractéristiques de grain d'émulsion en sortie qui utilise PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + film_grain_characteristics_repetition_period.

film_grain_characteristics_repetition_period supérieur à 1 indique qu'un autre message SEI de caractéristiques de grain d'émulsion doit être présent pour une image dans une unité d'accès en sortie qui utilise PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + film_grain_characteristics_repetition_period; à moins que le flux binaire ne se termine ou qu'une nouvelle séquence vidéo codée ne commence sans sortie d'une telle image.

D.2.21 Sémantique de message SEI de préférence d'affichage du filtre de démontage de bloc

Ce message SEI fournit au décodeur une indication sur la question de savoir si l'affichage du résultat recadré du processus de filtre de démontage de bloc spécifié dans le § 8.7 ou du résultat recadré du processus de construction d'image avant le processus de filtre de démontage de bloc spécifié dans le § 8.5.12 est préféré par le codeur pour l'affichage de chaque image décodée en sortie.

NOTE 1 – Le processus d'affichage n'est pas spécifié dans la présente Recommandation | Norme internationale. Le moyen par lequel un codeur détermine ce qu'il faut indiquer comme étant sa préférence (exprimée dans un message SEI de préférence d'affichage du filtre de démontage de bloc) n'est pas non plus spécifié dans la présente Recommandation | Norme internationale. L'expression d'une préférence dans un message SEI de préférence d'affichage du filtre de démontage de bloc n'impose aucune exigence quant au processus d'affichage.

deblocking_display_preference_cancel_flag égal à 1 indique que le message SEI annule la persistance de tout précédent message SEI de préférence d'affichage du filtre de démontage de bloc dans l'ordre de sortie. deblocking_display_preference_cancel_flag égal à 0 indique que des fanions display_prior_to_deblocking_preferred_flag et deblocking_display_preference_repetition_period font suite.

NOTE 2 – En l'absence du message SEI de préférence d'affichage du filtre de démontage de bloc, ou après réception d'un message SEI de préférence d'affichage du filtre de démontage de bloc dans lequel le fanion deblocking_display_preference_cancel_flag est égal à 1, le décodeur devrait en déduire que l'affichage du résultat recadré du processus de filtre de démontage de bloc spécifié dans le § 8.7 est préféré à l'affichage du résultat recadré du processus de construction d'image avant le processus de filtre de démontage de bloc spécifié dans le § 8.5.12 pour l'affichage de chaque image décodée en sortie.

display_prior_to_deblocking_preferred_flag égal à 1 indique que la préférence du codeur est que le processus d'affichage (qui n'est pas spécifié dans la présente Recommandation | Norme internationale) affiche le résultat recadré du processus de construction d'image avant le processus de filtre de démontage de bloc spécifié dans le § 8.5.12 plutôt que le résultat recadré du processus de filtre de démontage de bloc spécifié dans le § 8.7 pour chaque image qui est recadrée et sortie comme spécifié dans l'Annexe C. Un fanion display_prior_to_deblocking_preferred_flag égal à 0 indique que la préférence du codeur est que le processus d'affichage (qui n'est pas spécifié dans la présente Recommandation | Norme internationale) affiche le résultat recadré du processus de filtre de démontage de bloc spécifié dans le § 8.7 plutôt que le résultat recadré du processus de construction d'image avant le processus de filtre de démontage de bloc spécifié dans le § 8.5.12 pour chaque image qui est recadrée et sortie comme spécifié dans l'Annexe C.

NOTE 3 – La présence ou l'absence du message SEI de préférence d'affichage du filtre de démontage de bloc et la valeur du fanion display_prior_to_deblocking_preferred_flag n'ont pas d'incidence sur les exigences du processus de décodage spécifié dans la présente Recommandation | Norme internationale. En revanche, il ne fournit qu'une indication du moment où, en plus de la satisfaction des exigences de la présente Recommandation | Norme internationale pour le processus de décodage, une qualité visuelle améliorée pourra être obtenue par exécution du processus d'affichage (qui n'est pas spécifié dans la présente Recommandation | Norme internationale) d'une autre façon. Les codeurs qui utilisent le message SEI de préférence d'affichage du filtre de démontage de bloc devraient être conçus de façon à être informés du fait que, à moins que le codeur ne limite son utilisation de la capacité du tampon DPB spécifiée dans l'Annexe A pour le profil et le niveau en cours d'utilisation, certains décodeurs peuvent ne pas avoir une capacité de mémoire suffisante pour le stockage du résultat du processus de construction d'image avant le processus de filtre de démontage de bloc spécifié dans le § 8.5.12 en plus du stockage du résultat du processus de filtre de démontage de bloc spécifié dans le § 8.7 lors du réordonnancement et du retardement d'images à afficher, et de tels décodeurs ne seront donc pas en mesure de bénéficier de l'indication de préférence. C'est en limitant son utilisation de la capacité du tampon DPB qu'un codeur peut être en mesure d'utiliser au moins la moitié de la capacité du tampon DPB spécifiée dans l'Annexe A tout en permettant au décodeur d'utiliser la capacité restante pour le stockage d'images non filtrées qui ont été indiquées comme étant préférables pour l'affichage avant que l'instant de sortie de ces images soit arrivé.

dec_frame_buffering_constraint_flag égal à 1 indique que l'utilisation de la capacité du décodeur HRD à mettre des trames en mémoire tampon d'images décodées (DPB), spécifiée par max_dec_frame_buffering a été contrainte de façon que la séquence vidéo codée ne nécessite pas de tampon d'images décodées avec plus de Max(1, max_dec_frame_buffering) tampons de trames afin de permettre la sortie des images décodées avec ou sans filtrage, comme indiqué par les messages SEI de préférence d'affichage du filtre de démontage de bloc, aux instants de sortie spécifiés par l'élément DPB_output_delay des messages SEI de synchronisation d'image. Un fanion dec_frame_buffering_constraint_flag égal à 0 indique que l'utilisation de la capacité de mise des trames en mémoire tampon dans le décodeur HRD peut, le cas échéant, être contrainte de la façon qui sera indiquée par un fanion dec_frame_buffering_constraint_flag égal à 1.

Aux fins de la détermination de la contrainte imposée lorsque le fanion dec_frame_buffering_constraint_flag est égal à 1, la grandeur de la capacité de mise des trames en mémoire tampon utilisée à un moment quelconque par chaque tampon de trame de la mémoire DPB qui contient une image doit être calculée comme suit:

– Si les deux critères suivants sont satisfaits pour le tampon de trame, celui-ci est considéré comme utilisant une capacité de deux tampons de trames pour son stockage.
  – Le tampon de trame contient une trame ou une ou plusieurs sous-trames marquée(s) "utilisée comme référence",
  – Le tampon de trame contient une image pour laquelle les deux critères suivants sont satisfaits.
    – L'instant de sortie de l'image du décodeur HRD est supérieur à l'instant indiqué, et
    – Il a été indiqué dans un message SEI de préférence d'affichage du filtre de démontage de bloc que la préférence du codeur pour l'image est que le processus d'affichage affiche le résultat recadré du processus de construction d'image avant le processus de filtre de démontage de bloc spécifié dans le § 8.5.12, plutôt que le résultat recadré du processus de filtre de démontage de bloc spécifié dans le § 8.7, et
– Sinon, le tampon de trame est considéré comme utilisant une capacité d'un seul tampon de trame du tampon DPB pour son stockage.

Lorsque le fanion dec_frame_buffering_constraint_flag est égal à 1, la capacité de mise des trames en mémoire tampon utilisée par tous les tampons de trame dans le tampon DPB qui contiennent des images, calculée de cette façon, ne doit pas être supérieure à Max(1, max_dec_frame_buffering) pendant le fonctionnement du décodeur HRD pour la séquence vidéo codée.

