International Telecommunication Union

ITU-R

Radiocommunication Sector of ITU

Recommendation ITU-R BT.1885

(03/2011)

Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a reduced bandwidth reference

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, 2011

RECOMMENDATION ITU-R BT.1885

Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a reduced bandwidth reference

(2011)

Scope

This Recommendation describes objective video quality assessment methods for standard definition digital broadcast television which can measure the perceptual video quality of mobile and stationary receiving conditions when the features extracted from the reference video signal are readily available at the measurement point.

The ITU Radiocommunication Assembly,

considering

a) that the ability to measure automatically the impairments of broadcast video signals has been desirable;

b) that the perceptual video quality of mobile reception may dynamically change depending on reception conditions;

c) that objective measurement of perceived video quality may usefully complement subjective assessment methods;

d) that three video quality measurement techniques for standard definition digital broadcast television in the presence of a reduced reference have been proposed to ITU-R and they have been found to provide equivalent, well-aligned results;

e) that objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference have been specified in Recommendation ITU-R BT.1683,

recommends

1 that any of the objective video quality models given in Annex 1 should be used for objective measurement of perceived video quality for standard definition digital broadcast television in the presence of a reduced bandwidth reference.

1 Introduction

This RRNR-TV test addresses images as defined in Recommendation ITU-R BT.601-6 and two types of models: reduced reference (RR), and no reference (NR). RR models have limited bandwidth access to the source video, and NR models do not have access to the source video.

The HRCs in each experiment encompassed both coding only artefacts and coding with transmission errors. The coding schemes examined were MPEG-2 and H.264 (MPEG-4 part 10). The MPEG-2 coders were run at a variety of bit rates from 1.0 to 5.5 Mbit/s. The H.264 coders were run at a variety of bit rates ranging from 1.0 to 3.98 Mbit/s. Each experiment included 12 source sequences, of which two were secret source. Each experiment included 34 HRCs, and 156 processed video sequences (PVSs). Of these PVSs, 40 contained transmission errors and 116 contained coding only.

1.1 Application

This Recommendation provides video quality estimations for video classes TV3 to MM5B, as defined in ITU-T Recommendation P.911, Annex B. The applications for the estimation models 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 side-channels are available for features extracted from source video sequences;
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 estimation models described in this Recommendation cannot be used to fully 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 compare different codecs, it is advisable to use a quantitative method (such as that in ITU-T Recommendation J.149) to determine the model's accuracy for that particular context.

The models in this Recommendation were validated by measuring objective video quality that exhibits frame freezes up to 2 s.

The models in this Recommendation were not validated for measuring objective video quality that has a steadily increasing delay (e.g. video which does not discard missing frames after a frame freeze).

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

2 References

The following ITU Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

2.1 Normative references

Recommendation ITU-R BT.500-12 – Methodology for the subjective assessment of the quality of television pictures.

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

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

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

ITU-T Recommendation J.244 (2008) – Full reference and reduced reference calibration methods for video transmission systems with constant misalignment of spatial and temporal domains with constant gain and offset.

2.2 Informative references

ITU-T Recommendation J.149 (1998) – Subjective audiovisual quality assessment methods for multimedia applications.

ITU-T Recommendation J.144 (2001) – Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference.

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

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

ITU-T Recommendation H.261 (1993) – Video codec for audiovisual services at p x 64 kbits.

ITU-T Recommendation H.263 (1996) – Video coding for low bit rate communication.

ITU-T Recommendation H.263 (1998) – Video coding for low bit rate communication (H.263+).

ITU-T Recommendation H.264 (2003) – Advanced video coding for generic audiovisual services.

VQEG – Validation of reduced-reference and no-reference objective models for standard definition television, Phase I, 2009.

3 Definitions

3.1 Terms defined elsewhere:

This Recommendation uses the following terms defined elsewhere:

3.1.1 subjective assessment (picture) (ITU-T Recommendation J.144): optional quoted definition.

3.1.2 objective perceptual measurement (picture) (ITU-T Recommendation J.144): optional quoted definition.

3.1.3 Proponent (ITU-T Recommendation J.144): optional quoted definition.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 Anomalous frame repetition: is defined as an event where the HRC outputs a single frame repeatedly in response to an unusual or out of the ordinary event. Anomalous frame repetition includes but is not limited to the following types of events: an error in the transmission channel, a change in the delay through the transmission channel, limited computer resources impacting the decoder's performance, and limited computer resources impacting the display of the video signal.

3.2.2 Constant frame skipping: is defined as an event where the HRC outputs frames with updated content at an effective frame rate that is fixed and less than the source frame rate.

3.2.3 Effective frame rate: is defined as the number of unique frames (i.e. total frames – repeated frames) per second.

3.2.4 Frame rate: is defined as the number of unique frames (i.e. total frames – repeated frames) per second.

3.2.5 Intended frame rate: is defined as the number of video frames per second physically stored for some representation of a video sequence. The frame rate shall be constant. Two examples of constant intended frame rates are a BetacamSP® tape containing 25 fps and a VQEG FR-TV Phase I compliant 625-line YUV file containing 25 fps; these both have an intended frame rate of 25 fps.

3.2.6 Live network conditions: are defined as errors imposed upon the digital video bit stream as a result of live network conditions.

3.2.7 Pausing with skipping: is defined as events where the video pauses for some period of time and then restarts with some loss of video information. In pausing with skipping, the temporal delay through the system will vary about an average system delay, sometimes increasing and sometimes decreasing. One example of pausing with skipping is a pair of IP Videophones, where heavy network traffic causes the IP Videophone display to freeze briefly; when the IP Videophone display continues, some content has been lost. Constant frame skipping and variable frame skipping are subsets of pausing with skipping. A processed video sequence containing pausing with skipping will be approximately the same duration as the associated original video sequence.

3.2.8 Pausing without skipping: is defined as any event where the video pauses for some period of time and then restarts without losing any video information. Hence, the temporal delay through the system must increase.

3.2.9 Refresh rate: is defined as the rate at which the display is updated.

3.2.10 Simulated transmission errors: are defined as errors imposed upon the digital video bit stream in a highly controlled environment. Examples include simulated packet loss rates and simulated bit errors.

3.2.11 Source frame rate (SFR): is the intended frame rate of the original source video sequences. The source frame rate is constant. For the VQEG RRNR-TV test the SFR was either 25 fps or 30 fps.

3.2.12 Transmission errors: are defined as any error imposed on the video transmission. Example types of errors include simulated transmission errors and live network conditions.

3.2.13 Variable frame skipping: is defined as an event where the HRC outputs frames with updated content at an effective frame rate that changes with time. The temporal delay through the system will increase and decrease with time, varying about an average system delay. A processed video sequence containing variable frame skipping will be approximately the same duration as the associated original video sequence.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ACR	Absolute category rating (see ITU-T Recommendation P.910)
ACR-HR	Absolute category rating with hidden reference (see ITU-T Recommendation P.910)
AVI	Audio video interleave
DMOS	Difference mean opinion score
FR	Full reference
FRTV	Full reference television
HRC	Hypothetical reference circuit

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

5 Conventions

None.

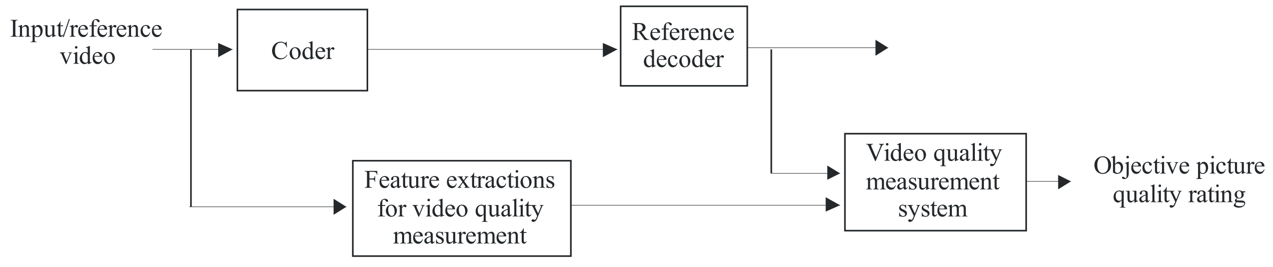
6 Description of the reduced reference measurement method

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

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

FIGURE 1

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


Input/reference video
Coder
Reference decoder
Feature extractions for video quality measurement
Video quality measurement system
Objective picture quality rating

BT.1885-01

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.

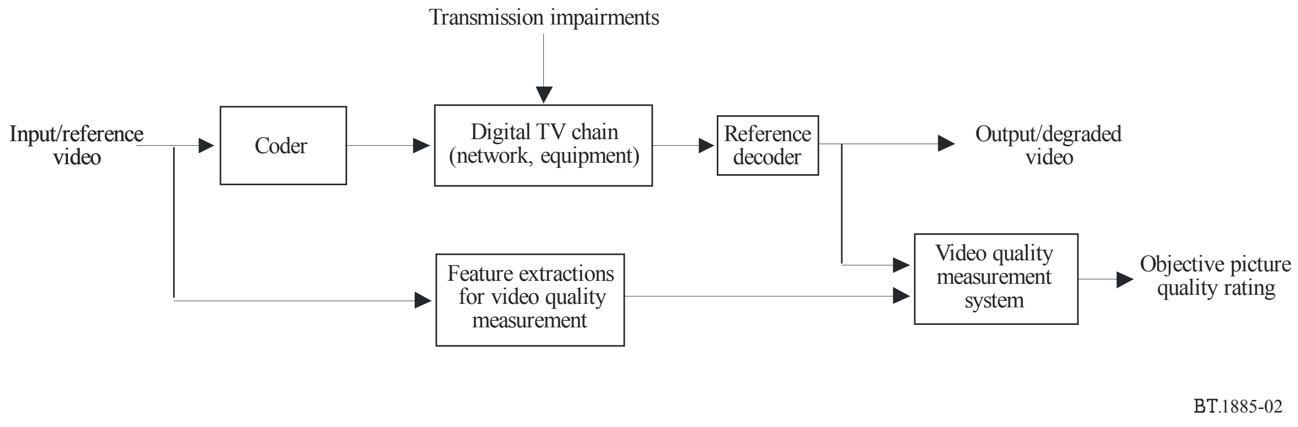
Alignment and gain adjustment is known as normalization. This process is required because most reduced reference methods compare the features extracted from reference pictures and processed pictures on what is effectively a pixel-by-pixel basis. An example would be calculation of peak signal-to-noise ratio (PSNR). Only time-invariant static changes in the video are removed, dynamic changes due to system under test are measured as part of the quality rating calculation. ITU-T Recommendations J.244 and ITU-T J.144 provide standard methods for reporting values needed for normalizing video prior to objective quality assessment. The video quality metrics described in Annex of this Recommendation include associated normalization methods. Alternate normalization methods can be used for the video quality metrics of Annex provided they deliver the required normalization accuracy.

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 reduced reference method to test a transmission chain.

FIGURE 2

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


Transmission impairments
Input/reference video
Coder
Digital TV chain (network, equipment)
Reference decoder
Output/degraded video
Feature extractions for video quality measurement
Video quality measurement system
Objective picture quality rating
BT.1885-02

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 Recommendation P.910.

7 Findings of the Video Quality Experts Group

Studies of perceptual video quality measurements are conducted in an informal group, called Video Quality Experts Group (VQEG), which reports to ITU-T Study Groups 9 and 12 and Radiocommunications Study Group 6. The recently completed RRNR-TV test of VQEG assessed the performance of proposed reduced reference perceptual video quality measurement algorithms for ITU-R 601-6 image formats. formats.

Based on the present evidence, six RR methods (Model_A 15k, Model_A 80k, Model_A 256k, Model_C 80k, Model_B 80k (525-line only), Model_B 256k (525-line only) can be recommended by ITU-T at this time.

The technical descriptions of these models can be found in Annexes A through C respectively. Note that the ordering of Annexes is purely arbitrary and provides no indication of quality prediction performance.

Tables 1 and 2 show significance tests in the VQEG RRNR-TV test. For the 525-line format, four models (Model_A 15k, Model_A 80k, Model_A 256k, Model_C 80k) are statistically better than PSNR and two models (Model_B 80k, Model_B 256k) are statistically equivalent to PSNR. It should be noted that PSNR was computed by NTIA using an exhaustive search of calibration limits. For the 625-line format, four models (Model_A 15k, Model_A 80k, Model_A 256k, Model_C 80k) are statistically equivalent and are statistically better than PSNR.

TABLE 1

Significance test for the 525-line format

525-line format	Compare best	Compare PSNR	Correlation
Model_A 15k	1	1	0.906
Model_A 80k	1	1	0.903
Model_A 256k	1	1	0.903
Model_C 80k	1	1	0.882
Model_B 80k	0	1	0.795
Model_B 256k	0	1	0.803
PSNR_NTIA	0	1	0.826

NOTE 1 – “1” in “Compare best” indicates that this model is statistically equivalent to the top performing model. “0” indicates that this model is not statistically equivalent to the top performing model. “1” in “Compare PSNR” indicates that this model is statistically equivalent to the top performing model. “0” indicates that this model is not statistically equivalent to the top performing model.

TABLE 2

Significance test for the 625 format

625-line format	Compare best	Compare PSNR	Correlation
Model_A 15k	1	1	0.894
Model_A 80k	1	1	0.899
Model_A 256k	1	1	0.898
Model_C 80k	1	1	0.866
PSNR_NTIA	0	1	0.857

NOTE 1 – “1” in “Compare best” indicates that this model is statistically equivalent to the top performing model. “0” indicates that this model is not statistically equivalent to the top performing model. “1” in “Compare PSNR” indicates that this model is statistically equivalent to the top performing model. “0” indicates that this model is not statistically equivalent to the top performing model.

Tables 3 and 4 provide informative details on the model’s performances in the VQEG RRNR-TV test.

TABLE 3

Informative description on the model's performances in the VQEG RRNR-TV test (525-line format)

525-line format	Correlation	RMSE	OR
Model_A 15k	0.906	0.418	0.385
Model_A 80k	0.903	0.423	0.378
Model_A 256k	0.903	0.424	0.378
Model_B 80k	0.795	0.598	0.667
Model_B 256k	0.803	0.587	0.647
Model_C 80k	0.882	0.465	0.513
PSNR_NTIA	0.826	0.556	0.571

TABLE 4

Informative description on the model's performances in the VQEG RRNR-TV test (625-line format)

625-line format	Correlation	RMSE	OR
Model_A 15k	0.894	0.524	0.468
Model_A 80k	0.899	0.513	0.462
Model_A 256k	0.898	0.516	0.468
Model_C_80k	0.866	0.585	0.583
PSNR_NTIA	0.857	0.605	0.564

Annex A

Model A: Yonsei University reduced reference method

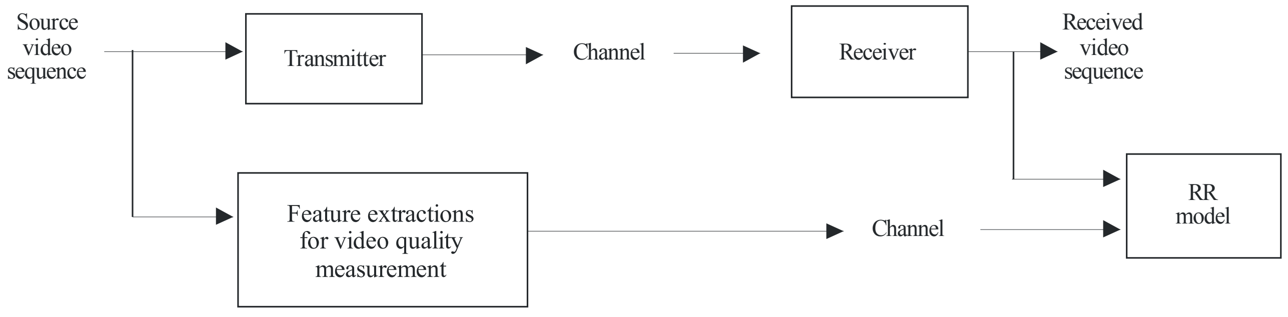
1 Introduction

Although PSNR has been widely used as an objective video quality measure, it is also reported that it does not well represent perceptual video quality. By analysing how humans perceive video quality, it is observed that the human visual system is sensitive to degradation around the edges. In other words, when the edge pixels of a video are blurred, evaluators tend to give low scores to the video even though the PSNR is high. Based on this observation, the reduced reference models which mainly measure edge degradations have been developed.

Figure 3 illustrates how a reduced-reference model works. Features which will be used to measure video quality at a monitoring point are extracted from the source video sequence and transmitted. The Table 5 shows the side-channel bandwidths for the features, which have been tested in the VQEG RRNR-TV test.

FIGURE 3

Block diagram of reduced reference model



BT.1885-03

TABLE 5

Side-channel bandwidths

Video format	Tested bandwidths
525 format	15 kbit/s, 80 kbit/s, 256 kbit/s
625 format	15 kbit/s, 80 kbit/s, 256 kbit/s

2 The EPSNR reduced-reference models

2.1 Edge PSNR

The RR models mainly measure on edge degradations. In the models, an edge detection algorithm is first applied to the source video sequence to locate the edge pixels. Then, the degradation of those edge pixels is measured by computing the mean squared error. From this mean squared error, the edge PSNR (EPSNR) is computed.

