International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

J.343.5

(11/2014)

SERIES J: CABLE NETWORKS AND TRANSMISSION OF TELEVISION, SOUND PROGRAMME AND OTHER MULTIMEDIA SIGNALS

Measurement of the quality of service – Part 3

Hybrid-FRe objective perceptual video quality measurement for HDTV and multimedia IP-based video services in the presence of a full reference signal and encrypted bitstream data

Recommendation ITU-T J.343.5

Recommendation ITU-T J.343.5

Hybrid-FRe objective perceptual video quality measurement for HDTV and multimedia IP-based video services in the presence of a full reference signal and encrypted bitstream data

Summary

Recommendation ITU-T J.343.5 provides hybrid full-reference encrypted (Hybrid-FRe) objective perceptual video quality measurement methods for HDTV and multimedia when a full reference signal and encrypted bitstream data are available. The following are example applications that can use this Recommendation:

- potentially real-time, in-service quality monitoring at the headend;
- video television streams over cable/IPTV networks including those transmitted over the Internet using Internet protocol;
- video quality monitoring at the receiver when a full reference signal and encrypted bitstream data are available;
- video quality monitoring at measurement nodes located between point of transmission and point of reception when a full reference signal and encrypted bitstream data are available;
- quality measurement for monitoring of a transmission system that utilizes video compression and decompression techniques, either a single pass or a concatenation of such techniques;
- lab testing of video transmission systems.

This Recommendation includes an electronic attachment containing test vectors, including video sequences, bitstream files and predicted objective model scores.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T J.343.5	2014-11-29	9	11.1002/1000/12320

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

© ITU 2015

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope	1
1.1	Applications	2
1.2	Limitations	2
2	References	3
3	Definitions	3
3.1	Terms defined elsewhere	3
3.2	Terms defined in this Recommendation	3
4	Abbreviations and acronyms	3
5	Conventions	4
6	Performance metrics	4
7	Description of the hybrid full-reference methodology	4
8	Models	5
Annex A – Description of video quality measure PEVQ-S (e)		6
A.1	Introduction	6
A.2	Description of the PEVQ-S (e) algorithm	6
Annex B – YHyFRe		37
B.1	Introduction	37
B.2	Hybrid-FRe VQM computation	37
Bibliography		41

Electronic attachment: Test vectors, including video sequences, bitstream files and predicted objective model scores.

Recommendation ITU-T J.343.5

Hybrid-FRe objective perceptual video quality measurement for HDTV and multimedia IP-based video services in the presence of a full reference signal and encrypted bitstream data

1 Scope

This Recommendation[1] describes algorithmic models for measuring the visual quality of IP-based video services.

The models are hybrid full-reference encrypted (Hybrid-FRe) models, which use the unimpaired reference video signal, packet header information, and video image data captured at the video player. The models operate without parsing or decoding the packet payload. Thus, these models can be used with encrypted bitstream data as well as non-encrypted bitstream data.

As output the models provide an estimate of visual quality on the [1,5] difference mean opinion score (DMOS) scale, derived from five-point absolute category rating (ACR) as in [ITU-T P.910]. The models address low-resolution (VGA/WVGA) application areas, including services such as mobile TV, as well as high-resolution (HD) application areas, including services such as IPTV.

This Recommendation is to be used with videos encoded using ITU-T H.264 and media payload encapsulated in RTP/UDP/IP packets for the low resolution and encapsulated in MPEG-TS/RTP/UDP/IP for the high resolution.

The models in this Recommendation measure the visual effect of spatial and temporal degradations as a result of video coding, erroneous transmission or video rescaling. The models may be used for applications such as to monitor the quality of deployed networks to ensure their operational readiness or to benchmark service quality. The models in this Recommendation can also be used for lab testing of video transmission systems.

The models identified in this Recommendation have limited precision. Therefore, directly comparing model results can be misleading. The accuracy of models has to be understood and taken into account (e.g., using [ITU-T J.149]).

The validation test material consisted of video encoded using different implementations of [ITU-T H.264]. It included media transmitted over wired and wireless networks, such as WIFI and 3G mobile networks. The transmission impairments included error conditions such as dropped packets, packet delay, both from simulations and from transmission over commercially operated networks.

The following source reference channel (SRC) conditions were included in the validation test:

- 1080i 60 Hz (29.97 fps);
- 1080p (25 fps);
- 1080i 50 Hz (25 fps);
- 1080p (29.97 fps);
- SRC duration: HD: 10s, VGA/WVGA: 10 s or 15 s (rebuffering);
- VGA at 25 and 30 fps;
- WVGA at 25 and 30 fps.

[1] This Recommendation includes an electronic attachment containing test vectors, including video sequences, bitstream files and predicted objective model scores.