La valeur du fanion dec_frame_buffering_constraint_flag doit être la même dans tous les messages SEI de préférence d'affichage du filtre de démontage de bloc de la séquence vidéo codée.

deblocking_display_preference_repetition_period spécifie la persistance du message SEI de caractéristiques de grain d'émulsion et peut spécifier un intervalle de comptage ordonné d'image à l'intérieur duquel un autre message SEI de

caractéristiques de grain d'émulsion ou la fin de la séquence vidéo codée doit être présent dans le flux binaire. La valeur de deblocking_display_preference_repetition_period doit être dans l'étendue 0 à 16 384, inclus.

deblocking_display_preference_repetition_period égal à 0 spécifie que le message SEI de préférence d'affichage du filtre de démontage de bloc s'applique seulement à l'image décodée actuelle.

deblocking_display_preference_repetition_period égal à 1 spécifie que le message SEI de préférence d'affichage du filtre de démontage de bloc persiste dans l'ordre de sortie jusqu'à ce que la totalité des conditions suivantes soient vraies.

– Une nouvelle séquence vidéo codée commence.

– Une image dans une unité d'accès contenant un message SEI de préférence d'affichage du filtre de démontage de bloc qui utilise PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) est sortie.

deblocking_display_preference_repetition_period supérieur à 1 spécifie que le message SEI de préférence d'affichage du filtre de démontage de bloc persiste jusqu'à ce que la totalité des conditions suivantes soient vraies.

– Une nouvelle séquence vidéo codée commence;

– Une image dans une unité d'accès contenant un message SEI de préférence d'affichage du filtre de démontage de bloc qui utilise PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + deblocking_ display_ preference _repetition period est sortie.

deblocking_display_preference_repetition_period supérieur à 1 indique qu'un autre message SEI de préférence d'affichage du filtre de démontage de bloc doit être présent pour une image dans une unité d'accès en sortie qui utilise PicOrderCnt( ) supérieur à PicOrderCnt(CurrPic) et inférieur ou égal à PicOrderCnt(CurrPic) + deblocking_display_preference_repetition_period; à moins que le flux binaire ne se termine ou qu'une nouvelle séquence vidéo codée ne commence sans sortie d'une telle image.

D.2.22 Sémantique de message SEI d'informations stéréoscopiques

Ce message SEI fournit au décodeur une indication du fait que l'ensemble de la séquence vidéo codée se compose de paires d'images formant un contenu stéréoscopique.

Le message SEI d'informations stéréoscopiques ne doit pas être présent dans une quelconque unité d'accès d'une séquence vidéo codée à moins qu'un message SEI d'informations stéréoscopiques ne soit présent dans la première unité d'accès de la séquence vidéo codée.

sub-frame_views_flag égal à 1 indique que toutes les images contenues dans la présente séquence vidéo codée sont des sous-trames et que toutes les sous-trames d'une parité particulière sont considérées comme une vue gauche et que toutes les sous-trames de la parité opposée sont considérées comme une vue droite pour un contenu stéréoscopique. Un fanion sub-frame_views_flag égal à 0 indique que toutes les images contenues dans la présente séquence vidéo codée sont des trames, et que des trames alternantes dans l'ordre de sortie représentent un aspect d'une vue stéréoscopique. La valeur du fanion sub-frame_views_flag doit être la même dans tous les messages SEI d'informations stéréoscopiques contenus dans une séquence vidéo codée.

Lorsque le message SEI d'informations stéréoscopiques est présent et que le fanion sub-frame_views_flag est égal à 1, la vue gauche et la vue droite d'une paire stéréoscopique doivent être codées comme une paire de sous-trames complémentaires, l'instant d'affichage de la première sous-trame de la paire de sous-trames dans l'ordre de sortie devrait être retardé de façon à coïncider avec l'instant d'affichage de la seconde sous-trame de la paire de sous-trames dans l'ordre de sortie, et les localisations spatiales des échantillons dans chaque sous-trame individuelle devraient être interprétées, aux fins de l'affichage, comme représentant des images complètes comme représenté dans la Figure 6-1 plutôt que des sous-trames spatialement distinctes, contenues dans une trame, comme représenté dans la Figure 6-2.

NOTE – Le processus d'affichage n'est pas spécifié dans la présente Recommandation | Norme internationale.

top_sub-frame_is_left_view_flag égal à 1 indique que les sous-trames supérieures dans la séquence vidéo codée représentent une vue gauche et que les sous-trames inférieures dans la séquence vidéo codée représentent une vue droite. Un fanion top_sub-frame_is_left_view_flag égal à 0 indique que les sous-trames inférieures dans la séquence vidéo codée représentent une vue gauche et que les sous-trames supérieures dans la séquence vidéo codée représentent une vue droite. Lorsqu'elle est présente, la valeur de top_sub-frame_is_left_view_flag doit être la même dans tous les messages SEI d'informations stéréoscopiques contenus dans une séquence vidéo codée.

current_frame_is_left_view_flag égal à 1 indique que l'image actuelle est la vue gauche d'une paire de vues stéréoscopiques. Un fanion current_frame_is_left_view_flag égal à 0 indique que l'image actuelle est la vue droite d'une paire de vues stéréoscopiques.

next_frame_is_second_view_flag égal à 1 indique que l'image actuelle et la prochaine image dans l'ordre de sortie forment une paire de vues stéréoscopiques, et l'instant d'affichage de l'image actuelle devrait être retardé de façon à coïncider avec l'instant d'affichage de l'image suivante dans l'ordre de sortie. Un fanion

next_frame_is_second_view_flag égal à 0 indique que l'image actuelle et la précédente image dans l'ordre de sortie forment une paire de vues stéréoscopiques, et l'instant d'affichage de l'image actuelle ne devrait pas être retardé aux fins de l'appariement de vues stéréoscopiques.

left_view_self_contained_flag égal à 1 indique qu'aucune opération d'interprédiction, contenue dans le processus de décodage pour les images de vue gauche de la séquence vidéo codée, ne se rapporte à des images de référence qui sont des images de vue droite. Un fanion left_view_self_contained_flag égal à 0 indique que certaines opérations d'interprédiction contenues dans le processus de décodage pour les images de vue gauche de la séquence vidéo codée peuvent, le cas échéant, ne se rapporter qu'à des images de référence qui sont des images de vue droite. A l'intérieur d'une séquence vidéo codée, la valeur de left_view_self_contained_flag doit être la même dans tous les messages SEI d'informations stéréoscopiques.

right_view_self_contained_flag égal à 1 indique qu'aucune opération d'interprédiction contenue dans le processus de décodage pour les images de vue droite de la séquence vidéo codée ne se rapporte à des images de référence qui sont des images de vue gauche. Un fanion right_view_self_contained_flag égal à 0 indique que certaines opérations d'interprédiction contenues dans le processus de décodage pour les images de vue droite de la séquence vidéo codée peuvent, le cas échéant, ne se rapporter qu'à des images de référence qui sont des images de vue gauche. A l'intérieur d'une séquence vidéo codée, la valeur de right_view_self_contained_flag doit être la même dans tous les messages SEI d'informations stéréoscopiques.

D.2.23 Sémantique de message SEI réservé

Ce message consiste en données réservées pour une utilisation future à compatibilité ascendante par l'UIT-T | ISO/CEI. Les codeurs conformes à la présente Recommandation | Norme internationale ne doivent pas envoyer de messages SEI réservés tant que l'utilisation de tels messages n'aura pas été spécifiée par l'UIT-T | ISO/CEI. Les décodeurs conformes à la présente Recommandation | Norme internationale qui rencontrent des messages SEI réservés doivent ignorer leur contenu sans effet sur le processus de décodage, sauf spécification contraire dans de futures Recommandations | Normes internationales spécsiiées par l'UIT-T | ISO/CEI.

reserved_sei_message_payload_byte est un octet réservé pour utilisation future par l'UIT-T | ISO/CEI.

Annexe E

Informations de facilité d'utilisation vidéo (VUI)

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie la syntaxe et la sémantique des paramètres VUI de l'ensemble de paramètres de séquences.

Les paramètres VUI ne sont pas nécessaires pour la construction des échantillons luma ou chroma par le processus de décodage. Les décodeurs conformes ne sont pas tenus de traiter ces informations afin d'être conformes à l'ordre de sortie de la présente Recommandation | Norme internationale (voir l'Annexe C pour la spécification de la conformité). Certains paramètres VUI sont nécessaires afin de vérifier la conformité du flux binaire et pour la conformité du délai de sortie du décodeur.