One can use any edge detection algorithm, though there may be minor differences in the results. For example, one can use any gradient operator to locate edge pixels. A number of gradient operators have been proposed. In many edge detection algorithms, the horizontal gradient image $g_{horizontal}(m,n)$ and the vertical gradient image $g_{vertical}(m,n)$ are first computed using gradient operators. Then, the magnitude gradient image $g(m,n)$ may be computed as follows:

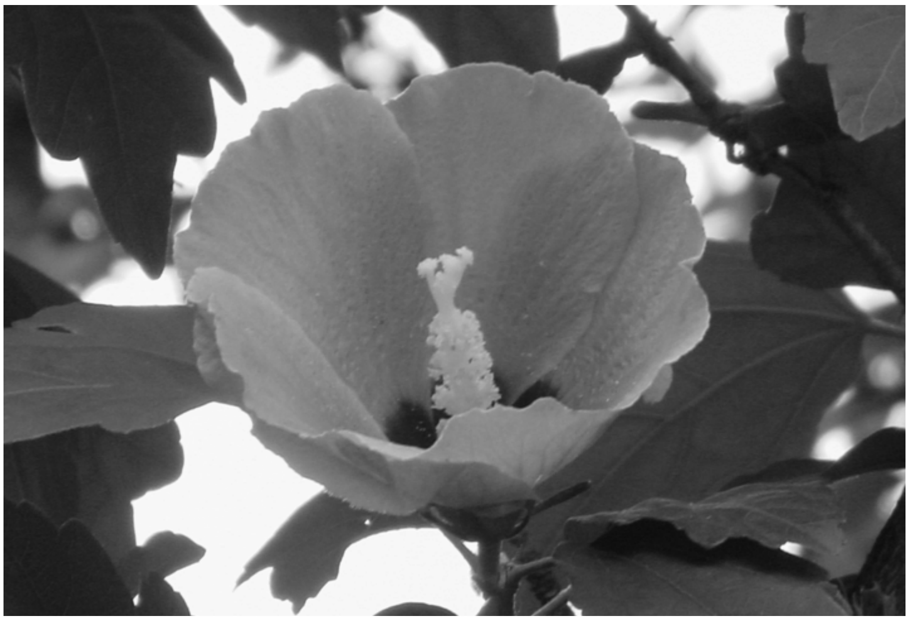
$$g(m,n) = \left|g_{horizontal}(m,n)\right| + \left|g_{vertical}(m,n)\right|$$

Finally, a thresholding operation is applied to the magnitude gradient image $g(m,n)$ to find edge pixels. In other words, pixels whose magnitude gradients exceed a threshold value are considered as edge pixels.

Figures 4 to Fig. 8 illustrate the procedure. Figure 4 shows a source image. Figure 5 shows a horizontal gradient image $g_{horizontal}(m,n)$, which is obtained by applying a horizontal gradient operator to the source image of Fig. 4. Figure 6 shows a vertical gradient image $g_{vertical}(m,n)$, which is obtained by applying a vertical gradient operator to the source image of Fig. 4. Figure 7 shows the magnitude gradient image (edge image) and Fig. 8 shows the binary edge image (mask image) obtained by applying thresholding to the magnitude gradient image of Fig. 7.

FIGURE 4

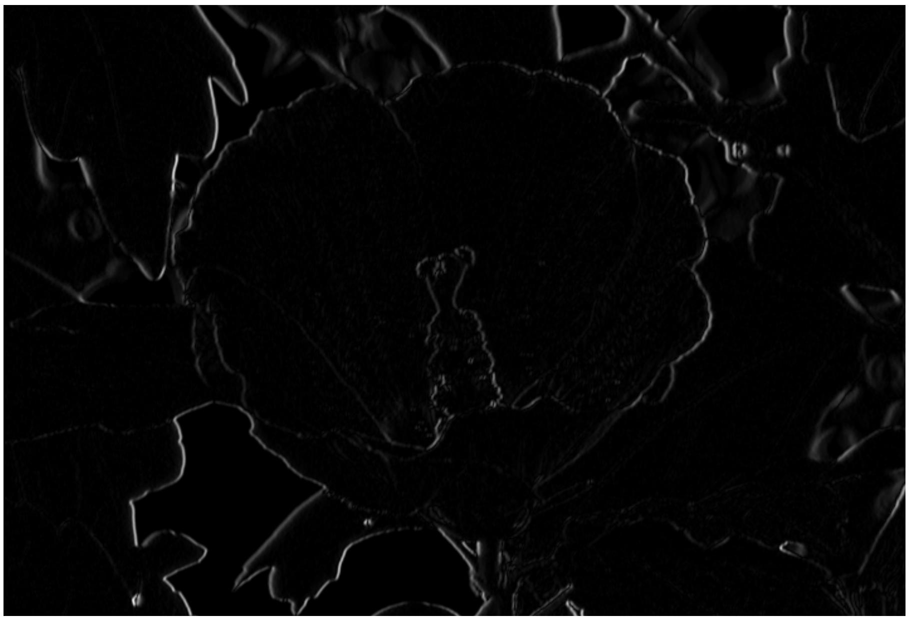
A source image (original image)



BT.1885-04

FIGURE 5

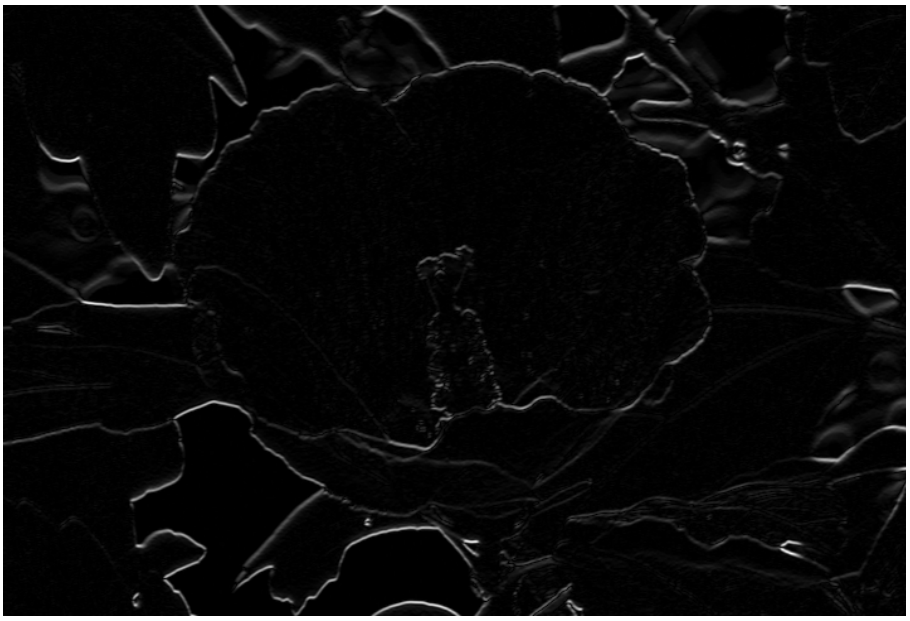
A horizontal gradient image, which is obtained by applying a horizontal gradient operator to the source image of Fig. 4



BT.1885-05

FIGURE 6

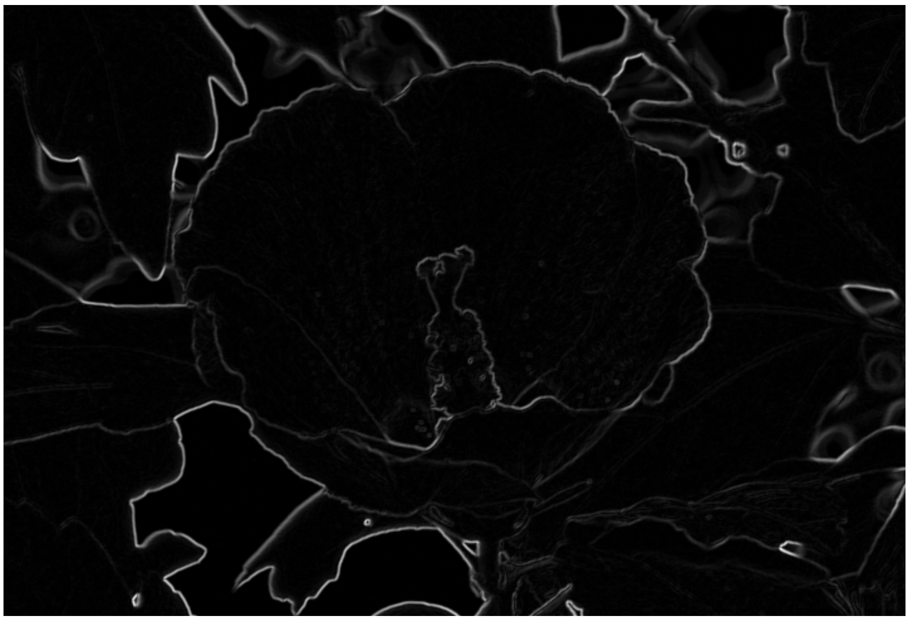
A vertical gradient image, which is obtained by applying a vertical gradient operator to the source image of Fig. 4



BT.1885-06

FIGURE 7

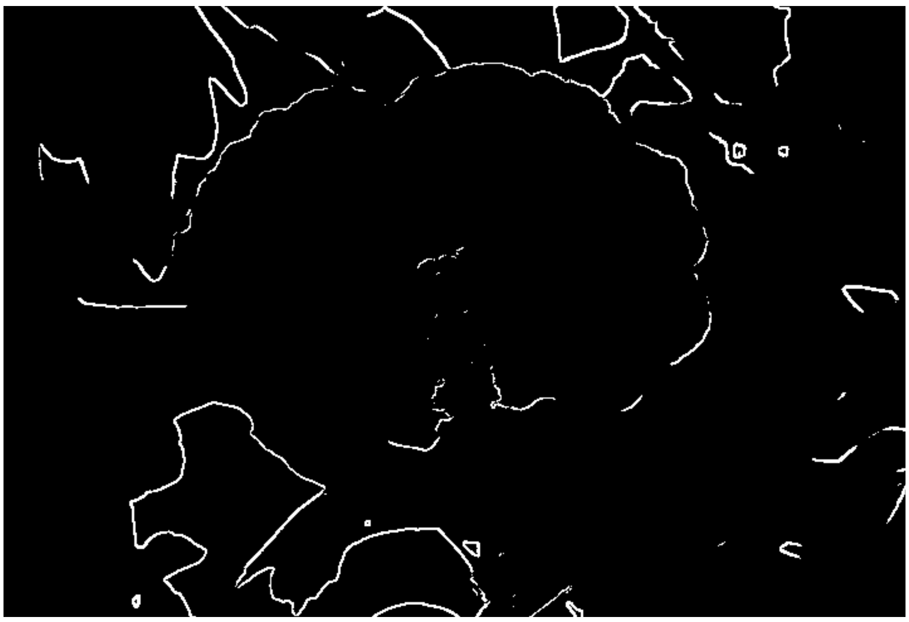
A magnitude gradient image



BT.1885-07

FIGURE 8

A binary edge image (mask image) obtained by applying thresholding to the magnitude gradient image of Fig. 7



BT.1885-08

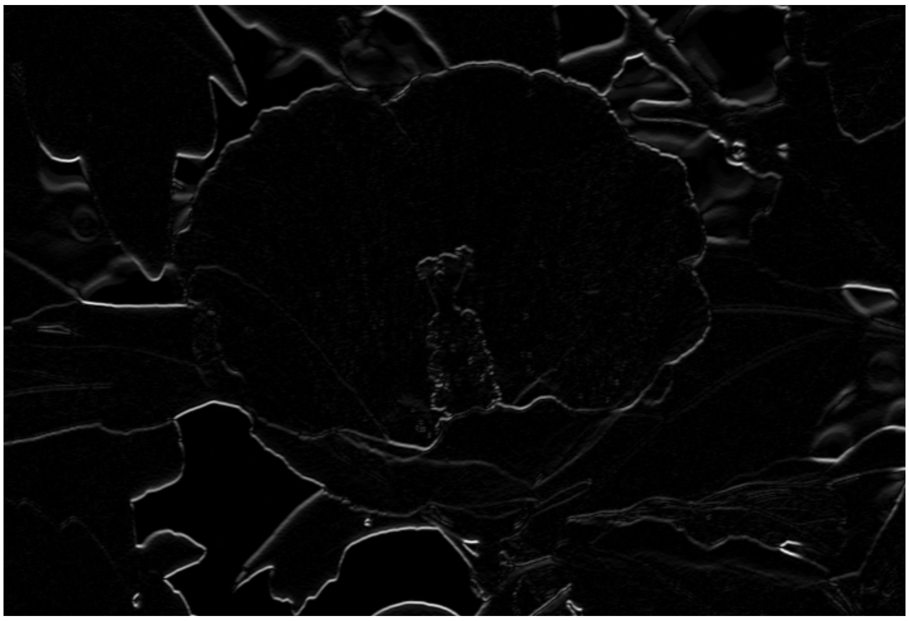
Alternatively, one may use a modified procedure to find edge pixels. For instance, one may first apply a vertical gradient operator to the source image, producing a vertical gradient image. Then, a horizontal gradient operator is applied to the vertical gradient image, producing a modified successive gradient image (horizontal and vertical gradient image). Finally, a thresholding operation may be applied to the modified successive gradient image to find edge pixels. In other words, pixels of the modified successive gradient image, which exceed a threshold value, are considered as edge pixels. Figures 9 to Fig. 12 illustrate the modified procedure. Figure 9 shows a vertical gradient image $g_{vertical}$ (m,n), which is obtained by applying a vertical gradient operator to the source image of Fig. 4. Figure 10 shows a modified successive gradient image (horizontal and vertical gradient image), which is obtained by applying a horizontal gradient operator to the vertical gradient image of Fig. 9. Figure 11 shows the binary edge image (mask image) obtained by applying thresholding to the modified successive gradient image of Fig. 10.

It is noted that both methods can be understood as an edge detection algorithm. One may choose any edge detection algorithm depending on the nature of videos and compression algorithms. However, some methods may outperform other methods.

Thus, in the model, an edge detection operator is first applied, producing edge images (see Figs 7 and 10). Then, a mask image (binary edge image) is produced by applying thresholding to the edge image (see Figs 8 and 11). In other words, pixels of the edge image whose value is smaller than threshold t_e are set to zero and pixels whose value is equal to or larger than the threshold are set to a nonzero value. Figures 8 and 11 show some mask images. Since a video can be viewed as a sequence of frames or fields, the above-stated procedure can be applied to each frame or field of videos. Since the model can be used for field-based videos or frame-based videos, the terminology “image” will be used to indicate a field or frame.

FIGURE 9

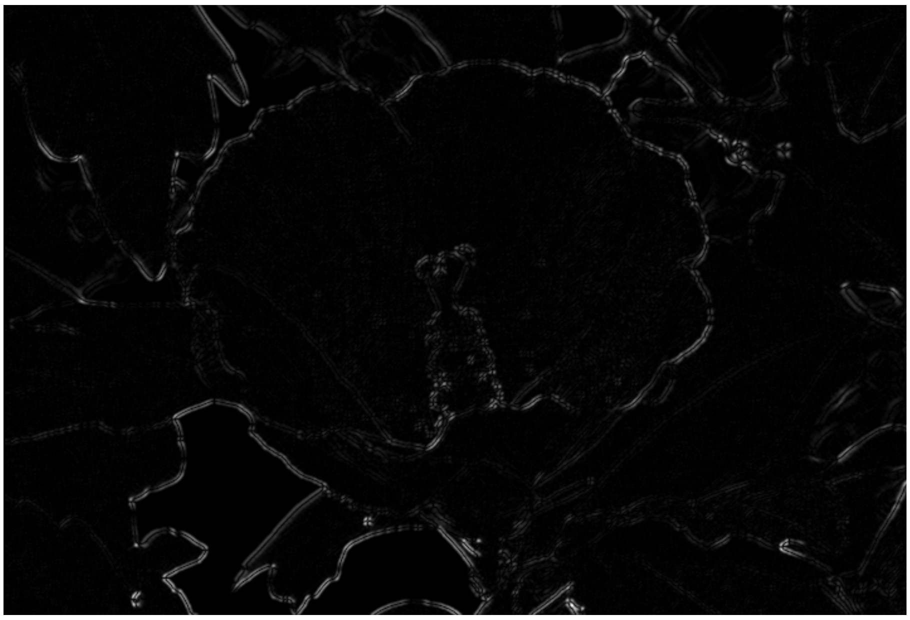
A vertical gradient image, which is obtained by applying a vertical gradient operator to the source image of Fig. 4



BT.1885-09

FIGURE 10

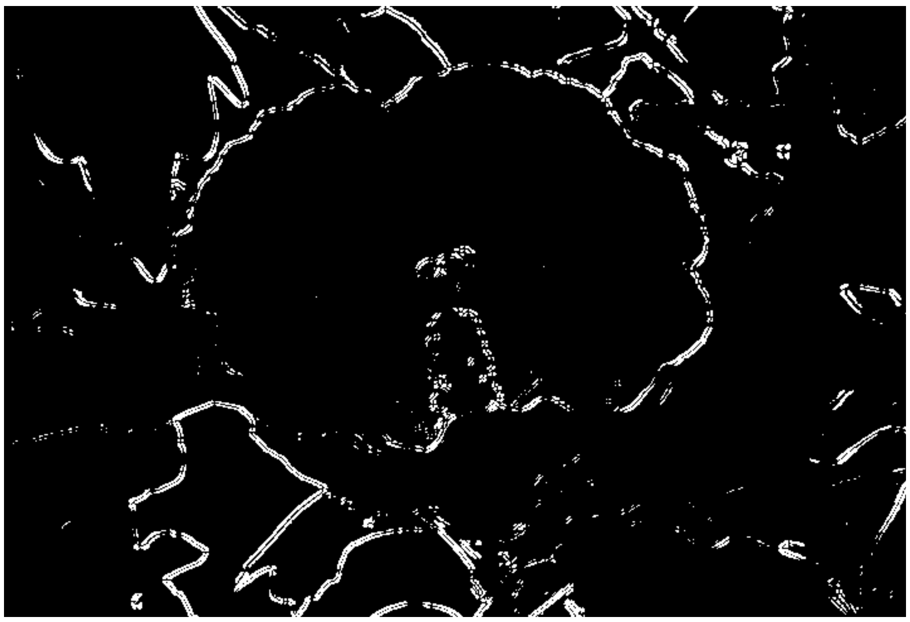
A modified successive gradient image (horizontal and vertical gradient image), which is obtained by applying a horizontal gradient operator to the vertical gradient image of Fig. 9



BT.1885-10

FIGURE 11

A binary edge image (mask image) obtained by applying thresholding to the modified successive gradient image of Fig. 10



BT.1885-11

2.2 Selecting features from source video sequences

Since the model is a RR model, a set of features need to be extracted from each image of a source video sequence. In the EPSNR RR model, a certain number of edge pixels are selected from each image. Then, the locations and pixel values are encoded and transmitted. However, for some video sequences, the number of edge pixels can be very small when a fixed threshold value is used. In the worst scenario, it can be zero (blank images or very low frequency images). In order to address this problem, if the number of edge pixels of an image is smaller than a given value, the user may reduce threshold value until the number of edge pixels is larger than a given value. Alternatively, one can select edge pixels which correspond to the largest values of the horizontal and vertical gradient image. When there are no edge pixels (e.g. blank images) in a frame, one can randomly select the required number of pixels or skip the frame. For instance, if 10 edge pixels are to be selected from each frame, one can sort the pixels of the horizontal and vertical gradient image according to their values and select the largest 10 values. However, this procedure may produce multiple edge pixels at the identical locations. To address this problem, one can first select several times of the desired number of pixels of the horizontal and vertical gradient image and then randomly choose the desired number of edge pixels among the selected pixels of the horizontal and vertical gradient image. In the models tested in the VQEG RRNR-TV test, the desired number of edge pixels is randomly selected among a large pool of edge pixels. The pool of edge pixels is obtained by applying a thresholding operation to the gradient image.

