International Telecommunication Union

ITU-R

Radiocommunication Sector of ITU

Recommendation ITU-R BT.1907-0

(01/2012)

Objective perceptual video quality measurement techniques for broadcasting applications using HDTV in the presence of a full reference signal

BT Series

Broadcasting service (television)

International Telecommunication Union

Series of ITU-R Recommendations

(Also available online at http://www.itu.int/publ/R-REC/en)

Series	**Title**
BO	Satellite delivery
BR	Recording for production, archival and play-out; film for television
BS	Broadcasting service (sound)
BT	**Broadcasting service (television)**
F	Fixed service
M	Mobile, radiodetermination, amateur and related satellite services
P	Radiowave propagation
RA	Radio astronomy
RS	Remote sensing systems
S	Fixed-satellite service
SA	Space applications and meteorology
SF	Frequency sharing and coordination between fixed-satellite and fixed service systems
SM	Spectrum management
SNG	Satellite news gathering
TF	Time signals and frequency standards emissions
V	Vocabulary and related subjects

Note*: This ITU-R Recommendation was approved in English under the procedure detailed in Resolution ITU-R 1.*

Electronic Publication
Geneva, 2020

RECOMMENDATION ITU-R BT.1907-0*

Objective perceptual video quality measurement techniques for broadcasting applications using HDTV in the presence of a full reference signal

(2012)

Scope

This Recommendation specifies methods for estimating the perceived video quality of broadcasting applications using HDTV when a full reference signal is available.

The ITU Radiocommunication Assembly,

considering

a) that the ability to automatically measure the quality of broadcast video has long been recognized as a valuable asset to the industry;

b) that Recommendation ITU-R BT.1683 describes objective methods for measuring the perceived video quality of standard definition digital broadcast television in the presence of a full reference;

c) that Recommendation ITU-R BT.709 describes parameter values for the HDTV standards for production and international programme exchange and Recommendation ITU-R BT.500 describes subjective assessment methods for image quality including high-definition television;

d) that HDTV is becoming widely used in broadcasting;

e) that ITU-T Study Group 9, based on the results of the HDTV report sent by VQEG, has produced Recommendation ITU-T J.341, which specified objective perceptual video quality measurement of HDTV in the presence of a full reference;

f) that objective measurement of the perceived video quality of HDTV may complement subjective assessment methods,

recommends

that the objective video quality model given in Annexes 1, 2 and 3 should be used for objective measurement of perceived video quality for broadcasting applications using HDTV in the presence of a full reference signal.

* Radiocommunication Study Group 6 made editorial amendments to this Recommendation in February 2020 in accordance with Resolution ITU-R 1.

Annex 1

1 Introduction

This Recommendation provides a perceptual video quality measurement method for use in high definition television (HDTV) non-interactive applications when the full reference (FR) measurement method can be used. The model was developed to estimate subjective quality scores.

The full reference measurement method can be used when the unimpaired reference video signal is readily available at the measurement point, as may be the case of measurements on individual equipment or a chain in the laboratory or in a closed environment such as a television broadcast station. The estimation method includes both calibration and objective video quality estimations.

The validation test material contained both H.264 and MPEG-2 coding degradations and various transmission error conditions (e.g. bit errors, dropped packets). The model in this Recommendation may be used to monitor the quality of deployed networks to ensure their operational readiness. The visual effects of the degradations may include spatial as well as temporal degradations. The model in this Recommendation can also be used for lab testing of video systems. When used to compare different video systems, it is advisable to use a quantitative method (such as that in ITU-T J.149) to determine the model's accuracy for that particular context.

This Recommendation is deemed appropriate for broadcasting services delivered between 1 Mbit/s and 30 Mbit/s. The following resolutions and frame rates were considered in the validation test:

− 1080/59.94/I

− 1080/25/P

− 1080/50/I

− 1080/29.97/P

The following conditions were allowed in the validation test for each resolution:

Test factors
Video resolution: 1920x1080 interlaced and progressive
Video frame rates 29.97 and 25 frames per second
Video bitrates: 1 to 30 Mbit/s
Temporal frame freezing (pausing with skipping) of maximum 2 seconds
Transmission errors with packet loss
Conversion of the SRC from 1080 to 720/P, compression, transmission, decompression, and then conversion back to 1080.
Coding technologies
H.264/AVC (MPEG-4 Part 10)
MPEG-2

Note that 720/P was considered in the validation test plan as part of the test condition (HRC). Because currently 720/P is commonly up-scaled as part of the display, it was felt that 720/P HRCs would more appropriately address this format.

1.1 Application

The applications for the estimation model described in this Recommendation include, but are not limited to:

1) Potentially real-time, in-service quality monitoring at the source;
2) Remote destination quality monitoring when a copy of the source is available at the point of measurement;
3) Quality measurement for monitoring of a storage or transmission system that utilizes video compression and decompression techniques, either a single pass or a concatenation of such techniques;
4) Lab testing of video systems.

1.2 Limitations

The video quality estimation model described in this Recommendation cannot be used to replace subjective testing. Correlation values between two carefully designed and executed subjective tests (i.e. in two different laboratories) normally fall within the range 0.95 to 0.98. If this Recommendation is utilized to make video system comparisons (e.g. comparing two codecs), it is advisable to use a quantitative method to determine the model's accuracy for that particular context.

When frame freezing was present, the test conditions typically had frame freezing durations less than 2 seconds. The model in this Recommendation was not validated for measuring video quality in a re-buffering condition (i.e. video that has a steadily increasing delay or freezing without skipping). The model was not tested on other frame rates than those used in TV systems (i.e. 29.97 frames per second and 25 frames per second, in interlaced or progressive mode).

It should be noted that in case of new coding and transmission technologies producing artefacts which were not included in this evaluation, the objective model may produce erroneous results. Here, a subjective evaluation is required.

2 References

None.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 Subjective assessment (picture): The determination of the quality or impairment of programme like pictures presented to a panel of human assessors in viewing sessions.

3.1.2 Objective perceptual measurement (picture): The measurement of the performance of a programme chain by the use of programme-like pictures and objective (instrumental) measurement methods to obtain an indication that approximates the rating that would be obtained from a subjective assessment test.

3.1.3 Proponent: An organization or company that proposes a video quality model for validation testing and possible inclusion in an ITU Recommendation.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 Frame rate: The number of unique frames (i.e. total frames – repeated frames) per second.

3.2.2 Simulated transmission errors: Errors imposed upon the digital video bit stream in a highly controlled environment. Examples include simulated packet loss rates and simulated bit errors. Parameters used to control simulated transmission errors are well defined.

3.2.3 Transmission errors: Any error imposed on the video transmission. Example types of errors include simulated transmission errors and live network conditions.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ACR	Absolute Category Rating (see ITU-R BT.500)
ACR-HR	Absolute Category Rating with Hidden Reference (see ITU-T P.910)
AVI	Audio Video Interleave
DMOS	Difference Mean Opinion Score
FR	Full Reference
FRTV	Full Reference TeleVision
HRC	Hypothetical Reference Circuit
ILG	VQEG's Independent Laboratory Group
MOS	Mean Opinion Score
MOSp	Mean Opinion Score, predicted
NR	No (or Zero) Reference
PSNR	Peak Signal-to-Noise Ratio
PVS	Processed Video Sequence
RMSE	Root Mean Square Error
RR	Reduced Reference
SFR	Source Frame Rate
SRC	Source Reference Channel or Circuit
VQEG	Video Quality Experts Group
YUV	Colour Space and file format

5 Conventions

None.