A l'Annexe E, la spécification de présence des paramètres VUI est également satisfaite lorsque ces paramètres (ou un de leurs sous-ensembles) sont convoyés aux décodeurs (ou au décodeur HRD) par d'autres moyens non spécifiés par la présente Recommandation | Norme internationale. Lorsqu'ils sont présents dans le flux binaire, les paramètres VUI doivent suivre la syntaxe et la sémantique spécifiée aux § 7.3.2.1 et 7.4.2.1 et dans la présente annexe. Lorsque le contenu des paramètres VUI est convoyé pour l'application par des moyens autres que la présence au sein du flux binaire, la représentation du contenu des paramètres VUI ne doit pas nécessairement utiliser la syntaxe spécifiée dans la présente annexe. Pour les besoins du comptage des bits, seuls sont comptés les bits appropriés qui sont réellement présents dans le flux binaire.

E.1 Syntaxe VUI

E.1.1 Syntaxe des paramètres d'informations VUI

vui_parameters( ) {	C	Descripteur
aspect_ratio_info_present_flag	0	u(1)
si(aspect_ratio_info_present_flag) {		
aspect_ratio_idc	0	u(8)
si(aspect_ratio_idc == Extended_SAR) {		
sar_width	0	u(16)
sar_height	0	u(16)
}		
}		
overscan_info_present_flag	0	u(1)
si(overscan_info_present_flag)		
overscan_appropriate_flag	0	u(1)
video_signal_type_present_flag	0	u(1)
si(video_signal_type_present_flag) {		
video_format	0	u(3)
video_full_range_flag	0	u(1)
colour_description_present_flag	0	u(1)
si(colour_description_present_flag) {		
colour_primaries	0	u(8)
transfer_characteristics	0	u(8)
matrix_coefficients	0	u(8)
}		
}		
chroma_loc_info_present_flag	0	u(1)
si(chroma_loc_info_present_flag) {		
chroma_sample_loc_type_top_field	0	ue(v)
chroma_sample_loc_type_bottom_field	0	ue(v)

}		
timing_info_present_flag	0	u(1)
si(timing_info_present_flag) {		
num_units_in_tick	0	u(32)
time_scale	0	u(32)
fixed_frame_rate_flag	0	u(1)
}		
nal_hrd_parameters_present_flag	0	u(1)
si(nal_hrd_parameters_present_flag)		
hrd_parameters( )		
vcl_hrd_parameters_present_flag	0	u(1)
si(vcl_hrd_parameters_present_flag)		
hrd_parameters( )		
si(nal_hrd_parameters_present_flag || vcl_hrd_parameters_present_flag)		
low_delay_hrd_flag	0	u(1)
pic_struct_present_flag	0	u(1)
bitstream_restriction_flag	0	u(1)
si(bitstream_restriction_flag) {		
motion_vectors_over_pic_boundaries_flag	0	u(1)
max_bytes_per_pic_denom	0	ue(v)
max_bits_per_mb_denom	0	ue(v)
log2_max_mv_length_horizontal	0	ue(v)
log2_max_mv_length_vertical	0	ue(v)
num_reorder_frames	0	ue(v)
max_dec_frame_buffering	0	ue(v)
}		
}		

E.1.2 Syntaxe des paramètres de décodeur HRD

hrd_parameters( ) {	**C**	**Descripteur**
cpb_cnt_minus1	0	ue(v)
bit_rate_scale	0	u(4)
cpb_size_scale	0	u(4)
pour(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
bit_rate_value_minus1[SchedSelIdx **]**	0	ue(v)
cpb_size_value_minus1[SchedSelIdx **]**	0	ue(v)
cbr_flag[SchedSelIdx **]**	0	u(1)
}		
initial_cpb_removal_delay_length_minus1	0	u(5)
cpb_removal_delay_length_minus1	0	u(5)
dpb_output_delay_length_minus1	0	u(5)
time_offset_length	0	u(5)
}		

E.2 Sémantique VUI

E.2.1 Sémantique des paramètres d'informations VUI

aspect_ratio_info_present_flag égal à 1 spécifie que aspect_ratio_idc est présent. Un fanion aspect_ratio_info_present_flag égal à 0 spécifie que aspect_ratio_idc n'est pas présent.

aspect_ratio_idc spécifie la valeur du rapport d'aspect d'échantillon des échantillons luma. Le Tableau E-1 indique la signification du code. Lorsque aspect_ratio_idc indique Extended_SAR, le rapport d'aspect d'échantillon est représenté par sar_width et par sar_height. Lorsque l'élément syntaxique aspect_ratio_idc n'est pas présent, la valeur de aspect_ratio_idc doit être supposée égale à 0.

Tableau E-1 – Signification de l'indicateur de rapport d'aspect d'échantillon

aspect_ratio_idc	Rapport d'aspect d'échantillon	(pour information) Exemples d'utilisation
0	Non spécifié	
1	1:1 ("carré")	Trame 1280x720 16:9 sans surbalayage horizontal Trame 1920x1080 16:9 sans surbalayage horizontal (recadrée à partir du 1920x1088) Trame 640x480 4:3 sans surbalayage horizontal
2	12:11	Trame 720x576 4:3 avec surbalayage horizontal Trame 352x288 4:3 sans surbalayage horizontal
3	10:11	Trame 720x480 4:3 avec surbalayage horizontal Trame 352x240 4:3 sans surbalayage horizontal
4	16:11	Trame 720x576 16:9 avec surbalayage horizontal Trame 528x576 4:3 sans surbalayage horizontal
5	40:33	Trame 720x480 16:9 avec surbalayage horizontal Trame 528x480 4:3 sans surbalayage horizontal
6	24:11	Trame 352x576 4:3 sans surbalayage horizontal Trame 480x576 16:9 avec surbalayage horizontal
7	20:11	Trame 352x480 4:3 sans surbalayage horizontal Trame 480x480 16:9 avec surbalayage horizontal
8	32:11	Trame 352x576 16:9 sans surbalayage horizontal
9	80:33	Trame 352x480 16:9 sans surbalayage horizontal
10	18:11	Trame 480x576 4:3 avec surbalayage horizontal
11	15:11	Trame 480x480 4:3 avec surbalayage horizontal
12	64:33	Trame 528x576 16:9 sans surbalayage horizontal
13	160:99	Trame 528x480 16:9 sans surbalayage horizontal
14	4:3	Trame 1440x1080 sans surbalayage horizontal
15	3:2	Trame 1280x1080 16:9 sans surbalayage horizontal
16	2:1	Trame 960x1080 16:9 sans surbalayage horizontal
17..254	Réservé	
255	Extended_SAR	

sar_width indique la taille horizontale du rapport d'aspect de l'échantillon (en unités arbitraires).

sar_height indique la taille verticale du rapport d'aspect de l'échantillon (dans les mêmes unités arbitraires que sar_width).

sar_width et sar_height doivent être premiers entre eux ou égaux à 0. Lorsque aspect_ratio_idc est égal à 0 ou que sar_width est égal à 0 ou que sar_height est égal à 0, le rapport d'aspect d'échantillon doit être considéré comme non spécifié par la présente Recommandation | Norme internationale.

overscan_info_present_flag égal à 1 spécifie que le fanion overscan_appropriate_flag est présent. Lorsque overscan_info_present_flag est égal à 0 ou n'est pas présent, la méthode d'affichage préférée pour le signal vidéo n'est pas spécifiée.

overscan_appropriate_flag égal à 1 indique que les sorties d'images décodées recadrées conviennent pour l'affichage au moyen du surbalayage. overscan_appropriate_flag égal à 0 indique que les sorties d'images décodées recadrées contiennent des informations visuellement importantes dans toute la région jusqu'aux bordures de la mire de l'image, et que les sorties d'images codées recadrées ne devraient pas être affichées au moyen du surbalayage. Au lieu de cela, elles

devraient être affichées soit au moyen d'une correspondance exacte entre la zone d'affichage et la mire, soit au moyen du sous-balayage.