In the EPSNR RR models, the locations and edge pixel values are encoded after a Gaussian low pass filter is applied to the selected pixel locations. Although the Gaussian LPF (5×3) was used in the VQEG RRNR-TV test, different low pass filters may be used depending on video formats. It is noted that during encoding process, cropping may be applied. In order to avoid selecting edge pixels in the cropped areas, the model selects edge pixels in the middle area (see Fig. 12). Table 6 shows the sizes after cropping. Table 6 also shows the number of bits required to encode the location and pixel value of an edge pixel.

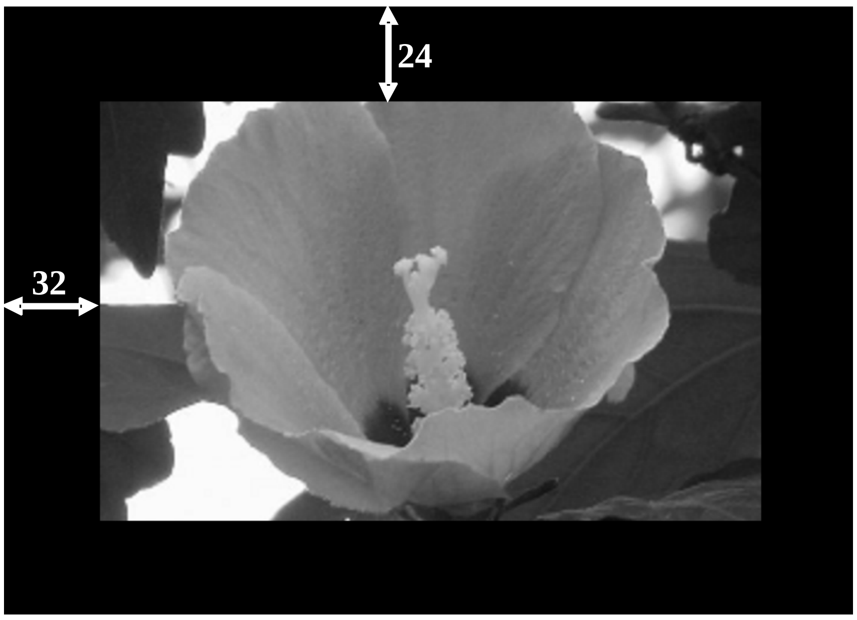
TABLE 6

Bits requirement per edge pixel

Video format	**Size**	**Size after cropping**	**Bits for location**	**Bits for pixel value**	**Total bits per pixel**
525	720 × 486	656 × 438	19	8	27
625	720 × 576	656 × 528	19	8	27

FIGURE 12

An example of cropping and the middle area



BT.1885-12

The model selects edge pixels from each frame in accordance with the allowed bandwidth (see Table 5). Table 7 shows the number of edge pixels per frame which can be transmitted for the tested bandwidths.

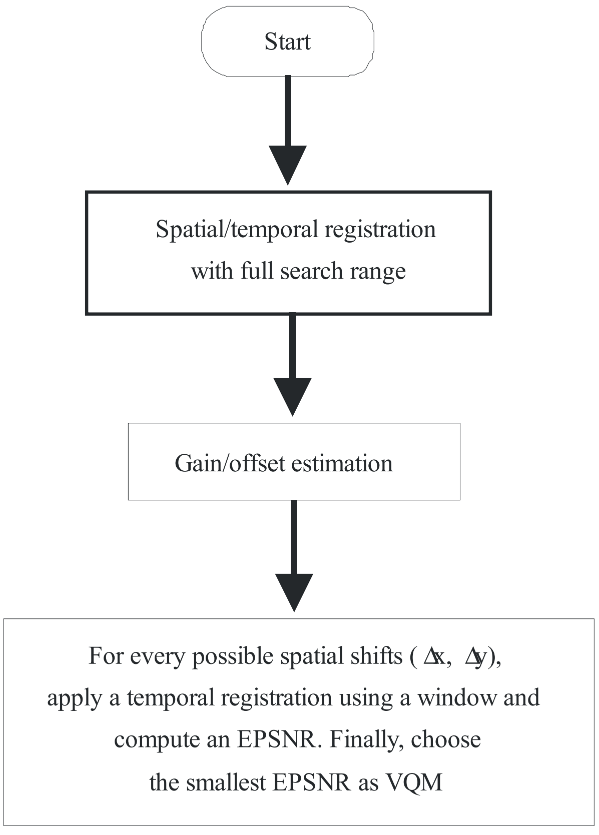
TABLE 7

Number of edge pixels per frame

Video format	**15 kbit/s**	**80 kbit/s**	**256 kbit/s**
525	16	74	238
625	20	92	286

FIGURE 13

Flowchart of the model


Start
Spatial/temporal registration with full search range
Gain/offset estimation
For every possible spatial shifts (Δx, Δy), apply a temporal registration using a window and compute an EPSNR. Finally, choose the smallest EPSNR as VQM

BT.1885-13

2.3 Spatial/temporal registration and gain/offset adjustment

Before computing the difference between the edge pixels of the source video sequence and those of the processed video sequence which is the received video sequence at the receiver, the model first applies a spatial/temporal registration and gain/offset adjustment. The calibration method (Annex B) of ITU-T Recommendation J.244 was used. To transmit the gain and offset features of ITU-T Recommendation J.244 (Annex B), 30% of available bandwidths was used in the VQEG RRNR-TV test. Since the video sequence is interlaced, the calibration method is applied three times: the even fields, odd fields and combined frames. If the difference between the even field error (PSNR) and the odd field error is greater than a threshold, the registration results (x-shift, y-shift) with the smaller PSNR were used. Otherwise, the registration results with the combined frames were used. In the VQEG RRNR-TV test, the threshold was set to 2 dB.

At the monitoring point, the processed video sequence should be aligned with the edge pixels extracted from the source video sequence. However, if the side-channel bandwidth is small, only a few edge pixels of the source video sequence are available (see Fig. 14). Consequently, the temporal registration can be inaccurate if the temporal registration is performed using a single frame (see Fig. 15). To address this problem, the model uses a window for temporal registration. Instead of using a single frame of the processed video sequence, the model builds a window which consists of a number of adjacent frames to find the optimal temporal shift. Figure 16 illustrates the procedure. The mean squared error within the window is computed as follows:

$$MSE_{window} = \frac{1}{N_{win}} \sum (E_{SRC}(i) - E_{PVS}(i))^2$$

where:

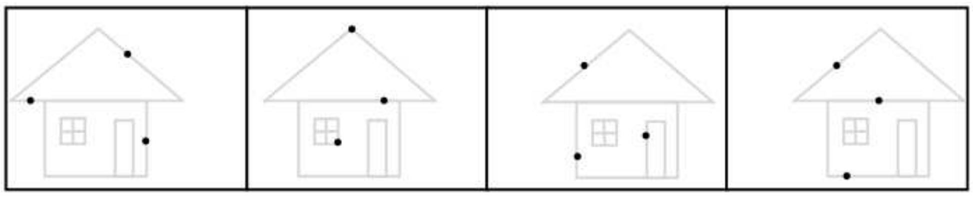
MSE_{window}:	the window mean squared error
$E_{SRC}(i)$:	an edge pixel within the window which has a corresponding pixel in the processed video sequence
$E_{PVS}(i)$:	a pixel of the processed video sequence corresponding to the edge pixel
N_{win}:	the total number of edge pixels used to compute MSE_{window}.

This window mean squared error is used as the difference between a frame of the processed video sequence and the corresponding frame of the source video sequence.

The window size can be determined by considering the nature of the processed video sequence. For a typical application, a window corresponding to two seconds is recommended. Alternatively, various sizes of windows can be applied and the best one which provides the smallest mean squared error can be used. Furthermore, different window centres can be used to consider frame skipping due to transmission errors (see Fig. 20).

FIGURE 14

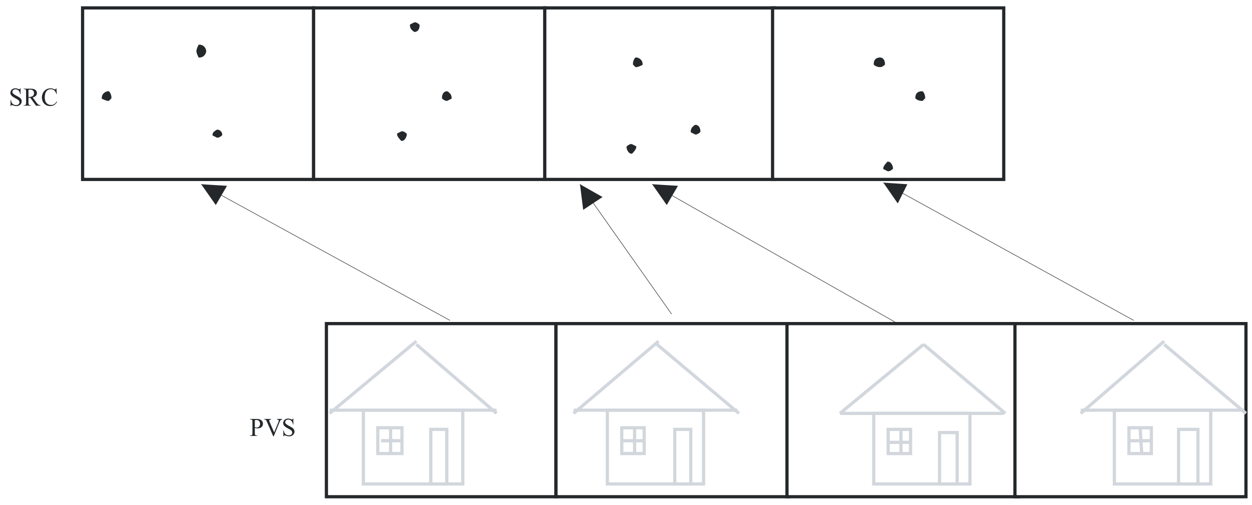
Edge pixel selection of the source video sequence



BT.1885-14

FIGURE 15

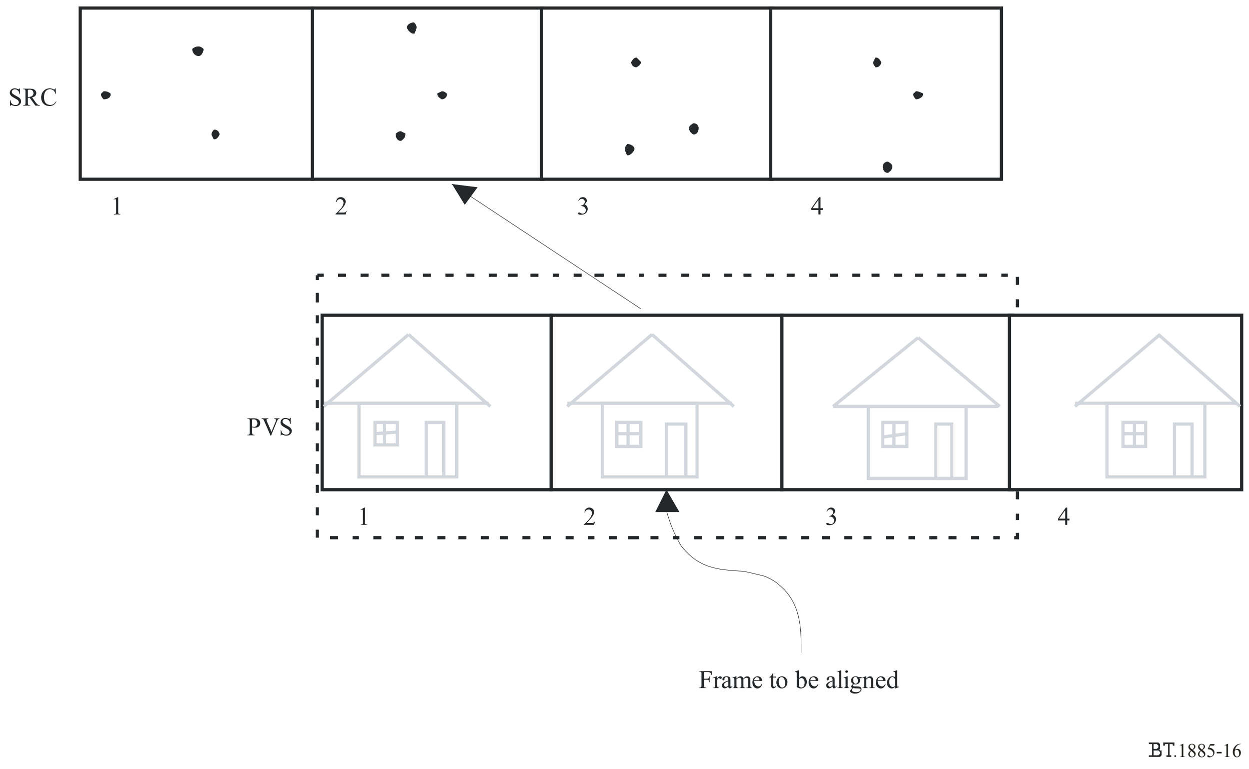
Aligning the processed video sequence to the edge pixels of the source video sequence



BT.1885-15

FIGURE 16

Aligning the processed video sequence to the edge pixels using a window



When the source video sequence is encoded at high compression ratios, the encoder may reduce the number of frames per second and the processed video sequence has repeated frames (see Fig. 17). In Fig. 17, the processed video sequence does not have frames corresponding some frames of the source video sequence (2, 4, 6, 8th frames). In this case, the model does not use repeated frames in computing the mean squared error. In other words, the model performs temporal registration using the first frame (valid frame) of each repeated block. Thus, in Fig. 18, only three frames (3, 5, 7th frames) within the window are used for temporal registration.

FIGURE 17

Example of repeated frames

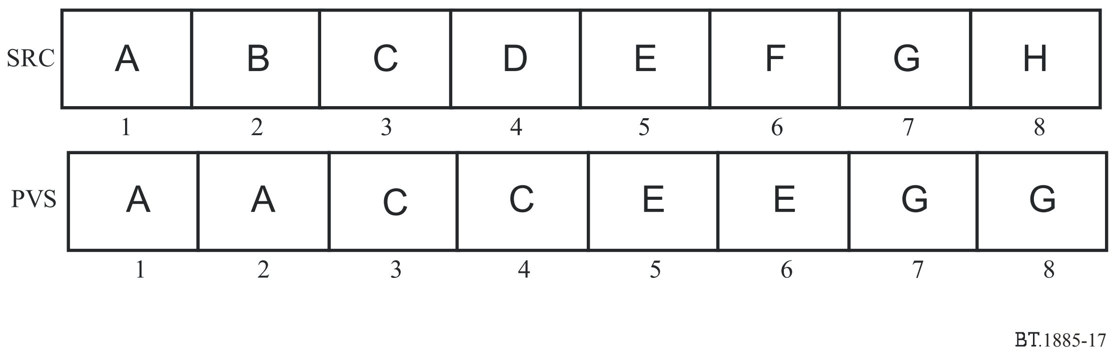
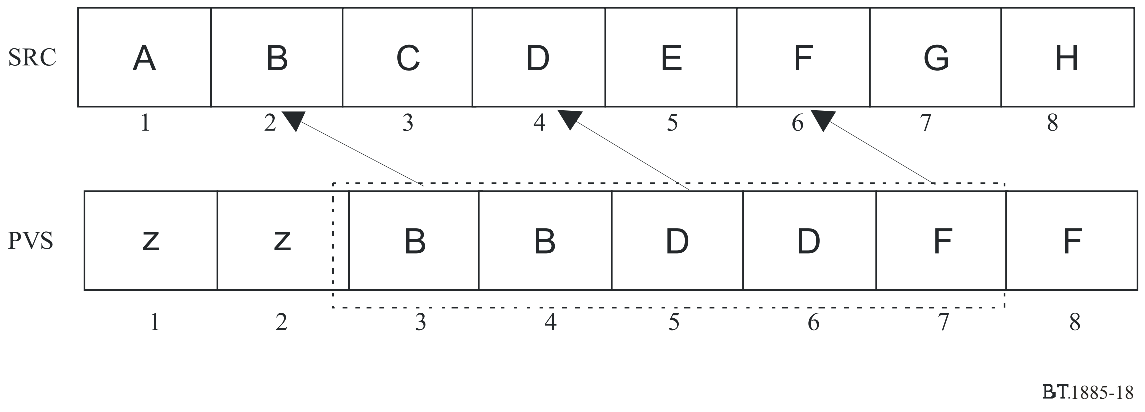


FIGURE 18

Handing repeated frames



It is possible to have a processed video sequence with irregular frame repetition, which may cause the temporal registration method using a window to produce inaccurate results. To address this problem, it is possible to locally adjust each frame of the window within a given value (e.g. ±1) as shown in Fig. 21 after the temporal registration using a window. Then, the local adjustment which provides the minimum MSE is used to compute the EPSNR.