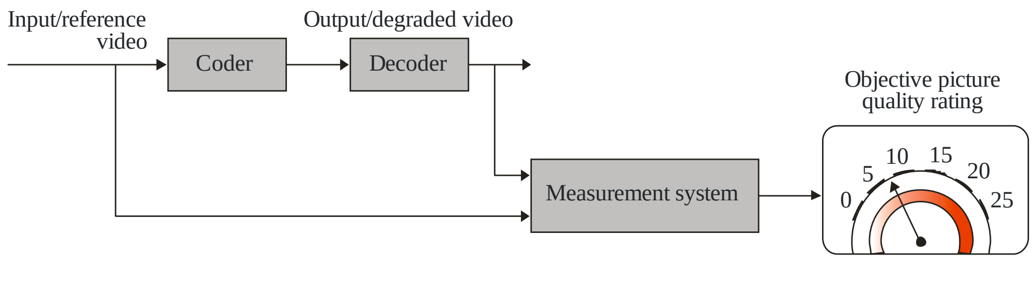
6 Description of the full reference methodology

The double-ended measurement method with full reference, for objective measurement of perceptual video quality, evaluates the performance of systems by making a comparison between the undistorted input, or reference, video signal at the input of the system, and the degraded signal at the output of the system (Fig. 1).

Figure 1 shows an example of application of the full reference method to test a codec in the laboratory.

FIGURE 1

Application of the full reference perceptual quality measurement method to test a codec in the laboratory



BT.1907-01

The comparison between input and output signals may require a temporal alignment or a spatial alignment process, the latter to compensate for any vertical or horizontal picture shifts or cropping. It also may require correction for any offsets or gain differences in both the luminance and the chrominance channels. The objective picture quality rating is then calculated, typically by applying a perceptual model of human vision.

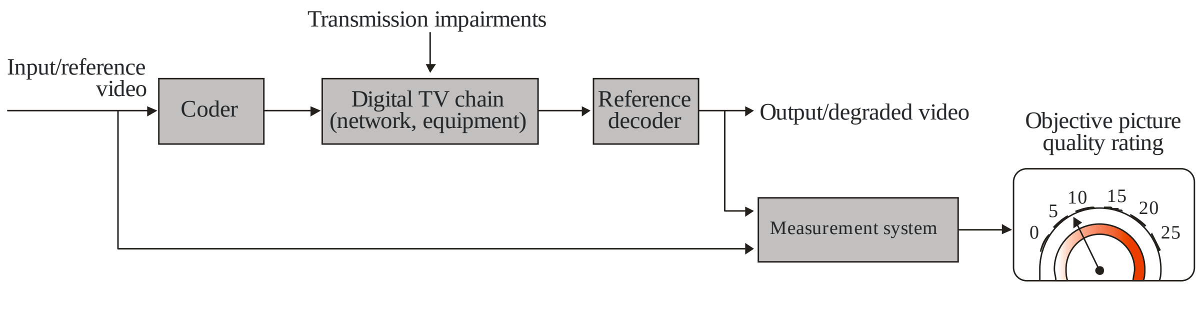
Alignment and gain adjustment is known as registration. This process is required because most full reference methods compare reference and processed pictures on what is effectively a pixel-by-pixel basis. The video quality metrics described in Annex 2 include registration methods.

As the video quality metrics are typically based on approximations to human visual responses, rather than on the measurement of specific coding artefacts, they are in principle equally valid for analogue systems and for digital systems. They are also in principle valid for chains where analogue and digital systems are mixed, or where digital compression systems are concatenated.

Figure 2 shows an example of the application of the full reference method to test a transmission chain.

FIGURE 2

Application of the full reference perceptual quality measurement method to test a transmission chain



BT.1907-02

In this case, a reference decoder is fed from various points in the transmission chain, e.g. the decoder can be located at a point in the network, as in Fig. 2, or directly at the output of the encoder, as in Fig. 1. If the digital transmission chain is transparent, the measurement of objective picture quality rating at the source is equal to the measurement at any subsequent point in the chain.

It is generally accepted that the full reference method provides the best accuracy for perceptual picture quality measurements. The method has been proven to have the potential for high correlation with subjective assessments made in conformity with the ACR-HR methods specified in ITU-T P.910.

7 Findings of the Video Quality Experts Group (VQEG)

Studies of perceptual video quality measurements are conducted in an informal group, called the Video Quality Experts Group (VQEG), which reports to ITU-T Study Groups 9 and 12 and ITU-R Study Group 6. The recently completed high definition television phase I test of VQEG assessed the performance of proposed full reference perceptual video quality measurement algorithms.

The following statistics are taken from the final VQEG HDTV report. Note that the body of the VQEG HDTV report includes other metrics including Pearson Correlation and RMSE calculated on individual experiments, confidence intervals, statistical significance testing on individual experiments, analysis on subsets of the data that include specific impairments (e.g. H.264 coding-only), scatter plots, and the fit coefficients.

Primary analysis

The performance of the FR model is summarized in Table 1. The PSNR is calculated according to ITU-T J.340 and included in this analysis for comparison purposes. "Superset RMSE" identifies the primary metric (RMSE) computed on the aggregated superset (i.e. all six experiments mapped onto a single scale). "Top performing group total" identifies the number of experiments (0 to 6) for which this model was either the top performing model or statistically equivalent to the top performing model. "Better than PSNR total" identifies the number of experiments (0 to 6) for which the model was statistically better than PSNR. "Better Than Superset PSNR" lists whether each model is statistically better than PSNR on the aggregated superset. "Superset Correlation" identifies the Pearson Correlation computed on the aggregated superset.

TABLE 1

Metric	**PSNR**	**SwissQual**
Superset RMSE	0.71	0.56
Top performing group total	1	5
Better than PSNR total	–	4
Better than superset PSNR	–	Yes
Superset correlation	0.78	0.87

Annex 2

Model description

Editor's note: The source code to be incorporated into this section forms a mandatory part of this Recommendation and is available at http://ifatemp.itu.int/t/2009/sg9/exchange/q2/.

Overview of the model

The model predicts the video quality as it is perceived by subjects in an experiment. The prediction model uses psycho-visual and cognitive-inspired modelling to emulate subjective perception.

As a full reference approach, the model compares the input or high-quality reference video and the associated degraded video sequence under test. This process is shown in Fig. 3.

Score estimation is based on the following steps:

1) First, the video sequences are preprocessed. In particular, noise is removed by filtering the frames and the frames are sub-sampled.

2) A temporal frame alignment between reference and processed video sequence is performed.

3) A spatial frame alignment between processed video frame and the corresponding reference video frame is performed.

4) Local spatial quality features are computed: a local similarity and a local difference measure, inspired by visual perception.

5) An analysis of the distribution of the local similarity and difference feature is performed.

6) A global spatial degradation is measured using a *blockiness* feature.

7) A global temporal degradation is measured using a *jerkiness* feature. The jerkiness measure is computed by evaluating local and global motion intensity and frame display time.