NOTE 1 – Par exemple, overscan_appropriate_flag égal à 1 pourrait être utilisé pour de la programmation de télévision de loisirs, ou pour la vue de gens en direct dans une visioconférence, et overscan_appropriate_flag égal à 0 pourrait être utilisé afin de capturer un écran d'ordinateur ou pour le contenu d'une caméra de sécurité.

video_signal_type_present_flag égal à 1 spécifie que video_format, video_full_range_flag et colour_description_present_flag sont présents. video_signal_type_present_flag égal à 0, spécifie que video_format, video_full_range_flag et colour_description_present_flag ne sont pas présents.

video_format indique la représentation des images comme spécifié au Tableau E-2, avant qu'elles ne soient codées en conformité avec la présente Recommandation | Norme internationale. Lorsque l'élément syntaxique video_format n'est pas présent, la valeur de video_format doit être supposée égale à 5.

Tableau E-2 – Signification de video_format

video_format	Signification
0	Composante
1	PAL
2	NTSC
3	SECAM
4	MAC
5	Format vidéo non spécifié
6	Réservé
7	Réservé

video_full_range_flag indique le niveau de noir et la gamme des signaux luma et chroma telle que calculée à partir des signaux de composantes analogiques E'_Y, E'_{PB}, et E'_{PR} ou E'_R, E'_G, et E'_B.

Lorsque l'élément syntaxique video_full_range_flag n'est pas présent, sa valeur doit être supposée égale à 0.

colour_description_present_flag égal à 1 spécifie que colour_primaries, transfer_characteristics et matrix_coefficients sont présents. Un fanion colour_description_present_flag égal à 0 spécifie que colour_primaries, transfer_characteristics et matrix_coefficients ne sont pas présents.

colour_primaries indique les coordonnées chromatiques des primaires de la source, comme spécifié au Tableau E-3, dans les termes de la définition de la norme CEI 1931 de x et y comme spécifié par l'ISO/CEI 10527.

Lorsque l'élément syntaxique colour_primaries n'est pas présent, la valeur de colour_primaries doit être supposée égale à 2 (le chromatisme n'est pas spécifié ou est déterminé par l'application).

Tableau E-3 – Couleurs primaires

Valeur	Primaires			Remarque informative
0	Réservé			Pour utilisation future par l'UIT-T | ISO/CEI
1	primaire	x	y	Recommandation UIT-R BT.709-5
	vert	0,300	0,600	
	bleu	0,150	0,060	
	rouge	0,640	0,330	
	blanc D65	0,3127	0,3290	
2	Non spécifié			Les caractéristiques d'image sont inconnues ou sont déterminées par l'application.
3	Réservé			
4	primaire	x	y	Recommandation UIT-R BT.470-6 Système M
	vert	0,21	0,71	
	bleu	0,14	0,08	
	rouge	0,67	0,33	
	blanc C	0,310	0,316	
5	primaire	x	y	Recommandation UIT-R BT.470-6 Système B, G
	vert	0,29	0,60	
	bleu	0,15	0,06	
	rouge	0,64	0,33	
	blanc D65	0,3127	0,3290	
6	primaire	x	y	Société des ingénieurs en images animées et télévision 170M (1999)
	vert	0,310	0,595	
	bleu	0,155	0,070	
	rouge	0,630	0,340	
	blanc D65	0,3127	0,3290	
7	primaire	x	y	Société des ingénieurs en images animées et télévision 240M (1999)
	vert	0,310	0,595	
	bleu	0,155	0,070	
	rouge	0,630	0,340	
	blanc D65	0,3127	0,3290	
8	primaire	x	y	Film générique (filtres colorés au moyen de l'illuminant C)
	vert	0,243	0,692 (Wratten 58)	
	bleu	0,145	0,049 (Wratten 47)	
	rouge	0,681	0,319 (Wratten 25)	
	blanc C	0,310	0,316	
9-255	Réservé			Pour utilisation future par l'UIT-T | ISO/CEI

transfer_characteristics indique les caractéristiques de transfert optoélectroniques de l'image source, comme spécifié au Tableau E-4, comme une fonction d'une entrée d'intensité optique linéaire L_c avec une gamme analogique de 0 à 1.

Lorsque l'élément syntaxique transfer_characteristics n'est pas présent, la valeur de transfer_characteristics doit être supposée égale à 2 (les caractéristiques de transfert ne sont pas spécifiées ou sont déterminées par l'application).

Tableau E-4 – Caractéristiques de transfert

Valeur	Caractéristiques de transfert	Remarque informative
0	Réservé	Pour utilisation future par l'UIT-T | ISO/CEI
1	$V = 1{,}099\ L_c^{0{,}45} - 0{,}099$ pour $1 >= L_c >= 0{,}018$ $V = 4{,}500\ L_c$ pour $0{,}018 > L_c >= 0$	Recommandation UIT-R BT.709-5
2	Non spécifié	Les caractéristiques d'image sont inconnues ou sont déterminées par l'application.
3	Réservé	Pour utilisation future par l'UIT-T | ISO/CEI
4	Gamma d'affichage présumé: 2.2	Recommandation UIT-R BT.470-6 Système M
5	Gamma d'affichage présumé: 2.8	Recommandation UIT-R BT.470-6 Système B, G
6	$V = 1{,}099\ L_c^{0{,}45} - 0{,}099$ pour $1 >= L_c >= 0{,}018$ $V = 4{,}500\ L_c$ pour $0{,}018 > L_c >= 0$	Société des ingénieurs en images animées et télévision 170M (1999)
7	$V = 1{,}1115\ L_c^{0{,}45} - 0{,}1115$ pour $1 >= L_c >= 0{,}0228$ $V = 4{,}0\ L_c$ pour $0{,}0228 > L_c >= 0$	Société des ingénieurs en images animées et télévision 240M (1999)
8	$V = L_c$ pour $1 > L_c >= 0$	Caractéristiques de transfert linéaires
9	$V = 1{,}0 - \text{Log10}(L_c) \div 2$ pour $1 >= L_c >= 0{,}01$ $V = 0{,}0$ pour $0{,}01 > L_c >= 0$	Caractéristiques de transfert logarithmiques (rapport: 100:1)
10	$V = 1{,}0 - \text{Log10}(L_c) \div 2.5$ pour $1 >= L_c >= 0{,}0031622777$ $V = 0{,}0$ pour $0{,}0031622777 > L_c >= 0$	Caractéristiques de transfert logarithmiques (rapport: 316,22777:1)
11..255	Réservé	Pour utilisation future par l'UIT-T | ISO/CEI

matrix_coefficients décrit les coefficients matriciels utilisés afin de déduire les signaux luma et chroma d'après les primaires vertes, bleues et rouges, comme spécifié au Tableau E-5.

L'élément matrix_coefficients ne doit pas être égal à 0 à moins que les deux conditions suivantes ne soient vraies:

– BitDepth_C est égal à BitDepth_Y

– chroma_format_idc est égal à 3 (4:4:4)

La spécification visant à utiliser l'élément matrix_coefficients égal à 0 dans toutes les autres conditions est réservé pour utilisation future par l'UIT-T | ISO/CEI.

matrix_coefficients ne doit pas être égal à 8 à moins qu'une des (ou les) deux conditions suivantes ne soit (soient) vraie(s):

– BitDepth_C est égal à BitDepth_Y

– BitDepth_C est égal à $\text{BitDepth}_Y + 1$ et chroma_format_idc est égal à 3 (4:4:4)

La spécification visant à utiliser l'élément matrix_coefficients égal à 8 dans toutes les autres conditions est réservé pour utilisation future par l'UIT-T | ISO/CEI.

Lorsque l'élément syntaxique matrix_coefficients n'est pas présent, sa valeur doit être supposée égale à 2.

L'interprétation de l'élément matrix_coefficients est définie comme suit:

– E'_R, E'_G, et E'_B sont analogiques avec des valeurs dans l'étendue de 0 à 1.

– Le blanc est spécifié comme ayant E'_R égal à 1, E'_G égal à 1, et E'_B égal à 1.

– Le noir est spécifié comme ayant E'_R égal à 0, E'_G égal à 0, et E'_B égal à 0,

– Si le fanion video_full_range_flag est égal à 0, les équations suivantes s'appliquent.