FIGURE 19

Windows of various sizes

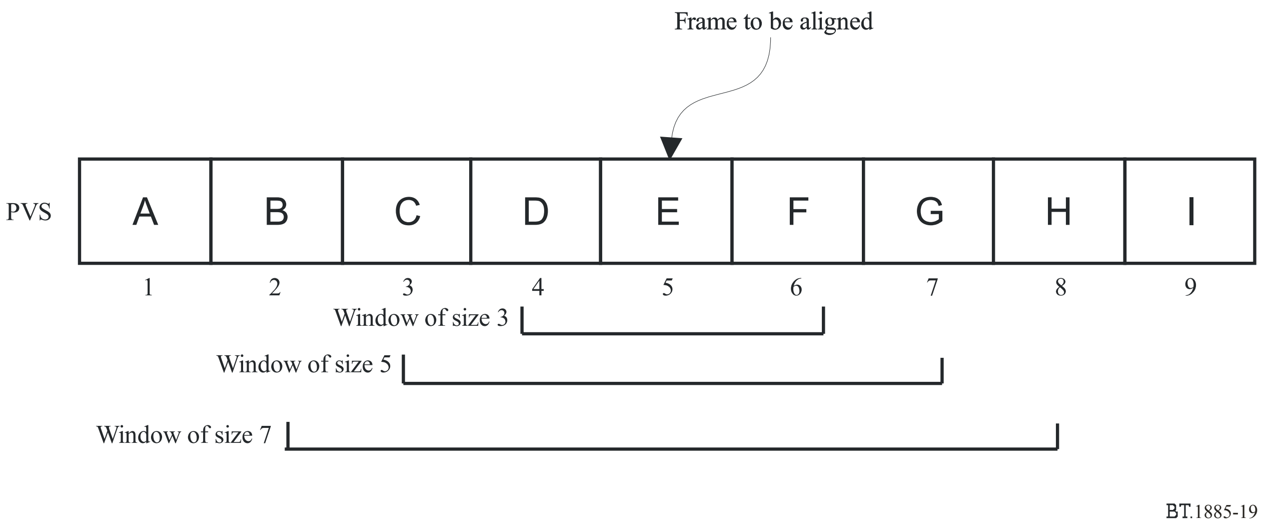
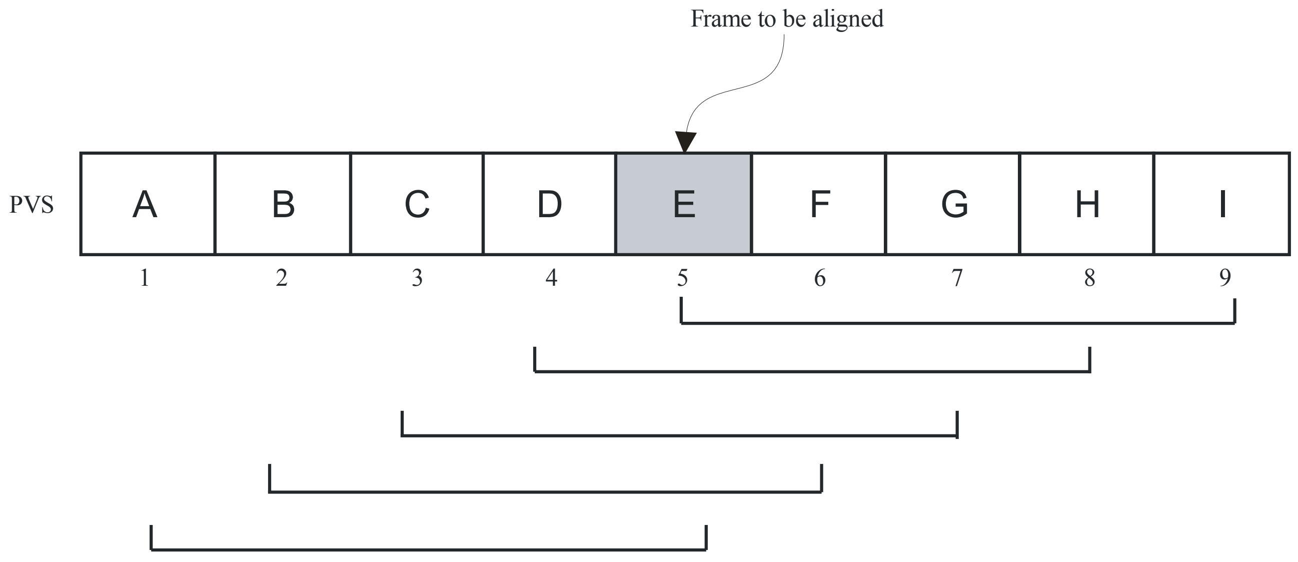


FIGURE 20

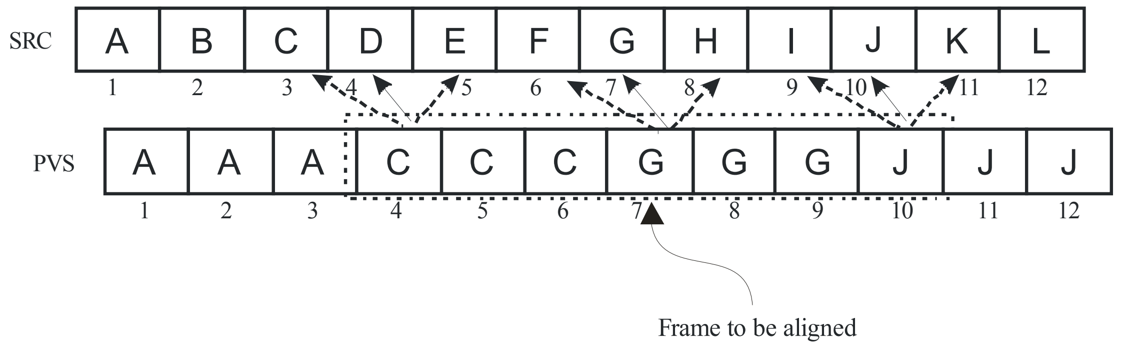
Window centres



BT.1885-20

FIGURE 21

Local adjustment for temporal registration using a window



BT.1885-21

2.4 Computing EPSNR and post-processing

After temporal registration is performed, the average of the differences between the edge pixels of the source video sequence and the corresponding pixels of the processed video sequence is computed, which can be understood as the edge mean squared error of the processed video sequence (MSE_{edge}). Finally, the EPSNR (edge PSNR) is computed as follows:

$$EPSNR = 10\log_{10}\left(\frac{P^2}{MSE_{edge}}\right)$$

where p is the peak value of the image.

1. Freezed frames

There can be frame repeating due to reduced frame rates and frame freezing due to transmission error, which will degrade perceptual video quality. In order to address this effect, the model applies the following adjustment before computing the EPSNR:

$$MSE_{freezed\_frame\_considered} = MSE_{edge} \times \frac{K \times N_{total\_frame}}{N_{total\_frame} - N_{total\_freezed\_frame}}$$

where:

$MSE_{freezed\_frame\_considered}$:	the mean squared error which takes into account repeated and freezed frames
N_{total_frame} :	the total number of frames, $N_{total\_freezed\_frame}$
K:	a constant.

In the model tested in the VQEG RRNR-TV test, K was set to 1.

2. High frequency and fast motion

If the video sequence contains a large amount of high frequencies and fast motions, the perceptual quality tends to increase for the same MSE. To consider this effect, the normalized frame difference (NFD) and the normalized high frequency energy (NHFE) are defined as follows:

$$NFD = \frac{FD}{average\ energy\ per\ pixel}$$

where $FD = \frac{1}{N_F}\sum_i \sum_{k=1}^{height} \sum_{j=1}^{width} (Frame_i[j,k] - Frame_{i-1}[j,k])^2$ and N_F is the number of frames used in the summation. It is noted that the largest three frame differences are excluded in computing FD in order to exclude scene changes in computing the average frame difference, assuming 8-second video sequences. The normalized high frequency energy (NHFE) is computed by computing the average high frequency energies (see Fig. 22) after the 2D Fourier transform is applied:

$$NHFE = \frac{average\ high\ frequency\ energies}{average\ energy\ per\ pixel}$$

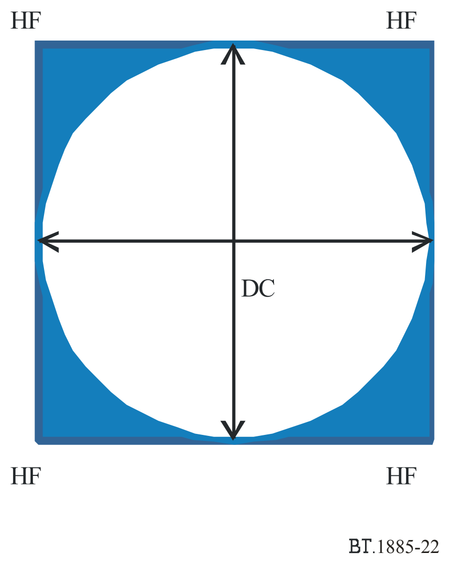
Finally, the following equations are used:

```
IF(SNFD > 0.35 && SNHFE > 2.5)  {
                IF(EPSNR < 20) EPSNR = EPSNR+3
                ELSE IF(EPSNR < 35) EPSNR = EPSNR+5
}
ELSE IF((SNFD > 0.2 && SNHFE > 1.5) || (SNFD>0.27) && SNHFE > 1.3))        {
        I       F(28 < EPSNR < 40) EPSNR = EPSNR + 3
                IF(EPSNR > 40) EPSNR = 40
}
```

where SNFD is the source NFD and SNHFE is the source NHFE. It is noted that SNFD and SNHFE are computed from SRC and transmitted as a part of feature data (1 byte for each).

FIGURE 22

Computing the normalized high frequency energy (NHFE). The high-frequency energies are computed from the shaded area



3. Blurring

To consider blurring effects, the following equations are used:

```
IF (NHFE/SNHFE < 0.5)
              IF(EPSNR>26)      EPSNR = 26
ELSE IF (NHFE/SNHFE < 0.6)
              IF(EPSNR>32)      EPSNR = 32
ELSE IF (NHFE/SNHFE < 0.7)
              IF(EPSNR>36)      EPSNR = 36
ELSE IF (NHFE/SNHFE > 1.2)
              IF(EPSNR>23)      EPSNR = 23
ELSE IF (NHFE/SNHFE > 1.1)
              IF (EPSNR>25)     EPSNR = 25
```

where NHFE is the PVS NHFE.

4. Blocking

To consider blocking effects, average column differences are computed. Assuming 8 modulo, the blocking score for the *i*-th frame is computed as follows:

$$Blk[i] = \frac{largest\ column\ difference}{second\ largest\ column\ difference}$$

The final blocking score (*Blocking*) is computed by averaging the frame blocking scores:

$$Blocking = \frac{1}{number\ of\ \ frames}\sum_i Blk[i]$$

Finally, the following equations are used:

```
IF(BLOCKING > 1.4)      {
                 IF (20≤EPSNR<25) EPSNR = EPSNR-1.086094*BLOCKING-0.601316
                 ELSE IF (EPSNR<30) EPSNR = EPSNR-0.577891*BLOCKING-3.158586
                 ELSE IF (EPSNR<35) EPSNR = EPSNR-0.223573*BLOCKING-3.125441
}
```

5. Maximum freezed frames

Transmission errors may cause long freezed frames. To consider long freezed frames, the following equations are used:

```
IF(MAX_FREEZE > 22 AND EPSNR>28) EPSNR = 28
ELSE IF(MAX_FREEZE > 10 AND EPSNR>34) EPSNR = 34
```

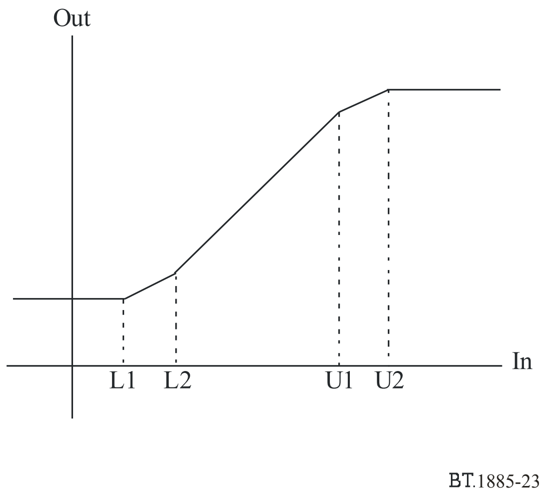
where MAX_FREEZE is the largest duration of freezed frames. It is noted that if the video sequence is not 8 s, different thresholds should be used.

6. Piecewise linear fitting

When the EPSNR exceeds a certain value, the perceptual quality becomes saturated. In this case, it is possible to set the upper bound of the EPSNR. Furthermore, when a linear relationship between the EPSNR and DMOS(difference mean opinion score) is desirable, one can apply a piecewise linear function as illustrated in Fig. 23. In the model tested in the VQEG RRNR-TV test, the upper bound is set to 48 and the lower bound was set to 15.

FIGURE 23

Piecewise linear function for linear relationship between the EPSNR and DMOS



The EPSNR reduced reference models for objective measurement of video quality are based on edge degradation. The models can be implemented in real time with moderate use of computing power. The models are well suited to applications which require real-time video quality monitoring where side-channels are available.

Annex B

Model B: NEC reduced reference method

This Annex provides a full functional description of the RR model. In the RR model, the activity values instead of the pixel-values for individual given-size pixel blocks are transmitted to the client side. Video quality is estimated on the basis of the activity-difference between the source reference channel (SRC) and the processed video sequence (PVS). Psychovisual weightings with respect to the activity-difference are to improve estimation accuracy.

Since this model does not need computationally demanding spatial and gain-and-offset registrations. Moreover, it can be implemented by a 30-line programme and a 250-line programme on the server and the client sides, respectively. Therefore, it is suitable for real-time video-quality monitoring in broadcasting services that most benefits from low-complexity and easy implementation.

1 Summary

In the RR model, the activity values instead of the pixel-values for individual given-size pixel blocks are transmitted to the client side. Video quality is estimated on the basis of the activity-difference between the SRC and the PVS. Psychovisual weightings with respect to the activity-difference are to improve estimation accuracy.

Since this model does not need computationally demanding spatial and gain-and-offset registrations. Moreover, it can be implemented by a 30-line and a 250-line programmes on the server and the client sides, respectively. Therefore, it is suitable for real-time video-quality monitoring in broadcasting services that most benefits from low complexity and easy implementation.

2 Definitions

Activity – An average of the absolute difference between each luminance value and the average of the luminance values for a given-size block.

Block – An M x N (M-column by N-row) array of pixels.

Frame – One complete television picture.

Gain – A multiplicative scaling factor applied by the hypothetical reference circuit (HRC) to all pixels of an individual image plane (e.g. luminance, chrominance). The gain of the luminance signal is commonly known as the contrast.

Hypothetical Reference Circuit (HRC) – A video system under test such as a codec or digital video transmission system.

Luminance (Y) – The portion of the video signal that predominantly carries the luminance information (i.e. the black and white part of the picture).

National Television Systems Committee (NTSC) – The 525-line analogue colour video composite system [1].

Offset or level offset – An additive factor applied by the hypothetical reference circuit (HRC) to all pixels of an individual image plane (e.g. luminance, chrominance). The offset of the luminance signal is commonly known as brightness.

Peak signal-to-noise ratio (PSNR) – A ratio between the maximum possible power of a signal and the power of corrupting noise.

Phase-Altering Line (PAL) – The 625-line analogue colour video composite system.

Raster scan – A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entry in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc. rows of the pattern (going down) each scanned from left to right.

Reduced-reference (RR) – A video quality measurement methodology that utilizes low bandwidth features extracted from the original or processed video streams, as opposed to using full-reference video that requires complete knowledge of the original and processed video streams [2]. Reduced-reference methodologies have advantages for end-to-end in-service quality monitoring since the reduced-reference information is easily transmitted over ubiquitous telecommunications networks.

Region of interest (ROI) – An image lattice (specified in rectangle coordinates) that is used to denote a particular sub-region of a field or frame of video.

Scene – A sequence of video frames.

Spatial registration – A process that is used to estimate and correct spatial shifts of the processed video sequence with respect to the original video sequence.

Temporal registration – A process that is used to estimate and correct a temporal shift (i.e. video delay) of the processed video sequence with respect to the original video sequence.

Video Quality Metric (VQM) – An overall measure of video impairment. VQM is reported as a single number and has a nominal output range from zero to one, where zero is no perceived impairment and one is maximum perceived impairment.