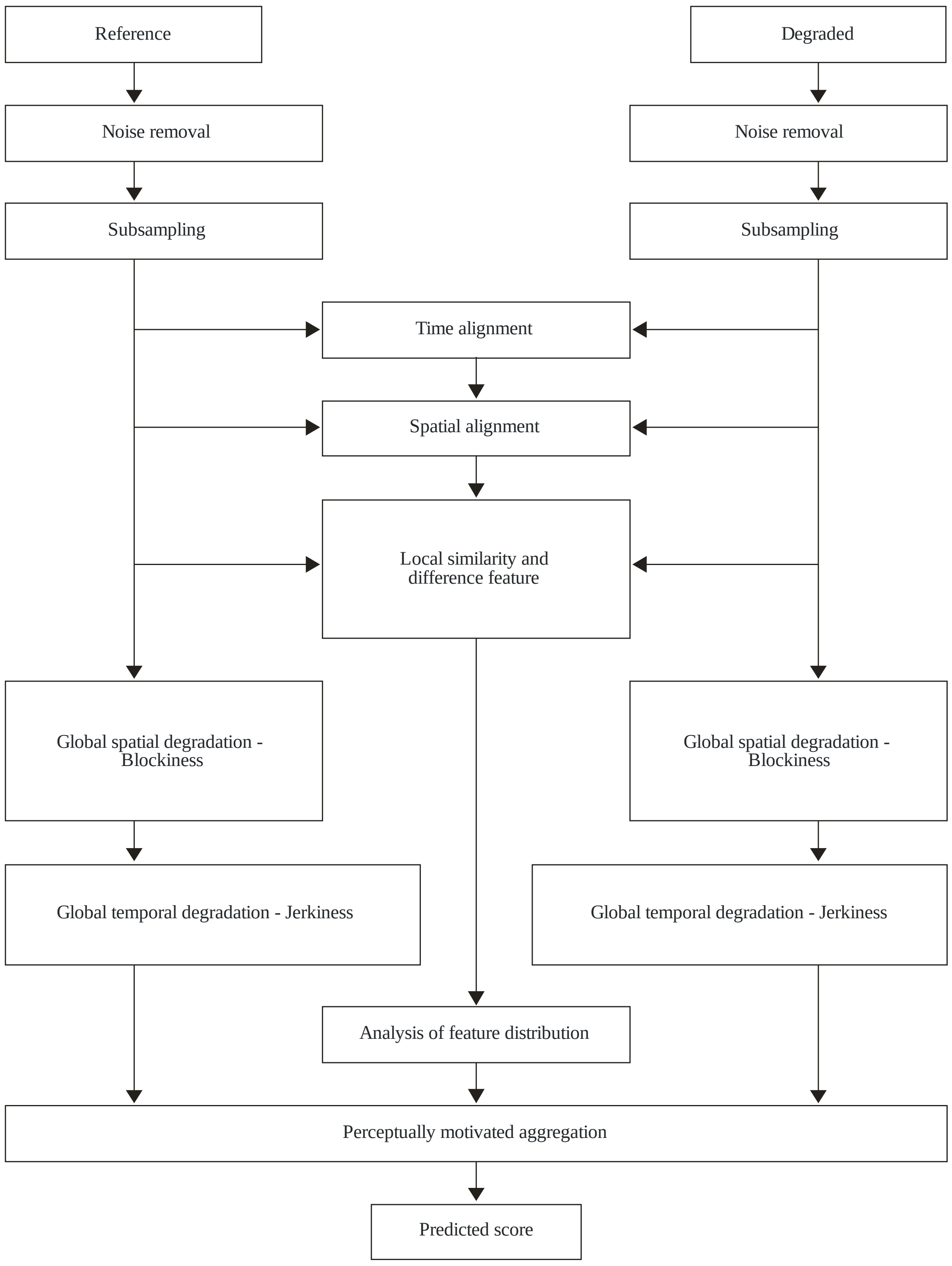
8) The quality score is estimated based on a non-linear aggregation of the above features.

9) To avoid mis-prediction in case of relatively large spatial misalignment between reference and processed video sequence, the above steps are computed for 3 different horizontal and vertical spatial alignments of the video sequence, and the maximum predicted score among all spatial positions is the final estimated quality score.

The individual steps are explained in more detail in §§ 2.1 through 2.9. Section 2.10 contains an embedded archive with C++ source code covering the essential parts and functions for an implementation compliant description of the model. The C++ function names mentioned in §§ 2.1 through 2.9 are referring to this reference source code (e.g. § 2.2 refers to `CFrameAnalysisFullRef::ContentTimeAlignment`).

FIGURE 3

Flow chart overview of the processing steps of the model. On top, the input is the reference and degraded (or processed) video sequences. Different processing steps yield the main model output, the predicted score at the bottom


Reference
Degraded
Noise removal
Noise removal
Subsampling
Subsampling
Time alignment
Spatial alignment
Local similarity and difference feature
Global spatial degradation - Blockiness
Global spatial degradation - Blockiness
Global temporal degradation - Jerkiness
Global temporal degradation - Jerkiness
Analysis of feature distribution
Perceptually motivated aggregation
Predicted score

BT.1907-03

2.1 Preprocessing

Each frame of the reference and the processed video sequence is spatially low-pass filtered and subsampled to 3 different resolutions, R1, R2, R3:

	original frame		R1		R2		R3
height x width	1080 × 1920	→	540 × 960	→	270 × 480	→	96 × 128

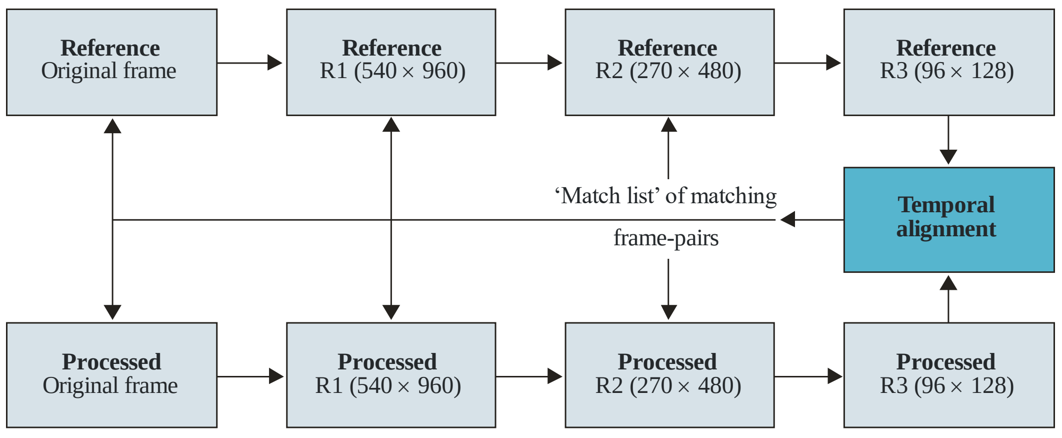
See method `CFrameAnalysisFullRef::ContentTimeAlignment` generating frames at resolution R3 and `CFrameSeq::ReadFrame` for the generation of frames at resolution R1 and R2.

Note that the implementation is not straightforward due to memory constraints.

Figure 4 shows the three subsampled resolutions.

FIGURE 4

The frames of the reference and processed video sequence are low-pass filtered and subsampled to 3 different resolutions. The smallest resolution R3 is used to perform the frame time alignment. The resulting list of matched frames can be used to match the frames at any other resolution



BT.1907-04

2.2 Time alignment

The time alignment is performed using the reference and processed video sequence at the low resolution R3.

Time alignment is performed in a recursive manner as follows:

1) Find an 'anchor' frame in the reference sequence (`Ref_anchor`).

2) Match this frame to the best matching frame in the degraded sequence (`Deg_best_match`).