  – Si matrix_coefficients est égal à 1, 4, 5, 6, ou 7, les équations suivantes s'appliquent.

$$Y = \text{Round}((1 << (\text{BitDepth}_Y - 8)) * (219 * E'_Y + 16)) \quad \text{(E-1)}$$

$$Cb = \text{Round}((1 << (\text{BitDepth}_C - 8)) * (224 * E'_{PB} + 128)) \quad \text{(E-2)}$$

$$Cr = Round((1 << (BitDepth_C - 8)) * (224 * E'_{PR} + 128)) \quad (E\text{-}3)$$

– Sinon, si matrix_coefficients est égal à 0 ou 8, les équations suivantes s'appliquent.

$$R = (1 << (BitDepth_Y - 8)) * (219 * E'_R + 16) \quad (E\text{-}4)$$

$$G = (1 << (BitDepth_Y - 8)) * (219 * E'_G + 16) \quad (E\text{-}5)$$

$$B = (1 << (BitDepth_Y - 8)) * (219 * E'_B + 16) \quad (E\text{-}6)$$

– Sinon, si matrix_coefficients est égal à 2, l'interprétation de l'élément syntaxique matrix_coefficients est inconnue ou est déterminée par l'application.

– Sinon (si matrix_coefficients n'est pas égal à 0, 1, 2, 4, 5, 6, 7, ou 8), l'interprétation de l'élément syntaxique matrix_coefficients est réservée pour future définition par l'UIT-T | ISO/CEI.

– Sinon (si video_full_range_flag est égal à 1), les équations suivantes s'appliquent.

– Si matrix_coefficients est égal à 1, 4, 5, 6, ou 7, les équations suivantes s'appliquent.

$$Y = Round(((1 << BitDepth_Y) - 1) * E'_Y) \quad (E\text{-}7)$$

$$Cb = Round(((1 << BitDepth_C) - 1) * E'_{PB} + (1 << (BitDepth_C - 1)) \quad (E\text{-}8)$$

$$Cr = Round(((1 << BitDepth_C) - 1) * E'_{PR} + (1 << (BitDepth_C - 1)) \quad (E\text{-}9)$$

– Sinon, si matrix_coefficients est égal à 0 ou 8, les équations suivantes s'appliquent.

$$R = ((1 << BitDepth_Y) - 1) * E'_R \quad (E\text{-}10)$$

$$G = ((1 << BitDepth_Y) - 1) * E'_G \quad (E\text{-}11)$$

$$B = ((1 << BitDepth_Y) - 1) * E'_B \quad (E\text{-}12)$$

– Sinon, si matrix_coefficients est égal à 2, l'interprétation de l'élément syntaxique matrix_coefficients est inconnue ou est déterminée par l'application.

– Sinon (si matrix_coefficients n'est pas égal à 0, 1, 2, 4, 5, 6, 7, ou 8), l'interprétation de l'élément syntaxique matrix_coefficients est réservée pour future définition par l'UIT-T | ISO/CEI.

– Si matrix_coefficients n'est pas égal à 0 ou 8, les équations suivantes s'appliquent.

$$E'_Y = K_R * E'_R + (1 - K_R - K_B) * E'_G + K_B * E'_B \quad (E\text{-}13)$$

$$E'_{PB} = 0.5 * (E'_B - E'_Y) \div (1 - K_B) \quad (E\text{-}14)$$

$$E'_{PR} = 0.5 * (E'_R - E'_Y) \div (1 - K_R) \quad (E\text{-}15)$$

NOTE 2 – Alors E'Y est analogique avec des valeurs dans l'étendue de 0 à 1, E'_{PB} et E'_{PR} sont analogiques avec des valeurs dans l'étendue de -0,5 à 0,5, et le blanc est donné en équivalence par $E'_Y = 1$, $E'_{PB} = 0$, $E'_{PR} = 0$,

– Sinon, si matrix_coefficients est égal à 0, les équations suivantes s'appliquenNt.

$$Y = Round(G) \quad (E\text{-}16)$$

$$Cb = Round(B) \quad (E\text{-}17)$$

$$Cr = Round(R) \quad \text{(E-18)}$$

– Sinon (si matrix_coefficients est égal à 8), ce qui suit est applicable:

  – Si $BitDepth_C$ est égal à $BitDepth_Y$, les équations suivantes s'appliquent.

$$Y = Round(0.5 * G + 0.25 * (R + B)) \quad \text{(E-19)}$$

$$Cb = Round(0.5 * G – 0.25 (R + B)) \quad \text{(E-20)}$$

$$Cr = Round(0.5 * (R – B)) \quad \text{(E-21)}$$

NOTE 3 – Aux fins de la nomenclature YCgCo qui est utilisée dans le Tableau E-5, les variables Cb et Cr des équations E-20 et E-21 peuvent être respectivement rapportées à Cg et Co. La conversion en inverse pour les quatre équations ci-dessus devrait être calculée comme suit.

$$t = Y – Cb \quad \text{(E-22)}$$

$$G = Y + Cb \quad \text{(E-23)}$$

$$B = t – Cr \quad \text{(E-24)}$$

$$R = t + Cr \quad \text{(E-25)}$$

  – Sinon (si $BitDepth_C$ n'est pas égal à $BitDepth_Y$), les équations suivantes s'appliquent.

$$Cr = Round(R) – Round(B) \quad \text{(E-26)}$$

$$t = Round(B) + (Cr >> 1) \quad \text{(E-27)}$$

$$Cb = Round(G) – t \quad \text{(E-28)}$$

$$Y = t + (Cb >> 1) \quad \text{(E-29)}$$

NOTE 4 – Aux fins de la nomenclature YCgCo qui est utilisée dans le Tableau E-5, les variables Cb et Cr des équations E-28 et E-26 peuvent être respectivement rapportées à Cg et Co. Le conversion en inverse pour les quatre équations ci-dessus devrait être calculée comme suit:.

$$t = Y – (Cb >> 1) \quad \text{(E-30)}$$

$$G = t + Cb \quad \text{(E-31)}$$

$$B = t – (Cr >> 1) \quad \text{(E-32)}$$

$$R = B + Cr \quad \text{(E-33)}$$

Tableau E-5 – Coefficients matriciels

Valeur	Matrice	Remarque informative
0	GBR	Couramment désigné par RGB; voir équations E-16 à E-18
1	$K_R = 0{,}2126$; $K_B = 0{,}0722$	Recommandation UIT-R BT.709-5 et Société des ingénieurs en images animées et télévision RP177 (1993)
2	Non spécifié	Les caractéristiques d'image sont inconnues ou sont déterminées par l'application.
3	Réservé	Pour utilisation future par l'UIT-T | ISO/CEI
4	$K_R = 0{,}30$; $K_B = 0{,}11$	Commission fédérale des communications (Etats-Unis d'Amérique) – Titre 47 – Code de réglementation fédérale (2003) 73.682 (a) (20)
5	$K_R = 0{,}299$; $K_B = 0{,}114$	Recommandation UIT-R BT.470-6 – Système B, G
6	$K_R = 0{,}299$; $K_B = 0{,}114$	Société des ingénieurs en images animées et télévision 170M (1999)
7	$K_R = 0{,}212$; $K_B = 0{,}087$	Société des ingénieurs en images animées et télévision 240M (1999)
8	YCgCo	Voir équations E-19 à E-33
9-255	Réservé	Pour utilisation future par l'UIT-T | ISO/CEI

chroma_loc_info_present_flag égal à 1 spécifie que chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field sont présents. Un fanion chroma_loc_info_present_flag égal à 0 spécifie que chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field ne sont pas présents.

chroma_sample_loc_type_top_field et **chroma_sample_loc_type_bottom_field** spécifient la localisation des échantillons chroma pour la sous-trame supérieure et pour la sous-trame inférieure comme indiqué à la Figure E-1. La valeur de chroma_sample_loc_type_top_field et de chroma_sample_loc_type_bottom_field doit être dans la gamme de 0 à 5 inclus. Lorsque chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field ne sont pas présents, les valeurs de chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field doivent être supposées égales à 0.

NOTE 5 – Lors du codage de matériau source progressif, chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field devraient avoir la même valeur.