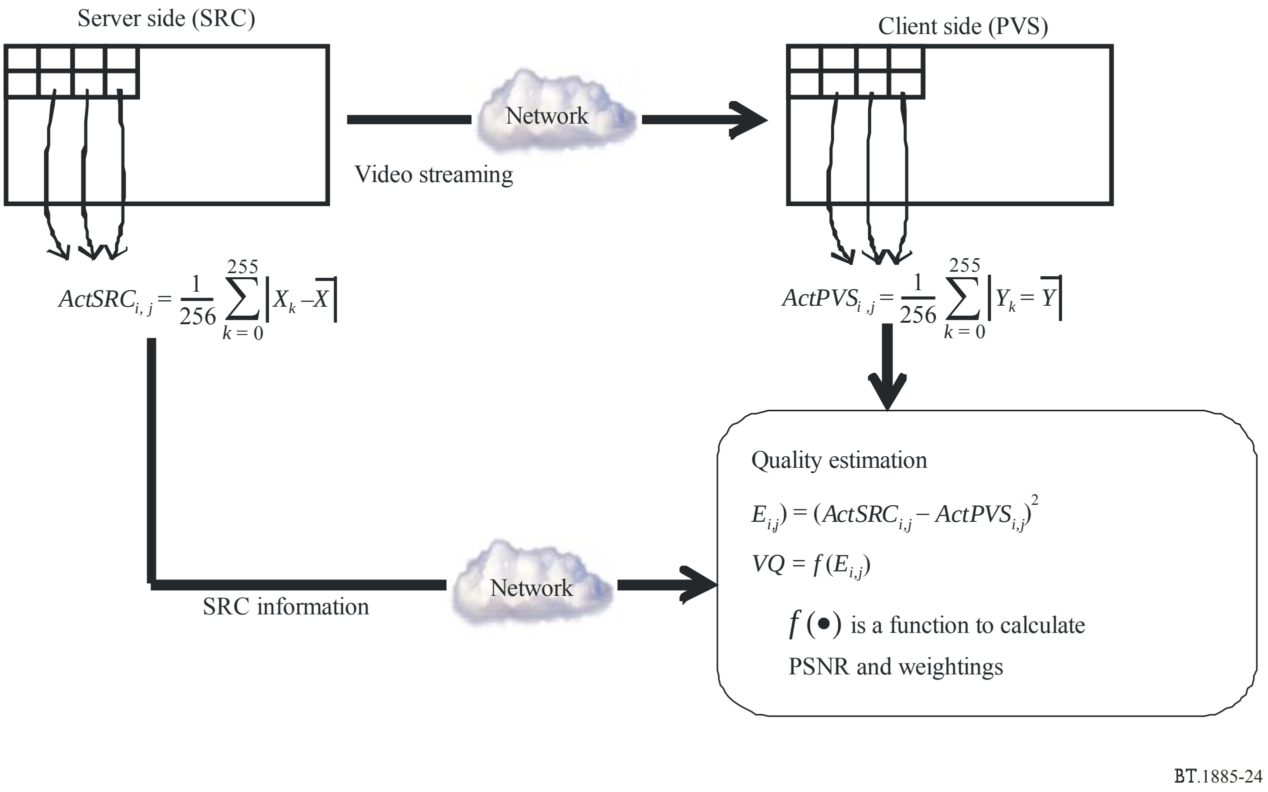
3 Overview of the Video Quality Metric computation

The RR model transmits the activity values for individual given-size pixel blocks to the client side. This value indicates the variance of the luminance values in the block. Figure 1 summarizes the RR model. As shown in Fig. 24, video quality is estimated on the basis of the activity-difference between the SRC and the PVS. In addition, psychovisual weightings with respect to the activity-difference are applied for higher estimation accuracy. Video quality estimation is conducted in the following steps:

1. The activity value for each 16 × 16 luminance pixel block of the SRC is calculated on the server side. All the activity values are then transmitted to the client side. The activity value of a block is defined as the mean absolute difference of individual values and their average.
2. The corresponding activity values are calculated on the client side with respect to the PVS.
3. On the client side, each block is initially evaluated with its squared error, that is the squared difference between the SRC and the PVS activity values.
4. Psychovisual-weightings are applied to the squared errors in blocks with high spatial frequency components, a specific colour, a large inter-frame difference, and a scene change.
5. A provisional video quality score is derived from the sum of the weighted squared errors in the same manner as in the PSNR calculation.
6. The score is modified to reflect perceptually fatal degradations due to blockiness and local impairment. Finally, the modified score represents the measured video quality of the PVS in the RR model.

FIGURE 24

Video quality estimation based on activity difference


Server side (SRC)
Client side (PVS)
Network
Video streaming
$ActSRC_{i,j} = \frac{1}{256}\sum_{k=0}^{255}\left|X_k - \overline{X}\right|$
$ActPVS_{i,j} = \frac{1}{256}\sum_{k=0}^{255}\left|Y_k = \overline{Y}\right|$
Quality estimation
$E_{i,j}) = (ActSRC_{i,j} - ActPVS_{i,j})^2$
$VQ = f(E_{i,j})$
$f(\bullet)$ is a function to calculate PSNR and weightings
Network
SRC information
BT.1885-24

4 Detailed algorithm

4.1 Server side

1. Luminance pixels of an SRC are divided into 16 × 16 pixel blocks in each frame after one second from the top of the video sequence. In the first one second, the SRC information is not transmitted since it is difficult for the human vision system to detect video quality degradation in the scenes just after the first frame.

2. In each block except for those in the frame rim, the activity values (SRC-activity: $ActSRC_{i,j}$) are calculated. Figure 25 describes the blocks which the activity values are calculated and transmitted. The SRC-activity is calculated as:

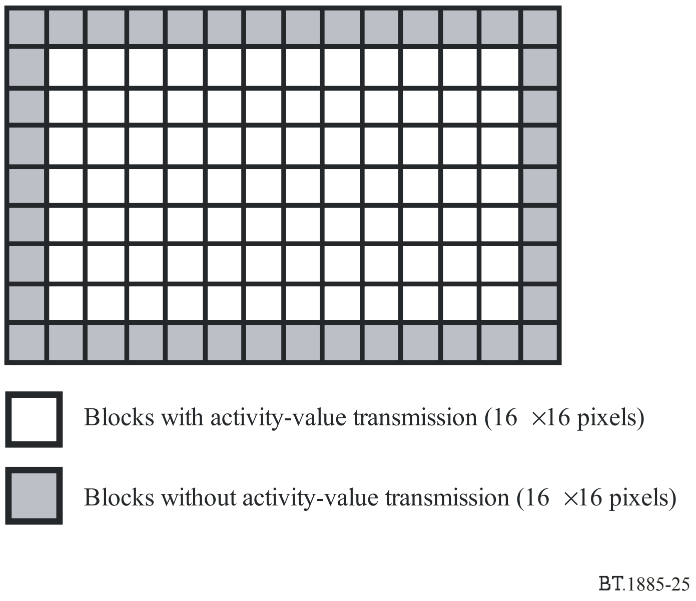
$$ActSRC_{i,j} = \frac{1}{256}\sum_{k=0}^{255}\left|X_k - \overline{X}\right|$$

where X_k is a luminance value in a given-size block of the SRC, $\overline{X}$ is its average, i is a frame number, and j is a block number in the frame.

3. Activity values, which are expressed with eight bits per block, are transmitted to the client side in the raster scan order after one second from the top of the video sequence. For SRC-information transmission at a rate of 256 kbit/s, the activity values are transmitted in all frames. When the rate is reduced to 80 kbit/s, the activity values are transmitted in every four frames.

FIGURE 25

Blocks with and without activity-value transmission


Blocks with activity-value transmission (16 ×16 pixels)
Blocks without activity-value transmission (16 ×16 pixels)
BT.1885-25

4.2 Client side

4.2.1 Squared-error calculation of the activity values

1. Luminance pixels of a PVS are divided into 16 × 16 pixel blocks in each frame after one second from the top of the video sequence.
2. In each block except for those in the frame rim, the activity values (PVS-activity: $ActPVS_{i,j}$) are calculated. At a rate of 256 kbit/s of SRC-information, the activity values are calculated in all frames. When the rate of SRC-information is reduced to 80 kbit/s, the activity values are calculated in every four frames.

$$ActPVS_{i,j} = \frac{1}{256}\sum_{k=0}^{255}\left|Y_k - \overline{Y}\right|$$

where:

Y_k:	a luminance value in a given-size block of the PVS
$\overline{Y}$:	its average
i:	a frame number
j:	a block number in the frame.

3. Squared errors between the SRC-activity and the PVS-activity are calculated

$$E_{i,j} = \left(ActSRC_{i,j} - ActPVS_{i,j}\right)^2$$

4.2.2 Psychovisual-weighting for the squared error

Three kinds of weightings, namely, weighting for the difference in spatial frequency, weighting for the difference in the specific colour region, and weighting for the inter-frame difference of luminance, are applied to $E_{i,j}$ to take into account the human visual-system characteristics.

1. Weighting for the difference in spatial frequency

A weighting factor W_{SF} and a threshold Th_{SF} are used for this weighting. (See Table 8 for the values of W_{SF} and Th_{SF}.)

$$E_{i,j} \Leftarrow \begin{cases} E_{i,j} \times W_{SF}, & ActPVS_{i,j} > Th_{SF} \\ E_{i,j}, & otherwise \end{cases}$$

2. Weighting for the difference in the specific colour region

For a given block and its surrounding eight blocks, if the number of pixels (*NumROIPixels*) within $48 \leq Y \leq 224$, $104 \leq Cb \leq 125$, and $135 \leq Cr \leq 171$ is larger than a threshold, the following weighting is performed using a weighting factor W_{CR} and a threshold Th_{CR}.

$$E_{i,j} \Leftarrow \begin{cases} E_{i,j} \times W_{CR}, & NumROIPixels > Th_{CR} \\ E_{i,j}, & otherwise \end{cases}$$

See Table 8 for W_{CR} and Th_{CR} values.

3. Weighting for the inter-frame difference of luminance

Mean absolute difference ($MAD_{i,j}$) of the luminance between a given block and that in the previous frame is calculated. $MAD_{i,j}$ is defined as:

$$MAD_{i,j} = \frac{1}{256} \sum_{k=0}^{255} \left| Y_k - Y'_k \right|$$

where Y_k is a luminance value in a 16 × 16 pixel block of the PVS and Y'_k is a luminance value in the same position in the previous frame.

The following weighting is performed using weighting factors W_{MAD1}, W_{MAD2} and thresholds Th_{MAD1}, Th_{MAD2}.

$$E_{i,j} \Leftarrow \begin{cases} E_{i,j} \times W_{MAD1}, & MAD_{i,j} > Th_{MAD1} \\ E_{i,j} \times W_{MAD2}, & MAD_{i,j} \leq Th_{MAD2} \\ E_{i,j}, & otherwise \end{cases}$$

See Table 8 for W_{MAD1}, W_{MAD2}, Th_{MAD1} and Th_{MAD2} values.

4.2.3 Weighting in case of scene change detection

An average of $MAD_{i,j}$ ($MADAve_i$) is calculated for each frame as:

$$MADAve_i = \frac{1}{M} \sum_{j=0}^{M-1} MAD_{i,j}$$

where M is the number of blocks in a frame.

If $MADAve_i$ is larger than a threshold Th_{SC}, it is considered as a scene change. When a scene change is detected, $E_{i,j}$ is set to 0 for 15 frames after the scene change.

$$SceneChange = \begin{cases} TRUE, & MADAve_i > Th_{SC} \\ FALSE & otherwise \end{cases}$$

$$E_{i,j} \Leftarrow \begin{cases} E_{i,j} \times W_{SC} & 15\ frames\ after\ SceneChange = TRUE \\ E_{i,j}, & otherwise \end{cases}$$

See Table 8 for W_{SC} and Th_{SC} values.

4.2.4 PSNR based on the squared error of the activity

A PSNR is calculated based on the activity-difference as:

$$VQ = 10 \times \log_{10} \frac{255 \times 255}{E_{Ave}}$$

$$E_{Ave} = \frac{1}{N \times M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} E_{i,j}$$

where N and M are the number of frames and blocks used for the PSNR calculation.

4.2.5 Weighting for blockiness artifacts

A weighting factor W_{BL}, a threshold Th_{BL}, and blockiness level information BL_{Ave} are used for this weighting. (See Table 8 for W_{BL} and Th_{BL} values.)

$$VQ \Leftarrow \begin{cases} VQ \times W_{BL}, & BL_{Ave} > Th_{BL} \\ VQ, & otherwise \end{cases}$$

BL_{Ave} is calculated by the following steps:

Step 1: Activity values for 8 × 8 pixel blocks in a PVS are calculated. As shown in Fig. 26, the average value (Act_{Ave}) of the two activity values in horizontally adjacent blocks ($ActBlock_1$, $ActBlock_2$) is calculated by:

$$Act_{Ave} = \frac{1}{2}\left(ActBlock_1 + ActBlock_2\right)$$

Step 2: The absolute difference of the luminance values along the boundary between two blocks is calculated. As illustrated in Fig. 26, $Y_{1,0}$ and $Y_{2,0}$ represent luminance values in the left and right blocks along the boundary. An average value of the absolute luminance difference, *DiffBound*, is expressed as:

$$DiffBound = \frac{1}{8} \sum_{i=0}^{7} \left| Y_{1,i} - Y_{2,i} \right|$$

Step 3: Blockiness level ($BL_{i,j}$) is defined by the ratio between *DiffBound* and Act_{Ave}, i.e.:

$$BL_{i,j} = \frac{DiffBound}{Act_{Ave} + 1}$$

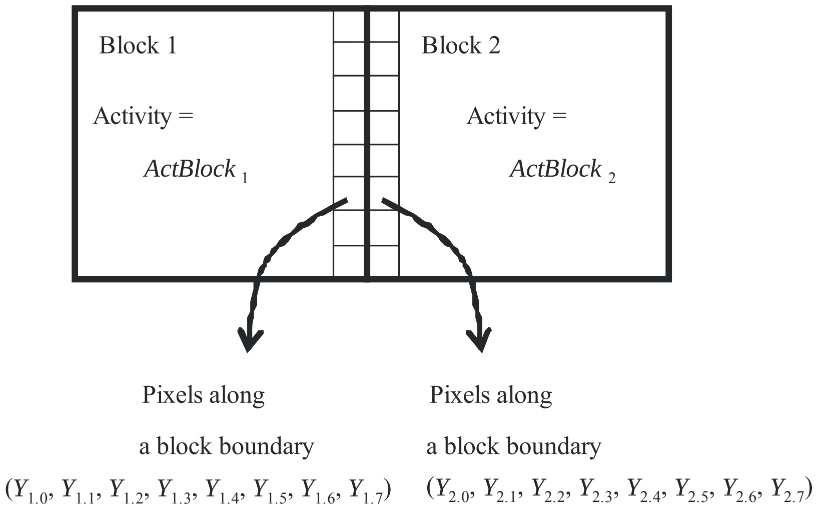
Step 4: The average value of the *BL* is calculated by:

$$BL_{Ave} = \frac{1}{N \times M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} BL_{i,j}$$

For the blocks in the rightmost in the frames, the $BL_{i,j}$ value is set to zero. If BL_{Ave} is larger than a predetermined threshold, it is considered that the video sequence includes a large level of blockiness and weighting is applied to the calculated video-quality value.

FIGURE 26

Pixel and activity values used for calculating blockiness level



BT.1885-26

4.2.6 Weighting for local impairment artifacts

A weighting factor W_{LI}, a threshold Th_{LI}, and local impairment *LI* are used for this weighting. (See Table 8 for W_{LI} and Th_{LI} values.)

$$VQ \Leftarrow \begin{cases} VQ \times W_{LI}, & LI < Th_{LI} \\ VQ, & otherwise \end{cases}$$

LI is calculated by the following steps. The difference of the activity-variance is used to detect local impairment by transmission errors.

1. For a given block and its surrounding eight blocks, the variance of the activity is calculated for both the SRC ($ActVar_{SRC}$) and the PVS ($ActVar_{PVS}$) and the absolute difference of these variance values is calculated as:

$$\Delta ActVar = \left| ActVar_{SRC} - ActVar_{PVS} \right|$$

2. The average value of this absolute difference values is calculated for each frame.
3. *LI* is calculated as the ratio of the maximum ($\Delta ActVar_{Max}$) to the minimum ($\Delta ActVar_{Min}$) of the average.

$$LI = \begin{cases} \Delta ActVar_{Min} / \Delta ActVar_{Max} & \Delta ActVar_{Max} \neq 0 \\ 1 & \Delta ActVar_{Max} = 0 \end{cases}$$

VQ represents the video quality score.

4.2.7 Parameters for the weightings

Table 8 shows the parameter values for the weightings. These values are determined by a preliminary experiment with training data set.

TABLE 8

Parameters for the weightings

Weighting operation type	Parameter values	
Spatial frequency weighting	W_{SF}	0.36
	Th_{SF}	25
Specific colour weighting	W_{CR}	4.0
	Th_{CR}	175
Inter-frame difference weighting	W_{MAD1}	0.06
	Th_{MAD1}	17
	W_{MAD2}	25
	Th_{MAD2}	13
Scene change detection	W_{SC}	0.0
	Th_{SC}	35
Blockiness weighting	W_{BL}	0.870
	Th_{BL}	1.0
Local impairment weighting	W_{LI}	0.870
	Th_{LI}	1.67

4.2.8 Registration

1. Spatial registration

The RR model does not need any spatial registration. This is because the squared error is calculated based on the activity values that are more robust to spatial shifts than those based on the pixel values.

2. Gain and offset registration

The RR model does not need gain-and-offset registration. The activity values are inherently free from the offset (i.e. DC components) and insensitive to the gain.

3. Temporal registration

The PVS sequence is divided into 1-second sub-sequences. For each sub-sequence, the mean squared errors of the activity are calculated with five delay variations of up to ±2 SRC frames. The minimum value of the mean squared errors is finally used as the mean squared error in this sub-sequence. The delay that results in this minimum mean squared error is adjusted as temporal registration.

5 Sample codes

C-language sample codes for the RR model are provided here.