Take this best matched frame in the degraded sequence (Deg_best_match) and match it to frames close to the 'anchor' frame of the reference (Ref_anchor). Try to find a better match, according to a similarity criterion, between the Deg_best_match and frames in the environment of Ref_anchor and store it as best matching pair. As similarity criterion between the Y-plane of the processed frame `x` and the reference frame `y`, the function:

```
sim = exp(-mean_square_diff(a*x+b,y))
```
(2.1)

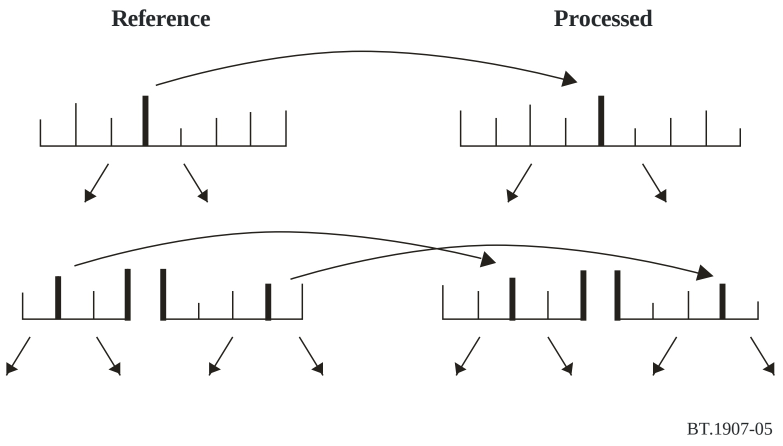
is used, with the parameters `a,b` chosen, such that the mean square difference is minimized between the values of the Y-plane of the processed frame `x` and the reference frame `y`, see method `FrameSimilarity::similarity` in the reference implementation.

1) If this matched frame-pair is a good match (similarity criterion has passed an acceptance threshold), split the reference and processed video sequence at the matching frame-pair, each into two video sequences before and after the matching frames. Start at 1) with both pairs of reference and degraded subsequences.

2) If the matching frame-pair is not a 'good match', start again at 1) with a different 'anchor' frame of the reference video. As there is no *a priori* knowledge of the expected value of a 'good' matching frame, the matching threshold is iteratively lowered. The following values were determined based on many training data samples: The starting threshold, with respect to the similarity criterion of equation (2.1), is 0.98. After failing to match 10 anchors it is lowered by a factor of 0.98, and the matching is restarted at 1). This way, at most 10 further anchors are tried, if they fail, the limit is again lowered. This proceeds until reaching a minimum value of 0.1. See `SQ_TimeAlignement::findAncorAndDescend` for full implementation details.

FIGURE 5

Illustration of the recursive approach used for time alignment. An anchor frame of the reference is matched to a frame of the processed sequence. Then both sequences are split and in each subsequence an anchor frame is chosen and matched



The result of the time alignment is a sequence (basically a 'match list'), assigning each frame of the processed video sequence a frame of the reference, or an indicator, indicating that no sufficiently good matching frame could be found. Thus, for the later processing stages, each matched frame of the processed video sequence has a corresponding frame of the reference video. Those frames of the processed video sequence having an indicator 'unmatched' will be compared to the two reference frames matching the previous and following 'matched' frame of the processed video sequence. Note that the 'matching limit' is chosen to be very low, such that only very strongly degraded frames have an indicator 'unmatched'.

See method `CFrameAnalysisFullRef::sqVTA_ContentFrameTimeAlignement_M` for all implementation details.

2.3 Spatial frame alignment

Iterate over all frames of the processed video sequence and:

1) If this frame is unmatched, use the previous spatial alignment. If this frame is matched, perform a spatial alignment between the processed and corresponding – according to the match list of the time alignment – reference frame:

a) For the first frame, initialize the spatial shift to be 0 (in both horizontal and vertical directions). For subsequent frames, use as a prior, the spatial alignment of the previous matched frame.

b) Iterate over all possible spatial shifts (horizontal and vertical), using the limit of point 2) below. If a different spatial shift leads to a significantly (with respect to a cost-function) smaller difference between the processed and corresponding reference frame, the spatial shift is adjusted. As a cost function the function:

```
rmse(Y(dv,dh),Y_ref) + abs(dv)+abs(dh),
```

is used, where Y denotes the Y-plane of the processed frame at resolution R1 and Y_ref denotes the reference frame at resolution R1, Y(dv,dh) denotes the shifted frame Y, shifted by dv and dh, where dv, dh are the vertical and horizontal shifts. The second term and third term are included in the cost function, to favour small spatial shifts. Note that a small border of the frames is skipped for the computation of the rmse, to avoid a more complicated border handling.

c) That way, time variant spatial shifts can be compensated. Mis-alignments at one frame can be corrected by the alignment of subsequent frames.

2) This first step of the automated spatial shift alignment is limited to ±4 pixels. For larger spatial shifts, see § 2.9.

3) After the spatial alignment, each frame in the processed video sequence has a corresponding reference frame (or two in the case of an unmatched frame) according to the match-list from time alignment and a well-defined spatial shift correction. Thus, the frames of the processed video sequence can be accurately compared to frames of the reference. This is fundamental for the following feature extractions.

See method `CFrameAnalysisFullRef::DetermineSpatialAlignment` for all implementation details. The constant threshold in step 2 (±4 pixels) can be increased to accommodate larger spatial shifts.

2.4 Computation of local similarity and local difference features

For each aligned frame-pair, a set of spatial quality features are calculated:

First, a local *similarity* and *difference* measure is computed by iterating over abutting, equally distributed squared regions of size 13 x 13 of the processed and reference frame at resolution R2. As the resolution R2 does not divide by 13, a small border is ignored.

The local regions are called `s_prc` and `s_ref`, and the similarity `S` and difference `D` are computed by:

```
S = (cor( s_prc,s_ref) + 25) / (var(s_ref) + 25)          (4.1)
      D = sqrt(avg(( S*( s_prc-mean( s_prc)) -
              (s_ref-mean(s_ref)))^2))                    (4.2)
```

where `cor` is the correlation and `var` is the variance of the pixel values in the corresponding squared region. The function `avg` computes the average over all pixels of the squared region, and `sqrt` denotes the square root. The values `D` and `S` are the main contributor to the spatial quality value.

At this point, the similarity and difference feature S, D are a matrix of values for each frame, one value corresponding to each squared local region. Important for the perceived quality is not only the mean value, but the form of the distribution of S, D, respectively.

2.5 Analysis of the distribution of local features

This section starts with the introduction of some notations:

Let `quantile` (X,c) denote the c-*quantile* of the distribution of the values (entries) of a vector or matrix X. More precisely, for a vector X and a constant c with 0<=c<=1, the quantile

q = `quantile` (X,c)

is the value q, such that a fraction c of all the values of X is smaller than or equal to q.

The function *trimmed_mean* is defined as follows. This notation will be used later. For a matrix `X`, the *trimmed mean*

`trimmedMean`(X,c)

is the mean of all entries of `X` between the c and `(1-c)` quantiles of `X`.

E.g. `trimmedMean(X,0.1)` is the mean of all values of `X`, ignoring 10% of the smallest and 10% of the largest values of `X`.