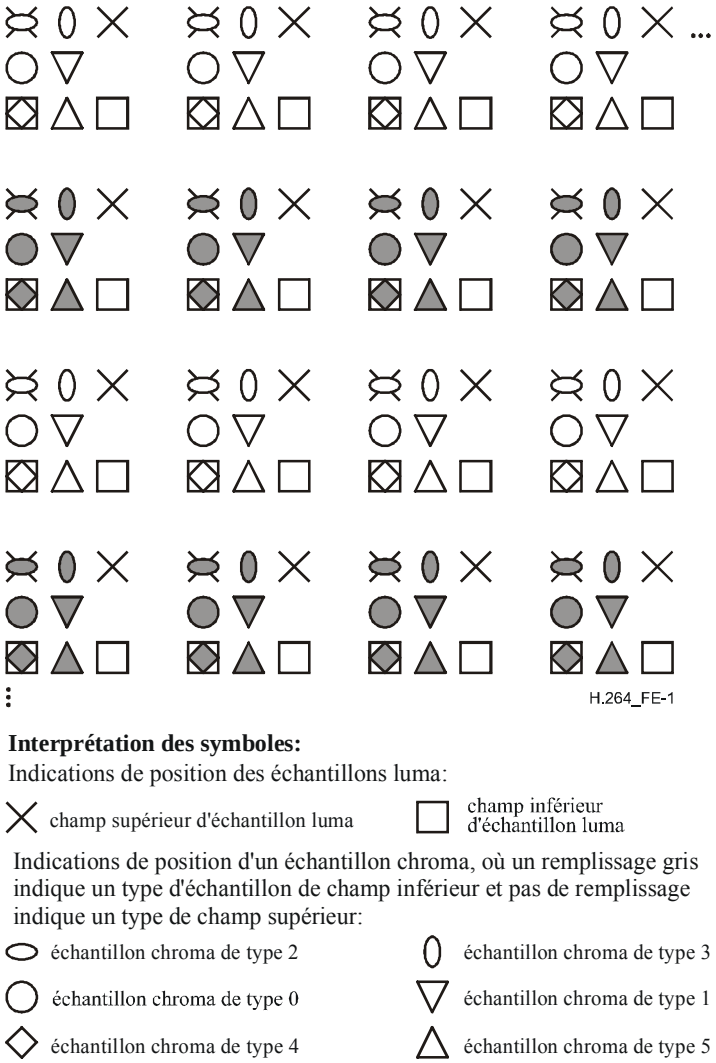


Figure E-1 – Localisation des échantillons chroma pour les sous-trames supérieures et inférieures en fonction de chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field

timing_info_present_flag égal à 1 spécifie que num_units_in_tick, time_scale et fixed_frame_rate_flag sont présents dans le flux binaire. Un fanion timing_info_present_flag égal à 0 spécifie que num_units_in_tick, time_scale et fixed_frame_rate_flag ne sont pas présents dans le flux binaire.

num_units_in_tick est le nombre d'unités de temps d'une horloge fonctionnant à la fréquence time_scale Hz qui correspond à un incrément (appelé *tic-tac d'horloge* ou *raie*) d'un compteur de tic-tac d'horloge. num_units_in_tick doit être supérieur à 0. Un tic-tac d'horloge est l'intervalle de temps minimal qui peut être représenté dans les données codées. Par exemple, lorsque la fréquence d'horloge d'un signal vidéo est 60 000 ÷ 1001 Hz, time_scale peut être 60 000 et num_units_in_tick peut être égal à 1001. Voir l'équation C-1.

time_scale est le nombre d'unités de temps qui passent en une seconde. Par exemple, un système de coordonnées temporelles qui mesure le temps en utilisant une horloge à 27 MHz a un nombre time_scale de 27 000 000. Le nombre time_scale doit être supérieur à 0.

fixed_frame_rate_flag égal à 1 indique que la distance en temps entre les temps de sortie du décodeur HRD de deux images quelconques consécutives dans l'ordre de sortie est soumise aux contraintes suivantes. Un fanion fixed_frame_rate_flag égal à 0 indique qu'aucune contrainte semblable ne s'applique à la distance en temps entre les moments de sortie du décodeur HRD de deux images quelconques consécutives dans l'ordre de sortie.

Pour chaque image n où n indique la $n^{ième}$ image sortie dans l'ordre de sortie et où l'image n n'est pas la dernière image dans le flux binaire sortie dans l'ordre de sortie, la valeur de $\Delta t_{fi,dpb}(n)$ est spécifiée par:

$$\Delta t_{fi,dpb}(n) = \Delta t_{o,dpb}(n) \div \text{DeltaTfiDivisor} \quad \text{(E-34)}$$

où $\Delta t_{o,dpb}(n)$ est spécifié à l'équation C-13 et DeltaTfiDivisor est spécifié par le Tableau E-6 sur la base de la valeur de pic_struct_present_flag, field_pic_flag, et pic_struct pour la séquence vidéo codée contenant l'image n. Les entrées marquées "–" dans le Tableau E-6 indiquent une absence de dépendance de DeltaTfiDivisor à l'élément syntaxique correspondant.

Lorsque le fanion fixed_frame_rate_flag est égal à 1 pour une séquence vidéo codée contenant l'image n, la valeur calculée pour $\Delta t_{fi,dpb}(n)$ doit être égale à t_c comme spécifié dans l'équation C-1 (au moyen de la valeur de t_c pour la séquence vidéo codée contenant l'image n) lorsqu'une des deux (ou les deux) condition(s) suivante(s) est (sont) vraie(s) pour l'image suivante n_n qui est spécifiée pour utilisation dans l'équation C-13.

– l'image n_n est dans la même séquence vidéo codée que l'image n.

– l'image n_n est dans une autre séquence vidéo codée et le fanion fixed_frame_rate_flag est égal à 1 dans la séquence vidéo codée contenant l'image n_n et la valeur de num_units_in_tick ÷ time_scale est la même pour les deux séquences vidéo codées.

Tableau E-6 – Diviseur pour le calcul de $\Delta t_{fi,dpb}(n)$

pic_struct_present_flag	field_pic_flag	pic_struct	DeltaTfiDivisor
0	1	-	1
1	-	1	1
1	-	2	1
0	0	-	2
1	-	0	2
1	-	3	2
1	-	4	2
1	-	5	3
1	-	6	3
1	-	7	4
1	-	8	6

nal_hrd_parameters_present_flag égal à 1 spécifie que les paramètres NAL HRD (relevant de la conformité de flux binaire de Type II) sont présents. nal_hrd_parameters_present_flag égal à 0 spécifie que les paramètres NAL HRD ne sont pas présents.

NOTE 6 – Lorsque le fanion nal_hrd_parameters_present_flag est égal à 0, on ne peut pas vérifier la conformité du flux binaire sans fournir les paramètres NAL HRD, y compris les informations paramètriques du décodeur HRD de séquence NAL et tous les messages SEI de période de mise en mémoire tampon et de synchronisation d'image, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

Lorsque le fanion nal_hrd_parameters_present_flag est égal à 1, les paramètres NAL HRD (§ E.1.2 et E.2.2) suivent immédiatement le fanion.

La variable NalHrdBpPresentFlag est déduite comme suit.

– Si une des conditions suivantes est vraie, la valeur de NalHrdBpPresentFlag doit être mise égale à 1.

  – nal_hrd_parameters_present_flag est présent dans le flux binaire et est égal à 1;

  – la nécessité de la présence de périodes de mise en mémoire tampon pour le fonctionnement NAL HRD dans le flux binaire dans les messages SEI de période de mise en mémoire tampon est déterminée par l'application, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

– Sinon, la valeur de NalHrdBpPresentFlag doit être mise égale à 0.

vcl_hrd_parameters_present_flag égal à 1 spécifie que les paramètres VCL HRD (relevant de la conformité de tout flux binaire) sont présents. Un fanion vcl_hrd_parameters_present_flag égal à 0 spécifie que les paramètres VCL HRD ne sont pas présents.

NOTE 7 – Lorsque le fanion vcl_hrd_parameters_present_flag est égal à 0, la conformité du flux binaire ne peut pas être vérifiée sans la fourniture des paramètres de couche VCL du décodeur HRD et de tous les messages SEI de période de mise en mémoire tampon et de synchronisation d'image, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

Lorsque le fanion vcl_hrd_paramètres_present_flag est égal à 1, les paramètres de couche VCL du décodeur HRD (§ E.1.2 et E.2.2) suivent immédiatement le fanion.