5.1 Common code both server and client sides

```
// Calcule the activity value
unsigned int CalcActivitybyRect(unsigned char * lpBuffer, int nWidth, int iRectWidth, int iRectHeight)
{
              // lpBuffer: Luminance Frame Buffer
        // nWidth: Frame Buffer Width
        // iRectWidth: Width of the rectangle to calculate an activity value.
        // iHeightWidth: Height of the rectangle to calculate an activity value.
              unsigned int i, j, nTmp, nSum;
              unsigned char *pSrc;

              pSrc = lpBuffer;   nSum = 0;
              for (j = 0; j < iRectHeight; j++){
                     for (i = 0; i <iRectWidth; i++){
                            nSum += pSrc[i];
                     }
                     pSrc += nWidth;
              }
              nSum /= (iRectWidth*iRectHeight);

              pSrc = lpBuffer; nTmp = 0;
              for (j = 0; j < iRectHeight; j++){
                     for (i = 0; i <iRectWidth; i++){
                            nTmp += abs(pSrc[i] - nSum);
                     }
                     pSrc += nWidth;
              }
              return nTmp/iRectWidth/iRectHeight;
}
```

5.2 Server side

```
// Server side
int nStart = 30; // the frame number to start transmission (30 or 25)
int nMaxFrame = 240; // the number of total video frames (240 or 200)
int nFrameIncrement = 1;  // 1 for 256kbps,  4 for 80kbps
void ReadOneFrame(unsigned char, int, unsigned char *, int, int); // function to read one frame data
int nRim = 16 // 16 or 32 (use 32 to avoid the problem in HRC9)

// nWidth: Frame Buffer Width
// nHeight: Frame Buffer Height
// lpSrc: Frame Buffer
for(int nFrame = nStart;  nFrame < nMaxFrame;  nFrame+=nFrameIncrement){
              ReadOneFrame(SRC_file_name, nFrame, lpSrc, nWidth, nHeight);
              for (j= 16; j<nHeight-32; j+=16) {
                     for (i= nRim; i<nWidth- nRim; i+=16) {
                            lpOrg = lpSrc + i + j * nWidth;
                            nActSrc = CalcActivitybyRect(lpOrg, nWidth, 16, 16);
                // OutputSRCInfo(nActSrc);  // Output or transmission the SRC information
                     }
              }
}
```

5.3 Client side

```
// Client Side
int nStart = 30; // the frame number to start transmission (30 or 25)
int nMaxFrame = 240; // the number of total video frames  (240 or 200)
int nFrameIncrement = 1;  // 1 for 256kbps,  4 for 80kbps
int nFrameRate = 30;  //30 or 25
void ReadOneFrame(unsigned char, int, unsigned char **, int, int); // function to read one frame data
void ReadRRData(unsigned char, int, unsigned char *); // function to read RR-data

// nWidth: Frame Buffer Width
// nHeight: Frame Buffer Height
// lpPvsByte[3]: Frame Buffer  (0:Y, 1:Cb, 2:Cr)
// lpRRData: RR-data Buffer
// double ddActivityDifference[][]:  Store the activity-difference
// double ddActivityVariance[][]:   Store the activity-variance
// double ddBlock[][]:              Store the blockiness level
// int nSceneChange:                Scene change detection

for(int nTemporalAlign = -2; nTemporalAlign <=2; nTemporalAlign++){  // Changing temporal alignment
        for(int nFrame = 0; nFrame < nMaxFrames; nFrame++){
            if(nFrame+nTemporalAlign >= nMaxFrames || nFrame+nTemporalAlign < 0){
                continue;
            }
            ReadOneFrame(PVS_file_name, nFrame+nTemporalAlign, lpPvsByte, nWidth, nHeight);
            if(((nFrame-(nFrameRate+nStart)) % nFrameIncrement) == 0
                    && nFrame >= nStart ){
                ReadRRData(RR_file_name, nFrame, lpRRData);
                ddActivityDifference[nTemporalAlign+2][nFrame]
                                  = RRCalcObjectiveScore(lpPvsByte, lpRRData, nWidth, nHeight);
                ddActivityVariance[nTemporalAlign+2][nFrame] = gnActVar;
            }else{
                ddActivityDifference[nTemporalAlign+2][nFrame] = 0.0;
                ddActivityVariance[nTemporalAlign+2][nFrame] = 0.0;
            }
            // Blockiness Level
            if(nTemporalAlign ==0){
                ddBlock[nFrame] = BlockinessLevelEstimation(lpPvsByte[0], nWidth, nHeight);
            }
            // Pixel copy for inter-frame difference calculation
            memcpy(lpPrev, lpPvsByte[0], sizeof(char)*nWidth*nHeight);
            if(nSceneChange){
                nSceneChange--;
            }
        }
}

double ddSum[8][5]; // Sum of the Activity-difference for each second
double ddActVarSum[8][5]; // Sum of the Activity-variance for each second
double ddActVarMax[8][5]; // Maximum of the Sum of the Activity-variance
double ddActVarMin[8][5]; // Minimum of the Sum of the Activity-variance
int nnMin[8];
int nnNumFrames[8][5];
#define LARGENUMBER 100000
for(int nTemporalAlign = -2; nTemporalAlign <=2; nTemporalAlign++){
        for(int j=0;j<8;j++){ // for each one second
            nnNumFrames[j][nTemporalAlign+2] = 0;
            ddActVarMax[j][nTemporalAlign+2] = 0.0;
            ddActVarMin[j][nTemporalAlign+2] =  LARGENUMBER;
            ddActVarSum[j][nTemporalAlign+2] = 0.0;
            ddSum[j][nTemporalAlign+2] = 0.0;
            for(int i=nFrameRate*j;i< (j+1)*nFrameRate; i++){
                if(ddActivityDifference[nTemporalAlign+2][i]){
                    ddSum[j][nTemporalAlign+2] += ddActivityDifference[nTemporalAlign+2][i];
                    nnNumFrames[j][nTemporalAlign+2]++;
                }
                ddActVarSum[j][nTemporalAlign+2] += ddActivityVariance[nTemporalAlign+2][i];
                if(ddActivityVariance[nTemporalAlign+2][i]){
                                  if(ddActivityVariance[nTemporalAlign+2][i] >
                                  ddActVarMax[j][nTemporalAlign+2]){
                                          ddActVarMax[j][nTemporalAlign+2] =
                                          ddActivityVariance[nTemporalAlign+2][i];
                    }
                                  if(ddActivityVariance[nTemporalAlign+2][i] <
                                  ddActVarMin[j][nTemporalAlign+2]){
                                          ddActVarMin[j][nTemporalAlign+2] =
                                          ddActivityVariance[nTemporalAlign+2][i];
                    }
                }
            }
        }
}

// Local Impariment Level Calculation
```

```
double dSum = 0.0;
double dActMax = 0.0;
double dActMin = LARGENUMBER;
int nNumFrames = 0;
for(int j=1; j<8; j++){
        double dMin = LARGENUMBER;
        double dMinSum = LARGENUMBER;
        for(int nTemporalAlign = -2; nTemporalAlign <=2; nTemporalAlign++){
            if(ddActVarSum[j][nTemporalAlign+2] < dMin){
                dMin = ddActVarSum[j][nTemporalAlign+2];
                dMinSum = ddSum[j][nTemporalAlign+2];
                nnMin[j] = nTemporalAlign+2;
            }
        }
        dSum += dMinSum;
        nNumFrames += nnNumFrames[j][nnMin[j]];
        if(ddActVarMax[j][nnMin[j]] > dActMax){
            dActMax = ddActVarMax[j][nnMin[j]];
        }
        if(ddActVarMin[j][nnMin[j]] < dActMin){
            dActMin = ddActVarMin[j][nnMin[j]];
        }
}
double dTransError =  dActMax/dActMin;

// Blockiness Level Calculation
double dBlockinessLevel = 0.0;
for(int i=0;i<nMaxFrames; i++){
        dBlockinessLevel += ddBlock[i];
}
dBlockinessLevel = dSumBlock / (double)(nMaxFrames-nFrameRate);

// Calculating the Video Quality Score
if(nNumFrames && nNumberOfBlocks && dSum){
        dSum = dSum / (double)(nNumFrames)/(double)nNumberOfBlocks;
        dSum = 10*log10(255.0*255.0/dSum); //PSNR based on the activity difference
        if(dBlockinessLevel > dBlokinessTh){
            dSum /= nBlockinessWeighting;  // Weighting for blockiness level
        }
        if(dTransError > nErrorTh){
            dSum /=nErrorWeightin;   // Weighting for transmission errors
        }
}
return dSum;
------------------------------------------------------------------------------------------
// Calculating the MAD value
unsigned int CalcMAD(unsigned char *lpSrc, unsigned char *lpSrc2, int nWidth, int nHeight)
{
        // lpSrc:   Frame Buffer of the current frame
         // lpSrc2:  Frame Buffer of the previous frame
         unsigned int nSum = 0;
        for (y = 0; y < nHeight; y++) {
            for (x = 0; x < nWidth; x++) {
                nSrc =  lpSrc[x  + y*nWidth];
                nSrc2 = lpSrc2[x + y*nWidth];
                nSum += abs(nSrc - nSrc2);
            }
        }
        return nSum/nWidth/nHeight;
}

// Calculating a mean squared error with weightings
double RRCalcObjectiveScore(unsigned char *lpBuffer[], unsigned char *lpRRData, int nWidth, int nHeight)
{
        int    i, j, nActSrc, nActSrc2, nY, nCb, nCr, nYMin, nYMax, nCbMin, nCbMax, nCrMin, nCrMax;
        int    nMBX, nMBY, nMB, nStart;
        unsigned int nMAD;
        double     e2, dMADFrame;
        unsigned char *lpRec, *lpRecCb, *lpRecCr, *lpPrev1;
```

```
int nRim = 16 // 16 or 32 (use 32 to avoid the problem in HRC9)

nYMin = 48; nYMax = 224; nCbMin = 104; nCbMax = 125; nCrMin = 135; nCrMax = 171;
nMB = nMBY = nStart = 0;
e2 = dMADFrame = 0.0;

for (j=16+nStart; j<iImageHeight-32; j+=16) {
        nMBX = 0;
        for (i= nRim; i<nWidth- nRim; i+=16) {
                lpRec = lpBuffer[0] + i + j * nWidth;
                lpRecCb = lpBuffer[1] + i/2 + (j/2) * nWidth/2;
                lpRecCr = lpBuffer[2] + i/2 + (j/2) * nWidth/2;
                lpPrev1 = lpPrev + i + j * nWidth;

                nActSrc = lpRRData[nMB];  // SRC activity
                nActSrc2 = CalcActivitybyRect(lpRec, nWidth, 0, 16, 16); // PVS activity
                nActArray[nMB] = (double)nActSrc;
                nActArray2[nMB] = (double)nActSrc2;
                e2 += (double)(nActSrc - nActSrc2)*(nActSrc - nActSrc2);  // Mean squared error

                nMAD     = CalcMAD(lpRec, lpPrev1, 16, 16, nWidth);  // Inter-frame differnece
                dMADFrame += (double)nMAD;

                int nNumROIPixels=0;
                for(int jj=-16;jj<32; jj++){
                        for(int ii=-16;ii<32; ii++){
                           nY  = lpRec[ii];
                           nCb = lpRecCb[ii/2];
                           nCr = lpRecCr[ii/2];
                           if(nY >= nYMin && nY <= nYMax
                           && nCb >= nCbMin && nCb <= nCbMax
                           && nCr >= nCrMin && nCr <= nCrMax){
                                        nNumROIPixels++;
                           }
                        }
                        lpRec += nWidth;
                        if((jj & 1) == 1){
                           lpRecCb += nWidth/2;
                           lpRecCr += nWidth/2;
                        }
                }

                // Weighting for spatial frequency
                if(nActSrc2 > gdwActThHigh){
                        e2 *= dActWeightingHigh;
                }else if(nActSrc2 > gdwActThLow){
                        e2 *= dActWeightingMiddle;
                }else {
                        e2 *= dActWeightingLow;
                }

                // Weighting for specific color region
                if(nNumROIPixels > dwROITh){
                        e2 *= dROIWeighting;
                }

                // Weighting for inter-frame difference
                if(nMAD > dwMADThHigh){
                        e2 *= dMADWeightingHigh;
                }else if(nMAD > dwMADThLow){
                        e2 *= dMADWeightingMiddle;
                }else {
                        e2 *= dMADWeightingLow;
                }
                nMB++;          nMBX++;
        }
        nMBY++;
}
```

```
        // Calculating Activity-Variance for Surrounding Nine Blocks.
        double nSumActSrc, nSumActPvs, nActVar, nActVar2;
        nSumActSrc = nSumActPvs = nActVar = nActVar2 = 0.0;
        gnActVar = 0.0;
        for (j=1; j<nMBY-1; j++) {
            for (i=1; i<nMBX-1; i++) {
                nSumActSrc = 0.0;
                nSumActPvs = 0.0;
                for(int jj=-1; jj<2; jj++){
                    for(int ii=-1; ii<2; ii++){
                        nSumActSrc += nActArray[i+ii+nMBX*(j+jj)];
                        nSumActPvs += nActArray2[i+ii+nMBX*(j+jj)];
                    }
                }
                nSumActSrc /= 9.0;
                nSumActPvs /= 9.0;

                nActVar = 0.0;
                nActVar2 = 0.0;
                for(int jj=-1; jj<2; jj++){
                    for(int ii=-1; ii<2; ii++){
        nActVar +=  (nActArray[i+ii+nMBX*(j+jj)]-nSumActSrc)*(nActArray[i+ii+nMBX*(j+jj)]-nSumActSrc);
                                        nActVar2 += (nActArray2[i+ii+nMBX*(j+jj)]-
                                        nSumActPvs)*(nActArray2[i+ii+nMBX*
                                        (j+jj)]-nSumActPvs);
                    }
                }
                nActVar /= 9.0;
                nActVar2 /= 9.0;
                gnActVar += abs(nActVar- nActVar2);
            }
        }
        // Average of the Activity-Variance for the Frame
        gnActVar = gnActVar/(double)(nMBY-2)/(double)(nMBY-2);

        // Scene change detection
        if(dMADFrame/ nMB > 35){
            nSceneChange = 15;
        }
        if(nSceneChange){
            e2 = 0.0;
        }
        gnFrame++;

        return e2;
}


// Calculate Blockiness Level
double BlockinessLevelEstimation(unsigned char *lpBuffer, int nWidth, int nHeight)
{
        // lpBuffer: Frame Buffer
        int    i, j, nActSrc, nActSrc2, nDiff, nMB;
        unsigned char *lpRec = lpBuffer;
        double dBlock=0.0;

        nMB = 0;
        for (j=0; j<nHeight-16; j+=8) {
            for (i=0; i<nWidth-16; i+=8) {
                lpRec = lpBuffer + i + j * nWidth;
                nActSrc  = CalcActivitybyRect(lpRec, nWidth, 0, 8, 8); // Activity of the left block
                nActSrc2 = CalcActivitybyRect(lpRec+8, nWidth, 0, 8, 8); // Activity of the right block
                nActSrc = (nActSrc + nActSrc2)/2; // Average of the activity values
                nDiff = 0;
                for(int jj=0;jj<8; jj++){
                    nDiff += abs(lpRec[7+jj*nWidth] - lpRec[8+jj*nWidth]);
                                    // Difference of the luminance values at the boundary
                }
                nDiff/= 8;
                dBlock += (double)nDiff/(double)(nActSrc+1);
                nMB++;
            }
        }
        return (double)dBlock/(double)nMB;
}
```

6 Informative references

[1] SMPTE 170M, "SMPTE Standard for Television – Composite Analog Video Signal – NTSC for Studio Applications", Society of Motion Picture and Television Engineers.

[2] ITU-T Recommendation J.143 – User requirements for objective perceptual video quality measurements in digital cable television.

Annex C

Model C: NTIA reduced reference method

1 Background

In the 2003-2004 time-frame, the U.S. National Telecommunications and Information Administration (NTIA) developed two video quality models (VQMs) with a reduced reference (RR) bandwidth of approximately 12 to 14 kbits/s for Recommendation ITU-R BT.601 sampled video [1]. These models were called the "Low Bandwidth VQM" and "Fast Low Bandwidth VQM". The Fast Low Bandwidth VQM was a computationally efficient version of the Low Bandwidth VQM. The Fast Low Bandwidth VQM is about four times faster since it extracts spatial features from video frames that are first pre-averaged, rather than extracting spatial features directly from the ITU-R BT.601 video frames. Additional computational savings for the Fast Low Bandwidth VQM resulted from computing temporal information (i.e. motion) features based on a random sub-sampling of pixels in the luminance Y channel rather than using all pixels in all three video channels (Y, Cb, and Cr). Both VQMs have been available in our VQM software tools for several years and may be freely used for both commercial and non-commercial applications. Binary executable versions of these VQM tools and their associated source code are available for download [2].

Since NTIA wanted to submit both the Low Bandwidth and Fast Low Bandwidth VQMs to the reduced reference TV (RRTV) tests for independent evaluation by the Video Quality Experts Group (VQEG), NTIA choose to submit them to different bit-rate categories even though they have identical RR bit-rate requirements. NTIA chose to submit the Low Bandwidth VQM to the 256k category and the Fast Low Bandwidth VQM to the 80k category since the performance of the Low Bandwidth VQM was expected to be superior to that of the Fast Low Bandwidth VQM. Both VQMs utilized the NTIA RR calibration algorithm which is included in ITU-T Recommendation J.244 [3]. This calibration algorithm requires approximately 22 to 24 kbits/s of RR bandwidth to produce estimates for temporal delay, spatial shift, spatial scaling, gain, and level offset.

An interesting result from the VQEG RRTV evaluation tests [4] was that the Fast Low Bandwidth VQM outperformed the Low Bandwidth VQM for both the 525-line and 625-line test. This is an interesting result since it implies that extracting spatial features from averaged frames is superior to extracting them from non-averaged frames. Whether or not this result will prove true for other data sets is an area for further research. At this time, NTIA does not see a reason to standardize both models so this Annex only describes the Fast Low Bandwidth VQM.

2 Introduction

This Annex presents a description and reference code for the NTIA Fast Low Bandwidth VQM. The NTIA Fast Low Bandwidth VQM utilizes techniques similar to those of the NTIA General VQM, a description of which can be found in both ITU-T Recommendation J.144 [5] and Recommendation ITU-R BT.1683 [6]. The Fast Low Bandwidth VQM uses RR features with much less bandwidth than the NTIA General VQM, but the feature extraction and comparison process is similar for both. For Recommendation ITU-R BT.601 sampled video [1], the Fast Low Bandwidth VQM uses RR features that require approximately 12 to 14 kbits/s of transmission bandwidth. This Annex only describes the Fast Low Bandwidth VQM since its complementary low bandwidth calibration algorithms are fully documented in ITU-T Recommendation J.244 [3]. However, for completeness, the reference code in this Annex includes both the Fast Low Bandwidth VQM and its associated low bandwidth calibration algorithms. The reference code also contains example quantization functions for the features used by the low bandwidth calibration (these quantization functions are not part of ITU-T Recommendation J.244).

3 Fast Low Bandwidth VQM description

3.1 VQM overview

The VQM description will encompass three primary areas:

1) the low bandwidth features that are extracted from the original and processed video streams;

2) the parameters that result from comparing like original and processed feature streams;

3) the VQM calculation that combines the various parameters, each of which measures a different aspect of video quality.