The notation `X(X>c)` denotes the set of all values of `X` larger than `c`. E.g.

```
trimmedMean(X,c) = mean(X(X>quantile(X,c) and X<quantile(X,1-c)))
```

Using these notations, the following feature values are computed based on `S` from equation (4.1), and `D` from equation (4.2):

`s_m = trimmedMean(S,c)` (5.1)

`d_m = trimmedMean(D,c)` (5.2)

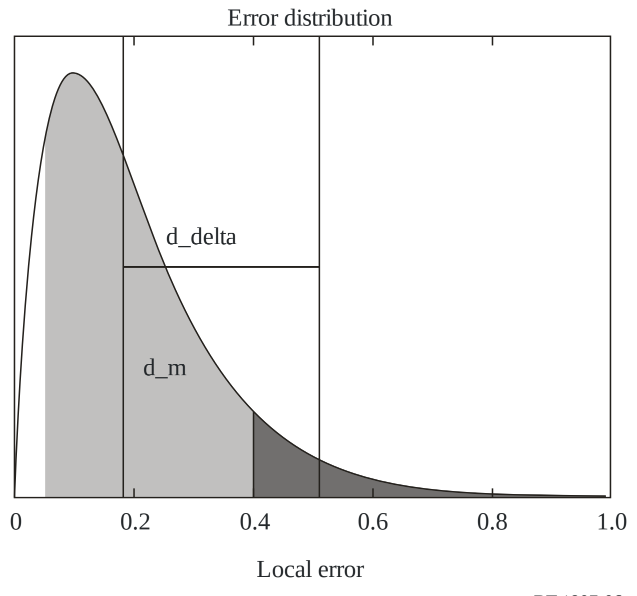
`s_delta = s_m – mean(S(S<quantile(S,c)))` (5.3)

`d_delta = mean(D(D>quantile(D,1-c))) – d_m` (5.4)

using `c=0.2`. This is shown visually in Fig. 6, which presents `d_m` and `d_delta`.

FIGURE 6

Shown is the distribution of a local feature D. The trimmed mean d_m corresponds to the mean of the light grey region (black vertical line). The value d_delta corresponds to the difference of the mean of the values in the dark grey and light grey region (horizontal line)



BT.1907-06

See method `CFrameAnalysisFullRef::ComputeSimilarity` for the computation of `S` and `D`.

2.6 Computation of the blockiness feature

A *Blockiness* feature is computed using the frames at resolution R1. This feature measures the visibility of block borders introduced by coding and/or transmission errors. Due to the computation at resolution R1, automatically a focus on the perceptual visibility of edges is set. Starting with an overview, the blockiness feature computes:

1) Directional derivatives (edge images) for horizontal and vertical edges. This results in two matrices, one for horizontal and one for vertical edges, for each frame of the video sequence, called `verGrad_n` and `horGrad_n` in the pseudo code below.

2) A row-wise and column-wise sum of the logarithm of horizontal and vertical edges, resulting in two vectors, one corresponding to the sum of horizontal, one corresponding to the sum of vertical edges, called `sumW` and `sumH` below.

3) An average of a subsample of `sumW` and `sumH`, respectively, at a step size n and offset m, computed by the function `vq_AvgSubsample` below.

The idea is that a strong block structure of blocks of size n will show as an important difference `delta_edge` in the `vq_AvgSubsample` at step size n computed for different offsets.

E.g. a block structure of size 4 in the original frame has a block structure of size 2 at resolution R1. Therefore, computing `vq_AvgSubsample(x,2,0)` and `vq_AvgSubsample(x,2,1)` should show an important difference, if a strong block structure is present. To avoid a content dependency, experiments using a large sample of video sequences showed how to relate the computed difference measured for the processed video sequence to the values of the reference sequence.

More details of the computation are best explained using the following pseudo code. Here, `horGrad` and `verGrad` are the horizontal and vertical spatial derivatives of a frame, given by the difference of adjacent pixels,

`verGrad_n(i,j) = Y_n(i+1,j) - Y_n(i,j)`, and

`horGrad_n(i,j) = Y_n(i,j+1) - Y_n(i,j)`,

where `Y_n(i,j)` denotes the pixel value at position `(i,j)` of the Y-plane of frame `n`. The function

`vq_AvgSubsample( x, step, offset )`

calculates the average value of the vector `x` over all samples at a step size `step` and starting at offset `offset`.

```
// loop over all frames and compute:
for( UINT i=0; i<horGrad.Height; i++ ){
       for( UINT j=0; j<horGrad.Width; j++ ){
           w = (double)verGrad(i,j);
           h = (double)horGrad(i,j);
           // sum edges (-2: small differences can be the result of integer
          // values used to store frames)
           sumW(i) += log(1.0+max(0.0,fabs(w)-2.0));
           sumH(j) += log(1.0+max(0.0,fabs(h)-2.0));
       }
}

double dH0 = vq_AvgSubsample( sumH, 2, 0 );
double dH1 = vq_AvgSubsample( sumH, 2, 1 );
double dW0 = vq_AvgSubsample( sumW, 2, 0 );
double dW1 = vq_AvgSubsample( sumW, 2, 1 );
```

```
edge_max = 0.5 * (vq_Max(dW0,dW1) + vq_Max(dH0,dH1) );
edge_min = 0.5 * (vq_Min(dW0,dW1) + vq_Min(dH0,dH1) );

// now: denote by edge_max(i) the value of edge_max above, corresponding to
// frame i of the processed video sequence, and by edge_max_ref(i) the values
// edge_max above corresponding to frame i of the reference video sequence,
// and analogously for edge_min(i), edge_min_ref(i). Then compute:

for( UINT i=0; i<nbOfFramesInProcessedVideo; i++ ){
     // get frame nb of ref frame (according to match-list)
       UINT i_ref = (UINT)floor(ref_frameNb_all(i)+0.5f);

       float delta_edge = edge_max(i) - edge_min(i);
       float delta_edge_ref = edge_max_ref(i_ref) - edge_min_ref(i_ref);

       x(i) = vq_Max(0.0f,delta_edge - delta_edge_ref) / (1.0f+edge_max(i));
}
// blockiness(i) is then a non-linear monotone transform of x(i) ...
```

Note that due to the possibility of upsampled 720-frames, the calculation is slightly more complicated:

See `vquad_hd::vq_BlockinessPhaseDiff` and `CQualityModelFullRef::Blockiness` for all implementation details of the computation of the blockiness feature.

2.7 Computation of jerkiness feature (temporal quality)

A *jerkiness* feature is computed by averaging the product of relative display time, a non-linear transform of display time, and a non-linear transform of motion intensity. The motion intensity is mainly derived by inter-frame differences on individual regions of the frame. The *display time* is the time, in milliseconds, a frame is displayed on the screen. Note that to determine the display time of each frame, a local motion intensity analysis is performed, as frames in the processed video sequence might be repetitions of previous frames.

The jerkiness feature takes into account the amount of missed information during playback of the processed video sequence. It is very low in case of a fluently played sequence, while it increases in case of pauses or lowered frame rates. On the other hand, for a fixed temporal impairment, the jerkiness measure takes larger values for video sequences with larger motion intensity.