The following hypothetical reference circuit (HRC) conditions were included in the validation test for each resolution:

Test factors
Video resolution: 1920x1080 interlaced and progressive
Video frame rates 29.97 and 25 fps
Video bitrates: 1 to 30 Mbit/s (HD), 100kbit/s to 3 Mbit/s (VGA/WVGA)
Temporal frame freezing (pausing with skipping) of up to 50% of video duration
Transmission errors with packet loss
Rebuffering (VGQ/WVGA only): up to 50% of SRC
Coding technologies
ITU-T H.264/AVC (MPEG-4 Part 10)
Tandem coding

1.1 Applications

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

- potentially real-time, in-service quality monitoring at the headend;
- video television streams over cable/IPTV networks including those transmitted over the Internet using Internet protocol;
- video quality monitoring at the receiver when a copy of the unimpaired reference video and the encrypted bitstream data are available;
- video quality monitoring at measurement nodes located between point of transmission and point of reception when a copy of the unimpaired reference video and the encrypted bitstream data are available;
- quality measurement for monitoring of transmission systems that utilize video compression and decompression techniques, including concatenations of such techniques;
- lab testing of video transmission systems.

1.2 Limitations

The video quality estimation models described in this Recommendation cannot be used to fully replace subjective testing.

When frame freezing was present, the test conditions had frame freezing durations up to 50% of SRC duration. The models in this Recommendation were validated for measuring video quality in a re-buffering condition (i.e., video that has a steadily increasing delay or freezing without skipping) only for VGA/WVGA. The models were not tested on other frame rates than those used in TV systems (i.e., 29.97 fps and 25 fps, in interlaced or progressive mode).

If forward error correction techniques are employed, the models in this Recommendation may not be used.

It is important that no additional transmission errors occur between the collection point of the bitstream data and the capture point of the processed video sequence (PVS).

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

2 References

The following ITU-T 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.

[ITU-T H.264] Recommendation ITU-T H.264 (2014), *Advanced video coding for generic audiovisual services*.

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

[ITU-T J.342] Recommendation ITU-T J.342 (2011), *Objective multimedia video quality measurement of HDTV for digital cable television in the presence of a reduced reference signal*.

[ITU-T J.343] Recommendation ITU-T J.343 (2014), *Hybrid perceptual bitstream models for objective video quality measurements*.

[ITU-T P.910] Recommendation ITU-T P.910 (2008), *Subjective video quality assessment methods for multimedia applications*.

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

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 hybrid full reference model [ITU-T J.343]: An objective video quality model that predicts subjective quality using the reference video, the decoded video frames, packet headers, and the video payload. Such models cannot analyse encrypted video.

3.1.2 hybrid full reference encrypted model [ITU-T J.343]: An objective video quality model that predicts subjective quality using the reference video, the decoded video frames, and packet headers. Such models are suitable for use with encrypted video.

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CODEC	Coder-Decoder
DMOS	Difference Mean Opinion Score
DTS	Decoding Time Stamp
FR	Full Reference
HRC	Hypothetical Reference Circuit
HVS	Human Visual System

Hybrid-FR	Hybrid Full Reference
Hybrid-FRe	Hybrid Full Reference Encrypted
LUT	Look-Up Table
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
PES	Packetized Elementary bitStream
PEVQ-S	Perceptual Evaluation of Video Quality for Streaming
PTS	Presentation Time Stamp
PVS	Processed Video Sequence
ROI	Region Of Interest
SEQ	Sequence index
SRC	Source Reference Channel (or Circuit)
TS	Transport Stream
VQEG	Video Quality Experts Group
VQM	Video Quality Metrics

5 Conventions

None.

6 Performance metrics

A summary of this and other hybrid models may be found in [ITU-T J.343]. See [b-VQEG Hybrid] for the complete analysis of the models included in this Recommendation.

7 Description of the hybrid full-reference methodology

This Recommendation specifies objective video quality measurement methods that use both processed video sequences and bitstream data. The bitstream data may be provided in the forms of elementary bitstream (ES), packetized elementary bitstream (PES) or packet video (Figure 1).

In addition, the Hybrid-FRe models also use reference video sequences. Figure 2 shows a Hybrid-FRe model. The Hybrid-FRe models need the source video sequence (SRC). While the Hybrid-FR models have access to all of this data, the Hybrid-FRe models do not have access to the video payload. Therefore, these models can be used with encrypted bitstreams.