This description makes use of readily available references for the technical details.

3.2 Feature description

3.2.1 Feature overview

Three types of features are used by the Fast Low Bandwidth VQM: colour, spatial, and temporal. Each of these feature types quantify perceptual distortions in their respective domains. The reference code subroutine "model_fastlowbw_features" provides a complete mathematical description of the features used by the Fast Low Bandwidth VQM.

3.2.2 Colour features

The colour features are the same f_{COHER_COLOR} features that are used by the NTIA General VQM. These features are described in detail in Annex D.7.3 of ITU-T Recommendation J.144. The f_{COHER_COLOR} features provide a 2-dimensional vector measurement of the amount of blue and red chrominance information (C_B, C_R) in each S-T region. Thus, the f_{COHER_COLOR} features are sensitive to colour distortions. The f_{COHER_COLOR} features of the NTIA Fast Low Bandwidth VQM are extracted from spatial-temporal (S-T) region sizes of 30 vertical lines by 30 horizontal pixels by 1 s of time (i.e. $30 \times 30 \times 1$ s) whereas the NTIA General VQM used S-T region sizes of $8 \times 8 \times 1$ frame.

3.2.3 Spatial features

The spatial features are the same f_{SI13} and f_{HV13} features that are used by the NTIA General VQM. These features are described in detail in Annex D.7.2.2 of ITU-T Recommendation J.144. The f_{SI13} and f_{HV13} features measure the amount and angular distribution of spatial gradients in spatial-temporal (S-T) sub-regions of the luminance (Y) images. Thus, the f_{SI13} and f_{HV13} features are sensitive to spatial distortions such as blurring and blocking.

The f_{SI13} and f_{HV13} features of the NTIA Fast Low Bandwidth VQM are extracted from spatial-temporal (S-T) region sizes of 30 vertical lines by 30 horizontal pixels by 1 second of time (i.e. 30 × 30 × 1s) whereas the NTIA General VQM used S-T region sizes of 8 × 8 × 0.2s. In addition, to save computations, the 1 s of luminance Y images are first pre-averaged across time before applying the two-dimensional 13 × 13 edge enhancement filters given in Annex D.7.2.1 of ITU-T Recommendation J.144.

One additional spatial feature is extracted from the 1 s of pre-averaged luminance (Y) images. This feature is the *mean* luminance (Y) level of each 30 × 30 × 1s S-T region (denoted here as f_{MEAN}). The purpose of the f_{MEAN} feature is to provide a luminance-level perceptual weighting function for spatial information (SI) loss as measured by the f_{SI13} and f_{HV13} features. This will be described in the parameter description section.

3.2.4 Temporal features

Powerful estimates of perceived video quality can be obtained from the colour and spatial feature set described above. However, since the S-T regions from which these features are extracted span many video frames (i.e. 1 s of video frames), they tend to be insensitive to brief temporal disturbances in the picture. Such disturbances can result from noise or digital transmission errors; and, while brief in nature, they can have a significant impact on the perceived picture quality. Thus, a temporal-based RR feature is used to quantify the perceptual effects of temporal disturbances. This feature measures the absolute temporal information (ATI), or motion, in the luminance Y image plane and is computed as:

$$f_{ATI} = rms\{rand5\%|Y(t) - Y(t - 0.2s)|\}$$

For computational efficiency, Y is randomly sub-sampled to contain only 5% of the image pixels (represented here by the *rand5%* function). The sub-sampled Y image at time t-0.2s is subtracted from the identically sub-sampled Y image at time t and the root mean square error (*rms*) of the result is used as a measure of ATI. Using the convention found in Annex D.8 of ITU-T Recommendation J.144, this will also be denoted as:

$$f_{ATI} \cong \text{Y\_rand5\%\_ati0.2s\_rms}$$

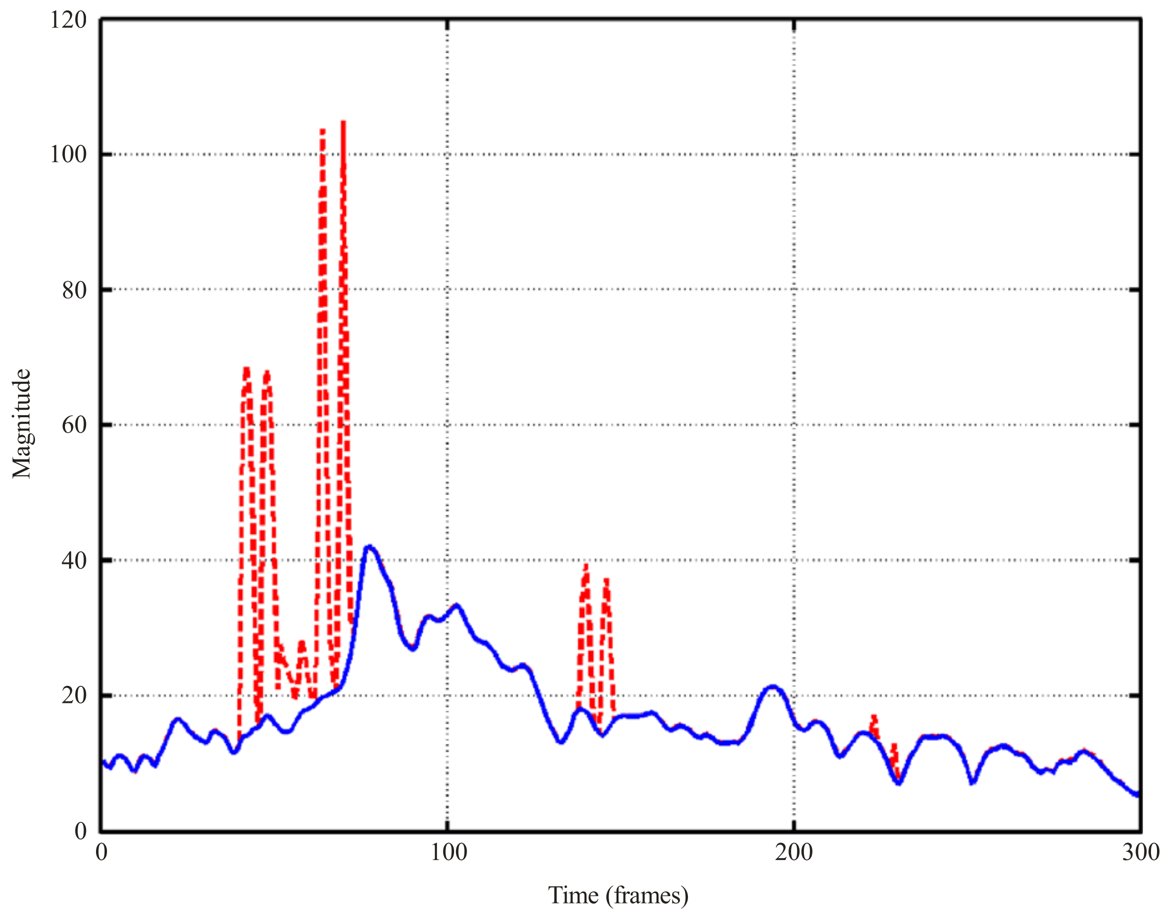
The feature f_{ATI} is sensitive to temporal disturbances. For 30 fps video, 0.2s is 6 video frames while for 25 fps video, 0.2s is 5 video frames. Subtracting images 0.2s apart makes the feature insensitive to real time 30 fps and 25 fps video systems that have frame update rates of at least 5 fps. The quality aspects of these low frame rate video systems, common in multimedia applications, are sufficiently captured by the f_{SI13}, f_{HV13}, and f_{COHER_COLOR} features. The 0.2s spacing is also more closely matched to the peak temporal response of the human visual system than differencing two images that are one frame apart in time.

Figure 27 provides an example plot of the f_{ATI} feature for a original (solid blue) and processed (dashed red) video scene from a digital video system with transient burst errors in the digital transmission channel. Transient errors in the processed picture create spikes in the f_{ATI} feature. The bandwidth required to transmit the f_{ATI} feature is extremely low since it requires only 30 samples per second for 30 fps video. Other types of additive noise in the processed video, such as might be generated by an analogue video system, will appear as a positive DC shift in the time history of the processed feature stream with respect to the original feature stream. Video coding systems that eliminate noise will cause a negative DC shift.

Before extracting a transient error parameter from the f_{ATI} feature streams shown in Fig. 27, it is advantageous to increase the width of the motion spikes (red spikes in Fig. 27). The reason is that short motion spikes from transient errors do not adequately represent the perceptual impact of these types of errors. One method for increasing the width of the motion spikes is to apply a maximum filter to both the original and processed feature streams before calculation of the error parameter function between the two waveforms. For the f_{ATI} based error parameter, a 7-point wide maximum filter (which will be denoted here as the *max7pt* function) was used that produces an output sample at each frame that is the maximum of itself and the 3 nearest neighbours on each side (i.e. earlier and later time samples).

FIGURE 27

Example time history of f_{ATI} feature



BT.1885-27

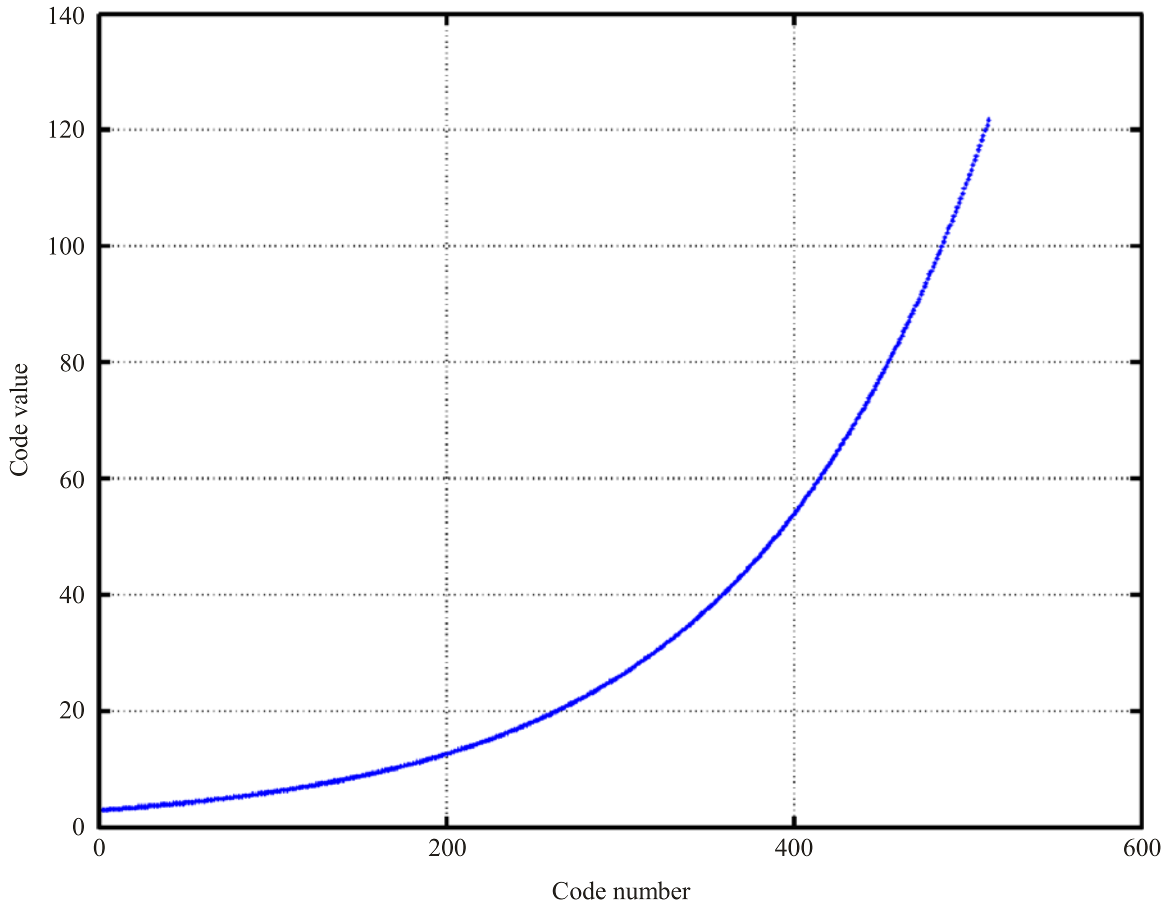
3.2.5 Quantization of features

Quantization to 9 bits of accuracy is sufficient for the Y_{MEAN}, f_{SI13}, f_{HV13}, and f_{COHER_COLOR} features, while the f_{ATI} feature should be quantized to 10 bits. To have minimal effect on the video quality parameter calculations, a non-linear quantizer design should be used where the quantizer error is proportional to the magnitude of the signal being quantized. Very low values are uniformly quantized to some cutoff value, below which there is no useful quality assessment information. Such a quantizer design minimizes the error in the corresponding parameter calculations because these calculations are normally based on an error ratio or log ratio of the processed and original feature streams (see the parameter description section given below).

Figure 28 provides a plot of the 9-bit non-linear quantizer used for the f_{SI13} original feature. The reference code subroutine "model_lowbw_compression" provides a complete mathematical description of the recommended quantizers used by the Fast Low Bandwidth VQM. If the features fall outside the range of the recommended quantizers on the low or high end (highly unlikely), then the S-T parameters derived from these features are zeroed so they do not influence the overall VQM.

FIGURE 28

Non-linear 9-bit quantizer for the f_{SI13} feature



BT.1885-28

3.3 Parameter description

3.3.1 Parameter overview

Several steps are involved in the calculation of parameters that track the various perceptual aspects of video quality. The steps may involve:

– applying a perceptual threshold to the extracted features from each S-T subregion;

– calculating an error function between processed features and corresponding original features;

– pooling the resultant error over space and time.

See Annex D.8 of ITU-T Recommendation J.144 for a detailed description of these techniques and their accompanying mathematical notation for parameter names, which will also be used here. The reference code subroutine "model_fastlowbw_parameters" provides a complete mathematical description of the parameters used by the Fast Low Bandwidth VQM. For simplicity, the description of the parameters in this section does not consider the effects of feature quantization (e.g. handling feature values that might lie outside of the recommended quantization ranges).

3.3.2 New methods

This section will summarize new methods that have been found to improve the objective to subjective correlation of parameters based on RR features with very low transmission bandwidths such as those utilized for the NTIA Fast Low Bandwidth VQM (i.e. new methods not found in ITU-T Recommendation J.144). It is worth noting that no improvements have been found for the basic form of the parameter error functions given in Annex D.8.2.1 of ITU-T Recommendation J.144. The two error functions that consistently produce the best parameter results (for spatial and temporal parameters) are a logarithmic ratio $\{\log 10\ [f_p(s,t)\ /\ f_o(s,t)]\}$ and an error ratio $\{[f_p(s,t) - f_o(s,t)]\ /\ f_o(s,t)\}$, where $f_p(s,t)$ and $f_o(s,t)$ are the processed feature and corresponding original feature extracted from the S-T region with spatial coordinates s and temporal coordinates t, respectively. Errors must be separated into gains and losses, since humans respond differently to additive (e.g. blocking) and subtractive (e.g. blurring) impairments. Applying a lower perceptual threshold to the features before application of these two error functions prevents division by zero.

After computation of the S-T parameters using one of the error functions, then the S-T parameters must be pooled over space and time to produce a parameter value for the video clip. This error pooling can occur in multiple stages (e.g. over space and then over time). One new error pooling method that is utilized by the Fast Low Bandwidth VQM is called Macro-Block (MB) error pooling. MB error pooling groups a contiguous number of S-T sub-regions and applies an error pooling function to this set. For instance, the function denoted as "MB(3,3,2)max" will perform a max function over parameter values from each group of 18 S-T sub-regions that are stacked 3 vertical by 3 horizontal by 2 temporal. For the 32 x 32 x 1s S-T sub-regions of the f_{SI13}, f_{HV13}, and f_{COHER_COLOR} features described above, each MB(3,3,2) region would encompass a portion of the video stream that spans 96 vertical lines by 96 horizontal pixels by 2 seconds of time. MB error pooling has been found to be useful in tracking the perceptual impact of impairments that are localized in space and time. Such localized impairments often dominate the quality decision process. MB error pooling can also be implemented as a filtering process so that instead of producing a single output value for each MB, each S-T sample is replaced with its MB filtered value, where the MB is centred on the S-T sample. This is called Overlapped MB (OMB) error pooling.

A second error pooling method is a generalized Minkowski(P,R) summation, defined as:

$$Minkowski(P,R) = \sqrt[R]{\frac{1}{N}\sum_{i=1}^{N}|v_i|^P}$$

Here v_i represents the parameter values that are included in the summation. This summation might, for instance, include all parameter values at a given instance in time (spatial pooling), or may be applied to the macro-blocks described above. The Minkowski summation where the power *P* is equal to the root *R* has been used by many developers of video quality metrics for error pooling. The generalized Minkowski summation, where $P \neq R$, provides additional flexibility for linearizing the response of individual parameters to changes in perceived quality. This is a necessary step before combining multiple parameters into a single estimate of perceived video quality, which is performed with a linear fit.