The following pseudo-code shows the details. Note that the inputs are the motion intensity vector `motionInt`, the vector of frame-repetition probabilities `repFrame` and a vector of frame display times `displayTime`. The output is the vector `jerkiness`, the jerkiness at each frame of the processed video sequence. In more detail, the vector `motionInt` denotes the root mean square interframe difference, measured on the Y plane at resolution R2. The vector `repFrame` denotes the probability of frame repetition, i.e. depending on motion intensity, each frame has a probability to be a repetition of the previous frame: in case of a perfect repetition of the previous frame, the actual frame has probability 1 to be a repetition. In case of a large motion intensity, the actual frame has probability 0 to be a repetition of the previous frame. Intermediate values of probabilities can occur if the motion intensity has very small but non zero values. Explicitly,

$$\text{repFrame}(i) = \begin{cases} 0 & \text{if } m(i) < p/2 \\ (m(i) - p/2)/p & \text{if } p(2) <= m(i) < 3/2*p \\ 1 & \text{if } 3/2*p <= m(i) \end{cases}$$

where `m(i)` denotes the motion intensity of frame `i`. Empirically, the parameter p=0.01 was chosen.

```
int vq_CalcJerkiness( const CVector<float> & motionInt,
                               const CVector<float> & repFrame,
                               const CVector<float> & displayTime,
                               CVector<float> & jerkiness ){

      // ----------------------------------------------------------
      // the 4 parameters of the jerkiness measure: determined using a
   // large sample of video sequences containing temporal degradations
   // only.
      float a = 0.9f;
      float b = 5.0f;

      float aT = 40.0f;
      float bT = 5.0f;
      // ----------------------------------------------------------

      // get probability of new frame = 1 - prob of repeated frame:
      CVector<float> newFrame = repFrame*(-1.0f) + 1.0f;

      // count number of non-repeated frames
      float fNbRepeated = repFrame.Sum();
      float fNbNonRepeated = repFrame.Length() - fNbRepeated;

      // calculate jerkiness
      float fR = 0.0f;
      // look for frame repetition intervals (=~ display time)
      // of length i
      for( UINT i=1; i<= iNbFrames; i++ )
      {
         // look for frame repetitions starting at position j
         for( UINT j=0; j<iNbFrames-i+1; j++ )
         {
            float fP = newFrame(j);   // prob. : start of repetition block

            for( UINT k=1; k<i; k++ )
            {
               fP *= repFrame(j+k);  // prob. : all repeated frames
            }
            if( i+j < iNbFrames )
            {
               fP *= newFrame(j+i);  // prob. : end of repetition block
            }

            // calculate the display time (in s) of frame j,
      // if displayed from
            // time t_j until t_(j+i), which occurs with probability fP
            float fDispTime = displayTime.SumPart(j,j+i)/1000.0f;

            // -> measure jerking and add to result
            float fIFDiff = motionInt( j+i-1 );

            // normalisation values: such that at 0 jerkiness value is 0,
      // and saturates at 1
```

```
              float c = 1.0f / (1.0f+exp(b));
              float cT = 1.0f / (1.0f+exp(bT));

              float fJ = 1.0f / (1.0f + exp( -( a * fIFDiff  - b) ));
              float fJT = 1.0f / (1.0f + exp( -( aT * fDispTime - bT)));
              fJ = (fJ - c )/(1.0f - c);
              fJT = (fJT - cT)/(1.0f - cT);

              // jerkiness value: propability * interframeDiffFactor *
        //         displayTimeFactor

              fR = fP * fJ * fJT * fDispTime;

              // add to jerkiness vector at position j+i (corresponding to the
              // end of display time)
              jerkiness( vq_Min(j+i,iNbFrames-1) ) += fR;
          }
       }
       return 0;

}
```

See method `vquad_hd::vq_ProbOfRepeatedFrame` and `vquad_hd::vq_CalcJerkiness` for the implementation details.

2.8 Aggregation to MOS

The features described above, Similarity given by `s_m` and `s_delta`, Difference given by `d_m` and `d_delta`, `blockiness` and `jerkiness` are the basis for score estimation, together with the vector of frame display times `displayTime`. These are vectors with their lengths equal to the number of frames of the processed video sequence.

To map these features onto a perceptual scale, a parameterized S-shaped function is used:

$$S: \mathbb{R} \longrightarrow \mathbb{R}, \quad y = S(x).$$

The function S, is parameterized by the triple (p_x,p_y,q). The parameters have the following interpretation: `(p_x,p_y)` is the location in ℝ x ℝ and `q` the slope of the inflection point, in detail:

```
S(x) = a * x^b        if x<=p_x

d/(1+ exp(-c*(x - p_x))+1-d)   else
```
(8.1)

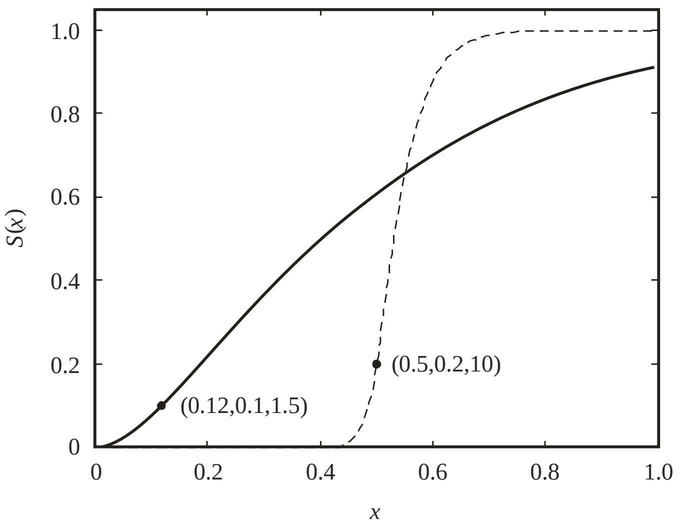
where:

```
a =    p_y/p_x^(q*p_x/p_y)
b =    q*p_x/p_y
c =    4*q/d
d =    2*(1-p_y)
```

See Fig. 7 for a plot of the S-function with different parameters. The S-shaped function starts at the origin, grows polynomially up to the inflection point and saturates exponentially towards 1.

FIGURE 7

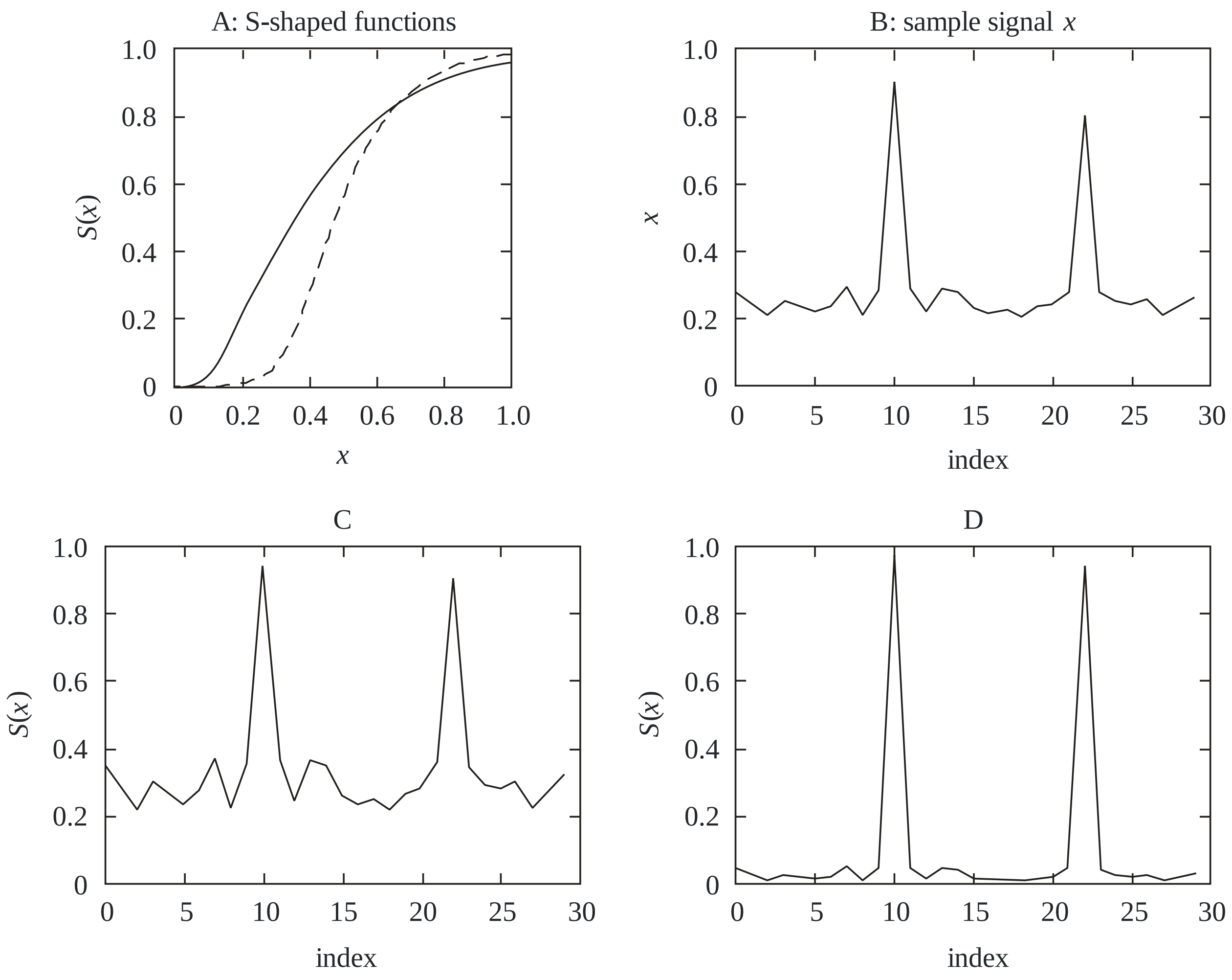
S-shaped functions, parameterized by the position and slope of the inflection point. Two sample functions are shown for different parameters