La variable VclHrdBpPresentFlag est déduite comme suit.

– Si une des conditions suivantes est vraie, la valeur de VclHrdBpPresentFlag doit être mise égale à 1.

  – vcl_hrd_paramèters_present_flag est présent dans le flux binaire et est égal à 1;

  – la nécessité de la présence de périodes de mise en mémoire tampon pour le fonctionnement VCL HRD dans le flux binaire dans les messages SEI de période de mise en mémoire tampon est déterminée par l'application, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

– Sinon, la valeur de VclHrdBpPresentFlag doit être mise égale à 0.

La variable CpbDpbDelaysPresentFlag est déduite comme suit.

– Si une des conditions suivantes est vraie, la valeur de CpbDpbDelaysPresentFlag doit être mise égale à 1.

  – nal_hrd_parameters_present_flag est présent dans le flux binaire et est égal à 1;

  – vcl_hrd_parameters_present_flag est présent dans le flux binaire et est égal à 1;

  – la nécessité de la présence de délais de sortie CPB et DPB dans le flux binaire dans les messages SEI de synchronisation d'image est déterminée par l'application, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

– Sinon, la valeur de CpbDpbDelaysPresentFlag doit être mise égale à 0.

low_delay_hrd_flag spécifie le mode de fonctionnement du décodeur HRD comme spécifié à l'Annexe C. Lorsque fixed_frame_rate_flag est égal à 1, low_delay_hrd_flag doit être égal à 0.

NOTE 8 – Lorsque low_delay_hrd_flag est égal à 1, "les grandes images", qui violent le temps nominal de retrait du tampon CPB à cause du nombre de bits utilisés par une unité d'accès, sont permises. On supposera, sans l'exiger, que de telles "grandes images" ne surviennent qu'occasionnellement.

pic_struct_present_flag égal à 1 spécifie que les messages SEI de synchronisation d'image (§ D.2.2) sont présents et qu'ils incluent l'élément syntaxique pic_struct. Un fanion pic_struct_present_flag égal à 0 spécifie que l'élément syntaxique pic_struct n'est pas présent dans les messages SEI de synchronisation d'image. Lorsque pic_struct_present_flag n'est pas présent, sa valeur doit être présumée égale à 0.

bitstream_restriction_flag égal à 1, spécifie que les paramètres de restriction de la séquence vidéo codée suivante du flux binaire sont présents. Un fanion bitstream_restriction_flag égal à 0, spécifie que les paramètres de restriction de la séquence vidéo codée suivante du flux binaire sont absents.

motion_vectors_over_pic_boundaries_flag égal à 0 indique qu'aucun échantillon en dehors des limites de l'image et aucun échantillon à une position fractionnaire d'échantillon dont la valeur est calculée au moyen d'un ou de plusieurs échantillons en dehors des limites de l'image n'est utilisé pour l'interprédiction d'un échantillon. Un fanion motion_vectors_over_pic_boundaries_flag égal à 1 indique qu'un ou plusieurs échantillons en dehors des limites de l'image peuvent être utilisés dans l'interprédiction. Lorsque l'élément syntaxique motion_vectors_over_pic_boundaries_flag n'est pas présent, la valeur de motion_vectors_over_pic_boundaries_flag doit être supposée égale à 1.

max_bytes_per_pic_denom indique un nombre d'octets qui n'est pas dépassé par la somme des tailles des unités NAL de couche VCL associées à toute image codée dans la séquence vidéo codée.

Le nombre d'octets qui représentent une image dans le flux d'unités NAL est spécifié à cette fin comme le nombre total d'octets de données d'unité NAL de couche VCL (c'est-à-dire le total des variables NumBytesInNALunit pour les unités NAL de couche VCL) pour l'image. La valeur de max_bytes_per_pic_denom doit être dans la gamme de 0 à 16 inclus.

En fonction de max_bytes_per_pic_denom on applique ce qui suit.

– Si max_bytes_per_pic_denom est égal à 0, aucune limite n'est indiquée.

– Sinon (si max_bytes_per_pic_denom n'est pas égal à 0), aucune image codée ne doit être représentée dans la séquence vidéo codée par plus que le nombre d'octets suivant:

$$(PicSizeInMbs * RawMbBits) \div (8 * max\_bytes\_per\_pic\_denom) \quad (E\text{-}35)$$

Lorsque l'élément syntaxique max_bytes_per_pic_denom n'est pas présent, la valeur de max_bytes_per_pic_denom doit être supposée égale à 2.

max_bits_per_mb_denom indique le nombre maximal de bits codés des données de macroblock_layer( ) pour tout macrobloc dans toute image de la séquence vidéo codée. La valeur de max_bits_per_mb_denom doit être dans la gamme de 0 à 16 inclus.

En fonction de max_bits_per_mb_denom on applique ce qui suit.

– Si max_bits_per_mb_denom est égal à 0, aucune limite n'est spécifiée.

– Sinon (si max_bits_per_mb_denom n'est pas égal à 0), aucun macroblock_layer) codé ne doit être représenté dans le flux binaire par plus que le nombre de bits suivant:

$$(128 + \text{RawMbBits}) \div \text{max\_bits\_per\_mb\_denom} \quad \text{(E-36)}$$

En fonction de entropy_coding_mode_flag, les bits des données de macroblock_layer( ) sont comptés comme suit.

– Si le fanion entropy_coding_mode_flag est égal à 0, le nombre de bits des données de macroblock_layer( ) est donné par le nombre de bits dans la structure syntaxique macroblock_layer( ) pour un macrobloc.

– Sinon (si entropy_coding_mode_flag est égal à 1), le nombre de bits des données de macroblock_layer( ) pour un macrobloc est donné par le nombre de fois que read_bits(1) est appelé aux § 9.3.3.2.2 et 9.3.3.2.3 lors de l'analyse syntaxique du macroblock_layer( ) associé au macrobloc.

Lorsque l'élément syntaxique max_bits_per_mb_denom n'est pas présent, la valeur de max_bits_per_mb_denom doit être supposée égale à 1.

log2_max_mv_length_horizontal et **log2_max_mv_length_vertical** indiquent la valeur absolue maximale d'une composante, respectivement horizontale et verticale, de vecteur cinétique décodé, en unités de 1/4 d'échantillon luma, pour toutes les images dans la séquence vidéo codée. Une valeur de n atteste qu'aucune valeur de composante ne doit être en dehors de la gamme de $–2^n$ à $2^n –1$ inclus, en unités de déplacement d'un quart d'échantillon luma. La valeur de log2_max_mv_length_horizontal doit être dans la gamme de 0 à 16 inclus. La valeur de log2_max_mv_length_vertical doit être dans la gamme de 0 à 16 inclus. Lorsque log2_max_mv_length_horizontal n'est pas présent, les valeurs de log2_max_mv_length_horizontal et log2_max_mv_length_vertical doivent être supposées égales à 16.

NOTE 9 – La valeur absolue maximal d'une composante verticale ou horizontale de vecteur cinétique décodé comporte aussi des limites de profil et de niveau, comme spécifié à l'Annexe A.

num_reorder_frames indique le nombre maximal de trames, de paires de sous-trames complémentaires, ou de sous-trames non appariées qui précède toute trame, toute paire de sous-trames complémentaires, ou toute sous-trame non appariée dans la séquence vidéo codée dans l'ordre de décodage, et qui la suit dans l'ordre de sortie. La valeur de num_reorder_frames doit être dans la gamme de 0 à max_dec_frame_buffering, inclus. Lorsque l'élément syntaxique num_reorder_frames n'est pas présent, la valeur de num_reorder_frames doit être supposée égale à max_dec_frame_buffering.

max_dec_frame_buffering spécifie la taille requise de la mémoire tampon d'images décodées (DPB) du décodeur HRD en unités de mémoire tampon de trame. La séquence vidéo codée ne doit pas exiger une mémoire tampon d'images décodées d'une taille supérieure à Max(1, max_dec_frame_buffering) afin de permettre la sortie des images décodées aux moments de sortie spécifiés par l'élément dpb_output_delay des messages SEI de synchronisation de l'image. La valeur de max_dec_frame_buffering doit être dans la gamme de num_ref_frames à MaxDpbSize (comme spécifié au § A.3.1 ou au § A.3.2), inclus. Lorsque l'élément syntaxique max_dec_frame_buffering n'est pas présent, la valeur de max_dec_frame_buffering doit être supposée égale à MaxDpbSize.