3.3.3 Colour parameters

Two parameters are extracted from the f_{COHER_COLOR} features. One of these parameters, called *color_extreme*, measures extreme colour distortions that might be caused by coloured blocks from transmission errors. The other parameter, called *color_spread*, provides an indication of the variance or spread in the colour errors. Rather than using the Euclidean distance measure to quantify distortions (as in Annex D.8.2.2 of ITU-T Recommendation J.144), both of these parameters use the square root of the Manhattan distance. Following the mathematical notation in Annex D.8.2.2 of ITU-T Recommendation J.144, where $f_p(s,t)$ and $f_o(s,t)$ represent the 2-dimensional f_{COHER_COLOR} feature extracted from a S-T region of the processed and original video streams, this feature comparison function is given by:

$$sqrtmanhat(s,t) = \sqrt{\sum_{C_B,C_R}\left|\underline{f}_p(s,t) - \underline{f}_o(s,t)\right|}$$

The Manhattan distance measure seems to be better than the Euclidean distance measure and the square root function is required to linearize the parameter's response to quality changes. Following the mathematical notations in Annex D.8 of ITU-T Recommendation J.144, the colour parameters are given by:

$$color\_extreme = \text{color\_coher\_color\_30x30\_1s\_mean\_sqrtmanhat\_OMB(3,3,2)above99\%\_Minkosski(0.5,1)}$$

$$color\_spread = \text{color\_coher\_color\_30x30\_1s\_mean\_sqrtmanhat\_OMB(3,3,2)Minkosski(2,4)\_90\%}$$

A combined colour parameter (*color_comb*) that contains the optimal combination of both the *color_extreme* and *color_spread* parameters is then computed as:

$$color\_comb = 0.691686 * color\_extreme - 0.617958 * color\_spread$$

This positive valued *color_comb* parameter is then clipped at the low end, which is represented mathematically by (following the notation in Annex D.8.5 of ITU-T Recommendation J.144):

$$color\_comb = color\_comb\_clip\_0.114$$

This *color_comb* parameter is included in the linear combination for the VQM calculation.

3.3.4 Spatial parameters

Two spatial parameters are computed from the f_{SI13} feature, one that measures a loss of spatial information (*si_loss*) and one that measures a gain of spatial information (*si_gain*). Following the mathematical notation in Annex D.8 of ITU-T Recommendation J.144, these parameters are given by:

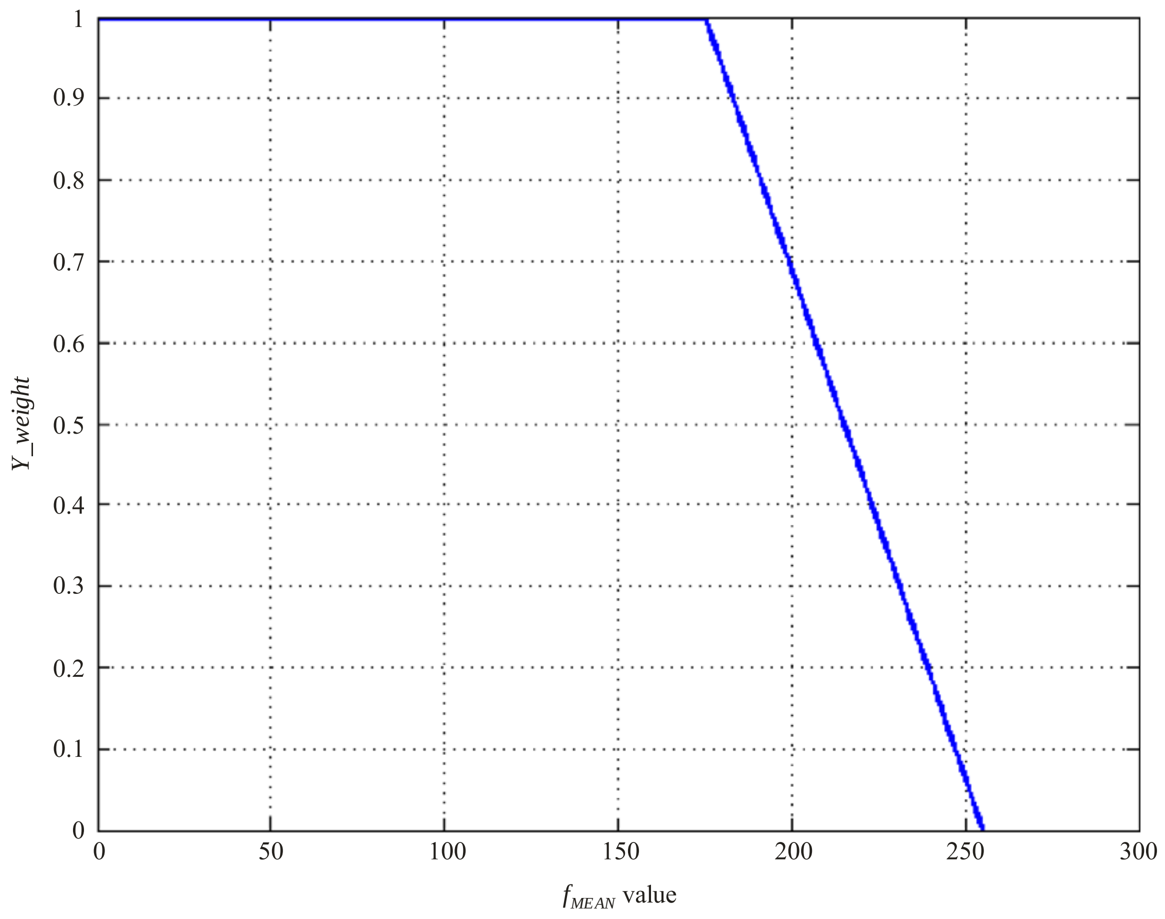
$$si\_loss = \text{avg1s\_Y\_si13\_30x30\_std\_3\_ratio\_loss\_OMB(3,3,2)Minkosski(1,2)\_Minkosski(1.5,2.5)\_clip\_0.12}$$

$$si\_gain = \text{avg1s\_Y\_si13\_30x30\_std\_3\_log\_gain\_clip\_0.1\_above95\%tail\_Minkosski(1.5,2)}$$

As the mean luminance (Y) level of the S-T subregion increases (i.e. as measured by the f_{MEAN} feature), the ability to perceive changes in spatial detail (e.g. such as blurring measured by *si_loss*) decreases. This can be accommodated by introducing a weighting function (*Y_weight*) as shown in Fig. 29 to the *si_loss* values from each S-T subregion (i.e. the *si_loss* values after the ratio loss comparison function is performed on each S-T subregion but before the spatial and temporal collapsing functions). The weighting function *Y_weight* is equal to one (i.e. full weighting) until an average luminance level of 175 is reached and then it decreases linearly to zero as the luminance values increase from 175 to 255. This intermediate correction is applied only to the *si_loss* values, not the *si_gain* values.

FIGURE 29

Weighting function *Y_weight* for modifying *si_loss* S-T parameters



BT.1885-29

Two spatial parameters are computed from the f_{HV13} feature, one that measures a loss of relative Horizontal and Vertical (HV) spatial information (*hv_loss*) and one that measures a gain (*hv_gain*).

Following the mathematical notation in Annex D.8 of ITU-T Recommendation J.144, these parameters are given by:

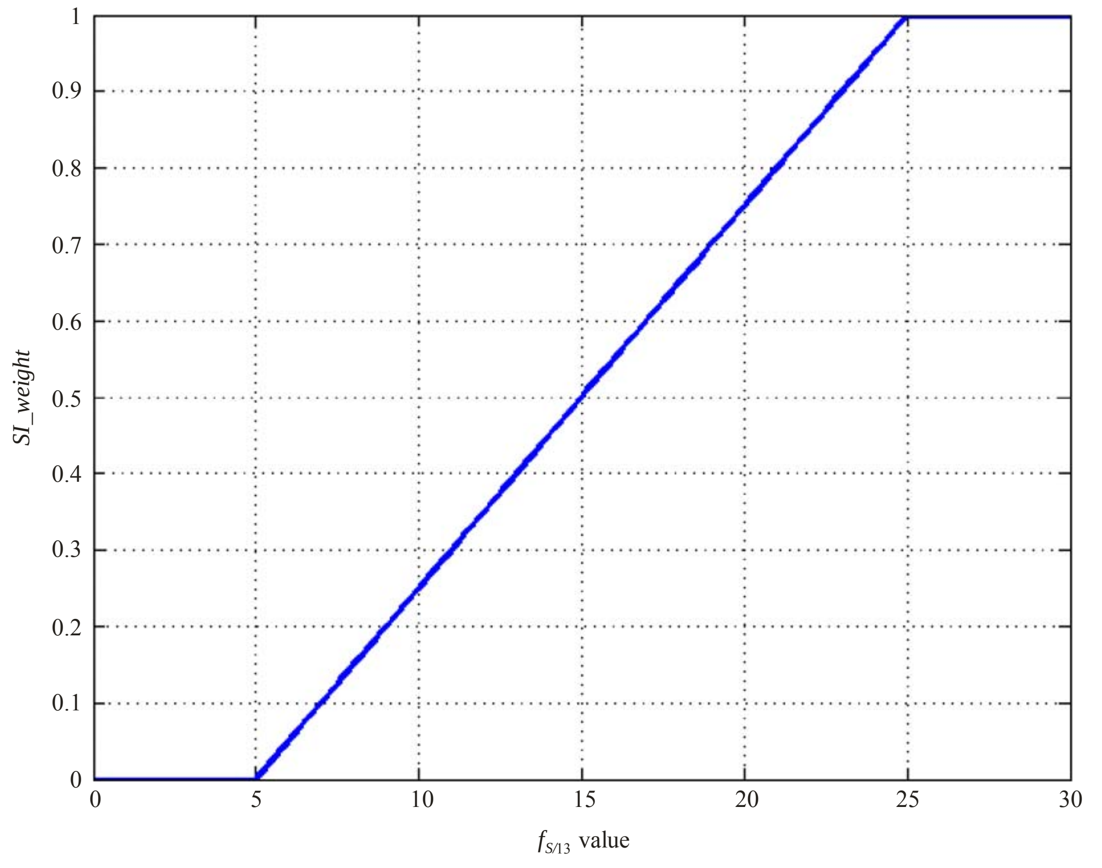
$$hv\_loss = \text{avg1s\_Y\_hv13\_angle0.225\_rmin20\_30x30\_mean\_4\_ratio\_loss\_...}$$
$$\text{OMB(3,3,2)below1\%\_Minkosski(1,1.5)\_clip\_0.08}$$

$$hv\_gain = \text{avg1s\_Y\_hv13\_angle0.225\_rmin20\_30x30\_mean\_4\_log\_gain\_...}$$
$$\text{clip\_0.06\_OMB(3,3,2)above99\%tail\_Minkosski(1.5,3)}$$

Not shown in the above equations is that the *Y_weight* function shown in Fig. 29 is also applied to both the *hv_loss* and *hv_gain* values from each S-T subregion before the spatial and temporal collapsing functions (after the ratio_loss and log_gain computations, respectively). An additional weighting function (*SI_weight* as shown in Fig. 30) is applied to the *hv_loss* values from each S-T subregion. This was necessary to reduce the sensitivity of *hv_loss* for S-T regions that have very little spatial information (i.e. low f_{SI13} original feature values).

FIGURE 30

Weighting function *SI_weight* for modifying *hv_loss* S-T parameters



BT.1885-30

The spatial distortion parameters can be crushed (i.e. excessive excursions beyond those found in the training data are limited or compressed) using functions like the VQM crushing function found in the VQM calculation section.

3.3.5 Temporal parameters

Two temporal parameters are computed from the f_{ATI} feature, one that measures added random noise (*ati_noise*) and one that measures motion disturbances caused by transmission errors (*ati_error*).

Following the mathematical notation in Annex D.8 of ITU-T Recommendation J.144, these parameters are given by:

$$ati\_noise = \text{Y\_rand5\%\_ati0.2s\_rms\_5\_ratio\_gain\_between25\%50\%}$$

$$ati\_error = \text{Y\_rand5\%\_ati0.2s\_rms\_max7pt\_12\_ratio\_gain\_above90\%}$$

To make the *ati_noise* and *ati_error* parameters more robust against temporal misalignments, the parameters are computed for all temporal alignments of the processed video that are within ±0.4 s of the best estimated temporal alignment to the original video, and then the minimum parameter value is selected.

3.4 VQM calculation

Similar to the NTIA General VQM in Annex D of ITU-T Recommendation J.144, the Fast Low Bandwidth VQM calculation linearly combines two parameters from the f_{HV13} feature (*hv_loss* and *hv_gain*), two parameters from the f_{SI13} feature (*si_loss* and *si_gain*), and two parameters from the f_{COHER_COLOR} feature (except that the two parameters have been combined into a single colour distortion parameter called *color_comb*). The one noise parameter in the NTIA General VQM has been replaced with 2 parameters based on the low bandwidth f_{ATI} feature described here (*ati_noise* and *ati_error*).

Thus, VQM_{FLB} (abbreviation for Fast Low Bandwidth VQM) consists of a linear combination of eight parameters. VQM_{FLB} is given by:

$$\begin{aligned} VQM_{FLB} = \{ & 0.38317338378290 * hv\_loss + 0.37313218013131 * hv\_gain + \\ & 0.58033514546526 * si\_loss + 0.95845512360511 * si\_gain + \\ & 1.07581708014998 * color\_comb + \\ & 0.17693274495002 * ati\_noise + 0.02535903906351 * ati\_error \, \} \end{aligned}$$

The total VQM (after the contributions of all the parameters are added up) is clipped at a lower threshold of 0.0 to prevent negative VQM numbers. Finally, a crushing function that allows a maximum of 50% overshoot is applied to VQM values over 1.0 to limit VQM values for excessively distorted video that falls outside the range of the training data.

If $VQM_{FLB} > 1.0$, then $VQM_{FLB} = (1 + c)*VQM_{FLB} / (c + VQM_{FLB})$, where $c = 0.5$.

VQM_{FLB} computed in the above manner will have values greater than or equal to zero and a nominal maximum value of one. VQM_{FLB} may occasionally exceed one for video scenes that are extremely distorted.

To make VQM_{FLB} more robust against spatial misalignments, VQM_{FLB} is computed for all spatial alignments of the processed video that are within plus or minus 1 pixel of the best estimated spatial alignment to the original video, and then the minimum VQM_{FLB} is selected.

4 References

[1] Recommendation ITU-R BT.601-6 (01/07) – Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios.

[2] Video Quality Model (VQM) Software Tools – Binary executables and source code, available from the National Telecommunications and Information Administration (NTIA) at http://www.its.bldrdoc.gov/n3/video/VQM_software.php.

[3] ITU-T Recommendation J.244 (04/08) – Full reference and reduced reference calibration methods for video transmission systems with constant misalignment of spatial and temporal domains with constant gain and offset.

[4] VQEG Final Report of MM Phase I Validation Test (2008), "Final report from the Video Quality Experts Group on the validation of objective models of multimedia quality assessment, phase I", Video Quality Experts Group (VQEG), http://www.its.bldrdoc.gov/vqeg/projects/multimedia, ITU-T Study Group 9 TD923, Study Period 2005-2008.

[5] ITU-T Recommendation J.144 (03/04) – Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference.

[6] Recommendation ITU-R BT.1683 (06/04) – Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference.

5 Reference code for implementing the Fast Low Bandwidth VQM

The purpose of this reference code is to assist the user with proper implementation of the Fast Low Bandwidth VQM. While MATLAB® code is used for the reference code, any software code can be used that reproduces the results given here. Each subsection in § 5 contains MATLAB code for the function named in the section header (e.g. save the contents of § 5.1 to a file called "fastlowbw_ref.m"). Execute fastlowbw_ref with no arguments to receive help information on how to call the routine. This code contains the flexibility of running the model on a short video clip (i.e. 5 to 15 s) within a larger video sequence (e.g. 1-minute sequence). This is done by shifting the short video clip by one second and re-computing the model for each temporal shift. While this functionality is not demonstrated below, comments within the code and returned arguments from "model_fastlowbw_parameters.m" will refer to this capability. This hidden capability may be useful for implementing an in-service video quality monitoring system.

When the sample test vectors (i.e. video clips) are processed with the Fast Low Bandwidth VQM reference code (function "fastlowbw_ref.m"), text files are produced that contain the calibration and model results. For the following example MATLAB function calls, output files similar to those given below should be obtained (due to random processes used by the Fast Low Bandwidth VQM, results may vary slightly from those presented here):

5.Reference
Code.doc

Appendix

Transmission error analyses

The VQEG validation tests for the RRNR-TV project included the 525 (NTSC) and 625 (PAL) formats. Each experiment included 12 source sequences and 156 processed video sequences (PVSs). Of these 156 PVSs, 40 contained transmission errors and 116 contained coding errors only. Tables 9 and 10 show the RMSE and OR for the PVSs with transmission errors. It is noted that the RMSE and OR were computed using the regression lines which were obtained from the entire data. In other words, the regression lines were computed using the entire data. Then, the RMSE and OR for transmission errors were computed using the PVSs with transmission errors.

TABLE 9

RMSE and OR for the RRNR-TV validation test (525 format). TE: transmission errors

525 format	**ALL**		**With TE**		**No TE**	
	RMSE	**OR**	**RMSE**	**OR**	**RMSE**	**OR**
Model_A_15k	0.418	0.385	0.574	0.500	0.362	0.293
Model_A_80k	0.423	0.378	0.582	0.475	0.366	0.293
Model_A_256k	0.424	0.378	0.584	0.475	0.367	0.293
Model_B_80k	0.598	0.667	0.768	0.650	0.544	0.586
Model_B_256k	0.587	0.647	0.763	0.600	0.530	0.578
Model_C_80k	0.465	0.513	0.557	0.550	0.440	0.405
Model_C_256k	0.511	0.609	0.584	0.450	0.495	0.578
PSNR_NTIA	0.556	0.571	0.549	0.500	0.568	0.491

TABLE 10

RMSE and OR for the RRNR-TV validation test (625 format). TE: transmission errors

625 format	**ALL**		**With TE**		**No TE**	
	RMSE	**OR**	**RMSE**	**OR**	**RMSE**	**OR**
Model_A_15k	0.524	0.468	0.597	0.450	0.508	0.414
Model_A_80k	0.513	0.462	0.594	0.500	0.494	0.379
Model_A_256k	0.516	0.468	0.593	0.500	0.499	0.379
Model_B_80k	0.887	0.724	0.545	0.500	0.986	0.716
Model_B_256k	0.864	0.744	0.523	0.600	0.962	0.716
Model_C_80k	0.585	0.583	0.282	0.200	0.663	0.647
Model_C_256k	0.657	0.590	0.292	0.175	0.747	0.638
PSNR_NTIA	0.605	0.564	0.338	0.250	0.678	0.517