BT.1907-07

FIGURE 8

Plot (A) shows S-shaped functions for 2 different parameter sets. Plot (B) shows a sample signal vector. All values of x are transformed using the two S-shaped functions of (A). The result is shown in (C) and (D). The S-shaped transformation is used to map a feature x to a perceptual scale, using parameters fitted to data samples, indicated in (C). Using an S-shaped function (dashed line in (A)) with data dependent parameters (the first parameter is set to a multiple of the signal mean of x), a detector for transient degradations can be built, (D)



BT.1907-08

Using an S-shaped function with constant parameters, a feature value can be mapped to a perceptual scale. Using data-dependent parameters, the S-shaped function can be used, e.g. to compress all values below the mean to a small value, and stretch all values above. This way, it can be used to compute a feature sensitive to transient degradations, see Fig. 8.

To continue with the model description, first, define a degradation based on the similarity features above, but putting more weight on the strongest degradations:

```
d_s = 1 - s_m + 1.5 s_delta.
```

The next idea is to use two S-shaped functions, one using a fixed set of parameters `cod_par`, to transform `d_s` to `d_cod` on a perceptual scale related to a base degradation, reflecting errors due to video encoding.

A second S-shaped function with *data-dependent* parameters, to transform `d_s` to `d_trans` on a perceptual scale related to transient degradations, reflecting transmission errors.

In detail, relating to the pseudo code below, call the function `SplitCodTrans` using as input the vector `d_s` and the vector of frame display times `disp_time` and having output `d_cod, d_trans,`

```
SplitCodTrans(d_s, disp_time, d_cod, d_trans).
```

The following pseudo-code shows the function details. Note that

```
stat.STransform(x,px,py,q)
```

denotes the S-shaped function and has as input the real value `x` that will be transformed, and the three parameters denoted `(px,py,q)` in equation (8.1).

```
SplitCodTrans( const CVec& v, const CVec& dispTime,
        CVec& v_cod, CVec& v_trans ){

   // these parameters are determined empirically
       float qPosSmall = 0.55f;
       float qPosLarge = 0.65f;

   // q is the mean of the values in v between the qPosSmall and qPosLarge
   // quantiles.
       float q = r.TrimmedMean( qPosSmall, qPosLarge, v, dispTime );

       for( UINT i=0; i<v.Length(); i++ ){

     // the parameters used here are the result of fitting to sample
     // data
          v_cod(i) = stat.STransform(v(i), 0.07f, 0.1f, 2.0f);

     // Note that the STransform is not directly applied to v, but
     // to v(i)-q . This is prefered here for numerical reasons of the
     // resulting STransform.

          // v_trans is part of v above quantile value q
          v_trans(i) = vq_Max(0.0f, v(i)-q);

          float px = 0.5f * (q+0.2f);
     // the parameters used here are the result of fitting to sample
     // data
          v_trans(i) = stat.STransform(v_trans(i),px, 0.1f,16.0f);
       }

}
```

See `CQualityModelFullRef::SplitCodTrans` for full implementation details.

In analogy, `d_diff_cod, d_diff_trans` are derived from `d_m, d_delta`, by setting

```
d_diff = d_m + 1.5 d_delta,
```

and calling

```
SplitCodTrans(d_diff, disp_time, d_diff_cod, d_diff_trans),
```

using the parameter triples for the two S-transforms

```
cod_par = (4.0f,0.05f,0.2f)
trans_par = (0.5*(q+4.0f),0.1f,0.4f),
```

where `q` denotes the interquantile mean, as in the pseudo-code above. The outputs of the function are `d_diff_cod, d_diff_trans`.

Next, a feature value related to transient large jerkiness values is calculated by an S-shaped transform of jerkiness:

```
d_t_trans = S(max(0,jerkiness-q))
```

with the parameters of the S-shape transform given by `(max(0.048,q), 0.2,40.0)` and `q` denotes the interquantile mean between the 0.55 and 0.65 quantiles of the jerkiness vector. The parameters were determined by fitting to a large set of sample data.

See `CQualityModelFullRef:: SplitTempTrans` for full implementation details.

Next, the base quality `q_cod` is defined as:

```
q_cod = (1 - d_cod) * (1 - d_diff_cod) * (1-blockiness)
```
(8.2)

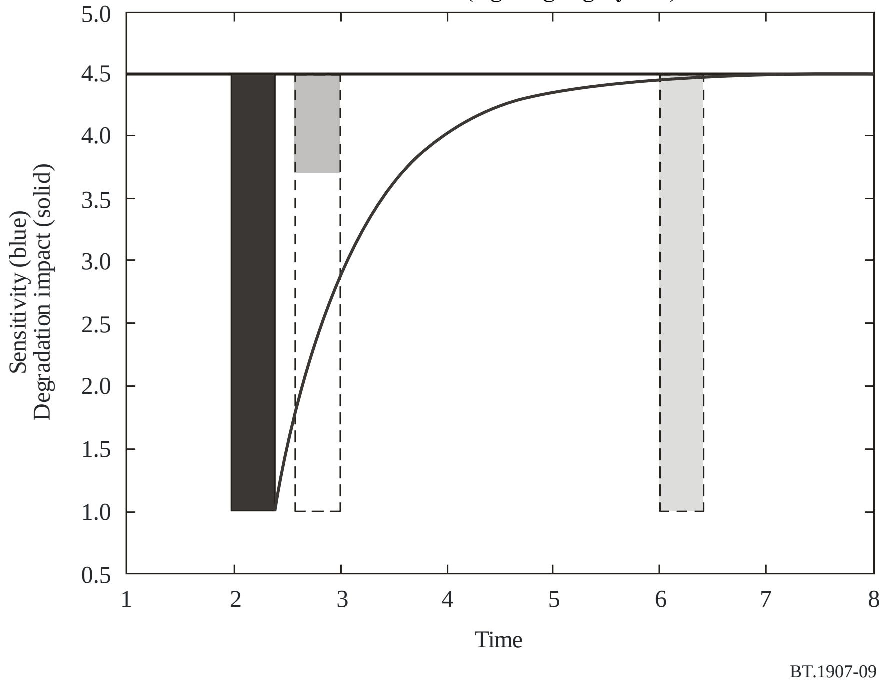
and the transient quality is defined as

```
q_trans = (1 - d_trans) * (1 - d_diff_trans) * (1 - d_t_trans).
```

The influence of an additional degradation is reduced if it is occurring shortly after a first degradation. To account for this effect, `q_trans` is transformed to `q_fq`. The idea is illustrated in Fig. 9.