E.2.2 Sémantique des paramètres de décodeur HRD

cpb_cnt_minus1 plus 1 spécifie le nombre de spécifications CPB à choisir dans le flux binaire. La valeur de cpb_cnt_minus1 doit être dans la gamme de 0 à 31 inclus. Lorsque low_delay_hrd_flag est égal à 1, cpb_cnt_minus1 doit être égal à 0. Lorsque cpb_cnt_minus1 n'est pas présent, il doit être supposé égal à 0.

bit_rate_scale (avec bit_rate_value_minus1[SchedSelIdx]) spécifie le débit binaire d'entrée maximal du $\text{SchedSelIdx}^{\text{ème}}$ CPB.

cpb_size_scale (avec cpb_size_value_minus1[SchedSelIdx]) spécifie la capacité du $\text{SchedSelIdx}^{\text{ème}}$ tampon CPB.

bit_rate_value_minus1[SchedSelIdx **]** (avec bit_rate_scale) spécifie le débit binaire d'entrée maximal pour le $\text{SchedSelIdx}^{\text{ème}}$ CPB. bit_rate_value_minus1[SchedSelIdx] doit être dans la gamme de 0 à $2^{32} – 2$ inclus. Pour tout SchedSelIdx > 0, bit_rate_value_minus1[SchedSelIdx] doit être supérieur à bit_rate_value_minus1[SchedSelIdx – 1]. Le débit binaire en bits par seconde est donné par

$$\text{BitRate}[\text{SchedSelIdx}] = (\text{bit\_rate\_value\_minus1}[\text{SchedSelIdx}] + 1) * 2^{(6 + \text{bit\_rate\_scale})} \quad \text{(E-37)}$$

Lorsque l'élément syntaxique bit_rate_value_minus1[SchedSelIdx] n'est pas présent, la valeur de BitRate[SchedSelIdx] doit être présumée comme suit:

– Si profil_idc est égal à 66, 77, ou 88, BitRate[SchedSelIdx] doit être présumé égal à 1000 * MaxBR pour les paramètres de décodeur HRD en couche VCL et égal à 1200 * MaxBR pour les paramètres de décodeur HRD en couche NAL, où MaxBR est spécifié dans le § A.3.1.

– Sinon, BitRate[SchedSelIdx] doit être présumé égal à cpbBrVclFactor * MaxBR pour les paramètres de décodeur HRD en couche VCL et égal à cpbBrNalFactor * MaxBR pour les paramètres de décodeur HRD en couche NAL, où cpbBrVclFactor, cpbBrNalFactor, et MaxBR sont spécifiés dans le § A.3.3.

cpb_size_value_minus1[SchedSelIdx **]** est utilisé avec cpb_size_scale afin de spécifier la taille du SchedSelIdx^{ème} tampon CPB. cpb_size_value_minus1[SchedSelIdx] doit être dans la gamme de 0 à $2^{32} - 2$ inclus. Pour tout SchedSelIdx supérieur à 0, cpb_size_value_minus1[SchedSelIdx] doit être inférieur ou égal à cpb_size_value_minus1[SchedSelIdx –1].

La taille du tampon CPB en bits est donnée par

$$CpbSize[SchedSelIdx] = (cpb\_size\_value\_minus1[SchedSelIdx] + 1) * 2^{(4 + cpb\_size\_scale)} \quad \text{(E-38)}$$

Lorsque l'élément syntaxique cpb_size_value_minus1[SchedSelIdx] n'est pas présent, la valeur de CpbSize[SchedSelIdx] doit être présumée comme suit:

– Si profil_idc est égal à 66, 77, ou 88, CpbSize[SchedSelIdx] doit être présumé égal à 1000 * MaxCPB pour les paramètres de décodeur HRD en couche VCL et égal à 1200 * MaxCPB pour les paramètres de décodeur HRD en couche NAL, où MaxCPB est spécifié dans le § A.3.1.

– Sinon, CpbSize[SchedSelIdx] doit être présumé égal à cpbBrVclFactor * MaxCPB pour les paramètres de décodeur HRD en couche VCL et égal à cpbBrNalFactor * MaxCPB pour les paramètres de décodeur HRD en couche NAL, où cpbBrVclFactor, cpbBrNalFactor, et MaxCPB sont spécifiés dans le § A.3.3.

cbr_flag[SchedSelIdx **]** égal à 0 spécifie que afin de faire décoder ce flux binaire par le décodeur HRD en utilisant la SchedSelIdx^{ème} spécification CPB, le programmateur factice de livraison de flux (HSS) fonctionne dans un mode de débit binaire intermittent. cbr_flag[SchedSelIdx] égal à 1 spécifie que l'ordonnanceur HSS fonctionne en mode de débit binaire constant (CBR). Lorsque l'élément syntaxique cbr_flag[SchedSelIdx] n'est pas présent, la valeur de cbr_flag doit être supposée égale à 0.

initial_cpb_removal_delay_length_minus1 spécifie la longueur en bits des éléments syntaxiques initial_cpb_removal_delay[SchedSelIdx] et initial_cpb_removal_delay_offset[SchedSelIdx] du message SEI de période de mise en mémoire tampon. La longueur de initial_cpb_removal_delay[SchedSelIdx] et de initial_cpb_removal_delay_offset[SchedSelIdx] est initial_cpb_removal_delay_length_minus1 + 1. Lorsque l'élément syntaxique initial_cpb_removal_delay_length_minus1 est présent dans plus d'une structure syntaxique hrd_parameters( ) au sein de la structure syntaxique des paramètres VUI, la valeur des paramètres initial_cpb_removal_delay_length_minus1 doit être égale dans les deux structures syntaxiques hrd_parameters( ). Lorsque l'élément syntaxique initial_cpb_removal_delay_length_minus1 n'est pas présent, il doit être supposé égal à 23.

cpb_removal_delay_length_minus1 spécifie la longueur en bits de l'élément syntaxique cpb_removal_delay. La longueur de l'élément syntaxique cpb_removal_delay du message SEI de synchronisation d'image est cpb_removal_delay_length_minus1 + 1. Lorsque l'élément syntaxique cpb_removal_delay_length_minus1 est présent dans plus d'une structure syntaxique hrd_parameters( ) au sein de la structure syntaxique de paramètres VUI, la valeur des paramètres cpb_removal_delay_length_minus1 doit être égale dans les deux structures syntaxiques hrd_parameters( ). Lorsque l'élément syntaxique cpb_removal_delay_length_minus1 n'est pas présent, il doit être supposé égal à 23.

dpb_output_delay_length_minus1 spécifie la longueur en bits de l'élément syntaxique dpb_output_delay. La longueur de l'élément syntaxique dpb_output_delay du message SEI de synchronisation d'image est dpb_output_delay_length_minus1 + 1. Lorsque l'élément syntaxique dpb_output_delay_length_minus1 est présent dans plus d'une structure syntaxique hrd_parameters( ) au sein de la structure syntaxique des paramètres VUI, la valeur des paramètres dpb_output_delay_length_minus1 doit être égale dans les deux structures syntaxiques hrd_parameters( ). Lorsque l'élément syntaxique dpb_output_delay_length_minus1 n'est pas présent, il doit être supposé égal à 23.

time_offset_length supérieur à 0 spécifie la longueur en bits de l'élément syntaxique time_offset. time_offset_length égal à 0 spécifie que l'élément syntaxique time_offset n'est pas présent. Lorsque l'élément syntaxique time_offset_length est présent dans plus d'une structure syntaxique hrd_paramètres( ) de la structure syntaxique des paramètres VUI, la valeur des paramètres time_offset_length doit être égale dans les deux structures syntaxiques hrd_parameters( ). Lorsque l'élément syntaxique time_offset_length n'est pas présent, il doit être supposé égal à 24.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	**Systèmes audiovisuels et multimédias**
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	Gestion des télécommunications y compris le RGT et maintenance des réseaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données, communication entre systèmes ouverts et sécurité
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2006