FIGURE 9

The black solid bar indicates a degradation occurring in a hypothetical video sequence. The black solid line indicates the sensitivity to subsequent degradations. The influence of an additional degradation with strength given by the dashed line is reduced (first dark grey bar) if it is occurring shortly after a first degradation. The influence of a subsequent degradation is not altered if occurring with a large time interval in between (right light grey bar)



The details of the computation are best described by the following pseudo-code section, which uses `1-q_trans` as input vector `v`, the vector `disp_time` is the vector of frame display times. The decay time constant `dT`=1000 ms was determined using sample data; then set

```
q_fq = 1 - w,
```
(8.3)

where `w` is the output vector of the function:

```
DegFreq( const CVec& v, const CVec& disp_time, CVec& w, float dT ){

       // time constant for temporal integration of degradations
       float t_const = 80.0f;
       w(0) = v(0)*vq_Min(t_const,disp_time(0))/t_const;

       for( UINT i=1; i<v.Length(); i++ ){
          // integrate degradation over the last t_const milliseconds:
          // use an indicator function, which is 1 over the interval
          // [t_i-t_const, t_i] and 0 otherwise
          float dT_sum = 0.0f;
          UINT j=0;
          float v_sum = 0.0f;
          while( dT_sum < t_const && (int)i-(int)j>=0 ){
             // b is the overlap in [0,1] of the display time interval
             // of frame i-j with respect to the integration interval
             //       t_i-t_const   t_i
             //       ________________
             //       |        |       integration window
             // ---------------------------------------------------> time
             //  |   |     |    |     |    |  frames
             //
             //      ->| b*dT |<-
             //
             float b = vq_Min(t_const-dT_sum,disp_time(i-j)) / t_const;

             v_sum += v(i-j)*b;
             dT_sum += disp_time(i-j);
             j++;
          }
          // compute the fall-off factor a:
          float a = exp(-disp_time(i-1)/dT);
          // the degradation is now:
          // 1) a linear combination of the fall-off of a previous degradation
          //  and the current (integrated) degradation, or
          // 2) the integrated current degradation (if it is stronger than 1)
).
          w(i) = vq_Max(v_sum, a * w(i-1) + (1.0f-a) * v_sum);


       }
```

Averaging process

Let `disp_time(i)` denote the display time of frame `i`, and `jerkiness(i)` the jerkiness value corresponding to frame `i`. Recall `q_cod` from equation (8.2) and `q_fq` from equation (8.3). Set

```
Q_t  = 1 - 1/T sum_i jerkiness(i)

Q_fq = 1/T sum_i q_fq(i)* disp_time (i)

Q_cod = 1/T sum_i q_cod(i)* disp_time (i)
```

where `T=sum_i disp_time(i)`, and `sum_i` denotes the sum over all indices `i=0,..,number_of_frames`.

The final predicted score is a product and rescaling to the [1,5] MOS range:

```
s = 4 * Q_t * Q_cod * Q_fq + 1
```

See method `CQualityModelFullRef::PredictScoreCodTrans` for details about score prediction.

2.9 Handling of heavily spatially misaligned video sequences

To avoid misprediction in case of relatively large spatial misalignment between reference and processed video sequences, the above steps are computed for three different horizontal and vertical spatial alignment steps of the video sequence, and the maximum predicted score among all spatial positions is the final estimated quality score.

A step size of four pixels is used in each direction. That way, a spatial search range of ±8 pixels is realized. This covers easily the maximum used spatial shift in the test set (±5 pixels). Since, this enlargement is performed in a high-level function of the model, the alignment range can be easily adapted to either larger shifts or can be reduced (e.g. ±4 pixels) for saving computational resources.

See `vquad_hd::vq_vquad08`.

2.10 Reference source code implementation

This section contains an embedded archive with C++ source code covering the essential parts and functions for an implementation compliant description of the model. All links to actual implementations made in the sections above are referring to this reference source code.

Annex 3

Conformance testing

This Annex is completed by a digital attachment containing the following information and files:

1) Sixteen short HD video sequences (reference and degraded sequences for eight test cases). These sequences will cover different distortions and contents. They are provided for testing a conformant model implementation made by a user of this Recommendation against the reference implementation of the model. The video sequences consist of a few frames only to reduce storage capacity. They are not indented for any visual test, but only for a conformance test of the HD model implementation.

2) Predicted MOS scores for HD sequences mentioned under 1) as 'HD_ConformanceReferenceResults.xls'. These scores were obtained with a reference implementation of the HD model.

3) Predicted MOS scores for five public HD databases available via VQEG. These databases can be used as an extended conformance test for implementations of the model.

Conformance test criteria:

i) The eight reference scores given in 2) have to be exactly reproduced by an implementation of the model. Exact stands for reproduction of the score with a resolution of three decimal digits.

ii) The predicted MOS scores of the five public VQEG databases have to be reproduced with a very limited deviation. Minor variations are allowed, since the experience showed that different optimizations in speed and storage use can lead to minor and negligible deviations in the final score.

TABLE 2

Allowed distribution of differences across all conformance test data

Absolute difference	Allowed occurrence
> 0.0001	5.00%
> 0.001	1.00%
> 0.01	0.50%
> 0.1	0.05%
> 0.3	0.00%

For other databases from the ones defined in this Annex 3, the same error distribution must not be exceeded. For unknown data a test set of at least 500 file pairs – preferrably from complete subjective experiments – has to be taken for those statistics.

Bibliography

VQEG Final Report of HDTV Phase I Validation Test (2010), "*Video Quality Experts Group: report on the validation of video quality models for high definition video content*", Video Quality Experts Group (VQEG), http://www.its.bldrdoc.gov/vqeg/projects/hdtv

Recommendation ITU-R BT.601-7 (2011), Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios.

Recommendation ITU-T J.244 (2008), Calibration methods for constant misalignment of spatial and temporal domains with constant gain and offset.

Recommendation ITU-R BT.500-12 (2009), Methodology for the subjective assessment of the quality of television picture.

Recommendation ITU-T J.149 (2004), Method for specifying accuracy and cross-calibration of Video Quality Metrics (VQM).

Recommendation ITU-T J.247 (2008), Objective perceptual multimedia video quality measurement in the presence of a full reference.

Recommendation ITU-T J.144 (2004), Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference.

Recommendation ITU-T P.931 (1998), Multimedia communications delay, synchronization and frame rate measurement.

Recommendation ITU-T J.148 (2003), Requirements for an objective perceptual multimedia quality model.

Recommendation ITU-T H.264 (2012), Advanced video coding for generic audiovisual services.

Recommendation ITU-T J.340 (2010), Reference algorithm for computing peak signal to noise ratio (PSNR) of a processed video sequence with constant spatial shifts and a constant delay.

Recommendation ITU-T P.910 (2008), Subjective video quality assessment methods or multimedia applications.

Recommendation ITU-T P.911 (1998), Subjective audiovisual quality assessment methods for multimedia applications.

Recommendation ITU-T J.143 (2000), User requirements for objective perceptual video quality measurements in digital cable television.