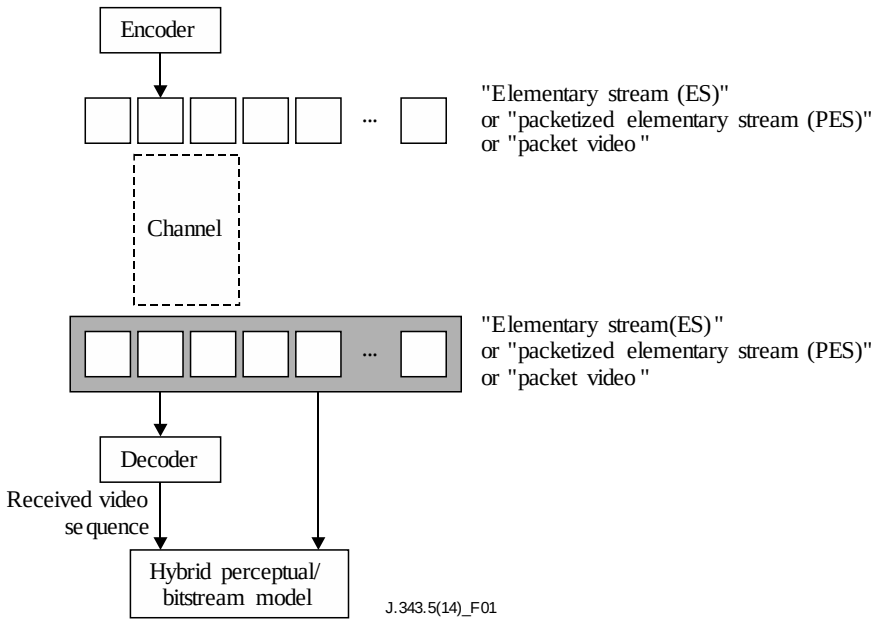
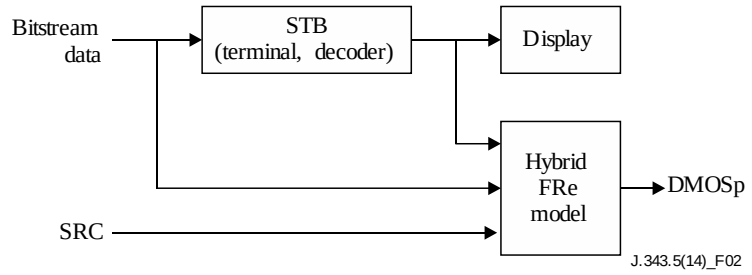


Figure 1 – Block-diagram depicts the core concept of hybrid perceptual bitstream models



DMOSp: predicted DMOS by the model

Figure 2 – Block-diagram depicts the Hybrid-FRe model

8 Models

Annexes A and B contain full disclosures of all models included in this Recommendation. These models are perceptual evaluation of video quality for streaming (PEVQ-S) (e) and YHyFRe.

Annex A

Description of video quality measure PEVQ-S (e)

(This annex forms an integral part of this Recommendation.)

A.1 Introduction

PEVQ-S (e) is a very robust model that is designed to predict the effects of transmission impairments on the video quality as perceived by a human subject. Its main targets are mobile applications and multimedia applications. The key features of PEVQ-S (e) are:

- temporal alignment of the input sequences based on multi-dimensional feature correlation analysis with limits that reach far beyond those tested by VQEG, especially with regard to the amount of time clipping, frame freezing and frame skipping which can be handled;
- full-frame spatial alignment;
- colour alignment algorithm based on cumulative histograms;
- detection and perceptually correct weighting of frame freezes and frame skips;
- perceptual estimation of degradations;
- degradation caused by packet loss and bit errors in the transmission stream;
- handling of encrypted and non-encrypted bitstreams.

Only five indicators are used to detect the video quality. Those indicators are motivated by the human visual system (HVS). Perceptual masking properties of the HVS are modelled at several stages of the algorithm. These indicators are integrated using a sophisticated spatial and temporal integration algorithm.

In its first stage the algorithm collects information required for the alignment, concerning monochrome and frozen or skipped frames. In the second step the frames are synchronized in the temporal, spatial and colour domain, followed by comparison for visual differences in the luminance as well as in the chrominance domain and taking masking and motion effects into account. In addition to the image comparison, the transmitted bitstream is analysed for transmission errors. The above steps results in a set of indicators where each indicator describes a certain quality aspect.

The last step is finally the integration of the indicators by non-linear functions in order to derive the final mean opinion score (MOS).

Due to the low number of indicators and the resulting low degree of freedom the model can hardly be over trained and is very robust. PEVQ-S (e) can be efficiently implemented without sacrificing the accuracy.

A.2 Description of the PEVQ-S (e) algorithm

A.2.1 Overview

The basic idea of the PEVQ-S (e) model is given in Figure A.1. Since PEVQ-S (e) is a full reference hybrid algorithm, it requires three input signals. The first must be the undistorted source reference channel (SRC), S, the second must be the PVS, P, and third must be the collected bitstream file. It is assumed that the PVS is the result of the processed SRC through an HRC, and the bitstream file is collected at the player side.

Both sequences must have the same resolution (e.g., VGA, HD 1080p, HD 1080i). The signals should follow the YCbCr colour representation, as defined in the [ITU-R BT.601].

The data storage for the images is planar oriented; no down sampling is used (sampling 4:4:4). Plane 1 is the Y component, Plane 2 the Cb component and Plane 3 the Cr component. The pixel values are stored as floating-point data.

Within the analysis two main modules can be found. A bitstream analysis module and a pixel based analysis module. Within the bitstream analysis module the bitstream is analysed for packet loss resulting in one indicator representative for additional temporal and spatial distortions.

Within the pixel-based analysis module uncompressed reference and degraded video sequences are used for detailed distortion analysis. The outcome of this analysis is indicators describing the severity of distortions in the temporal luminance and chrominance domain.

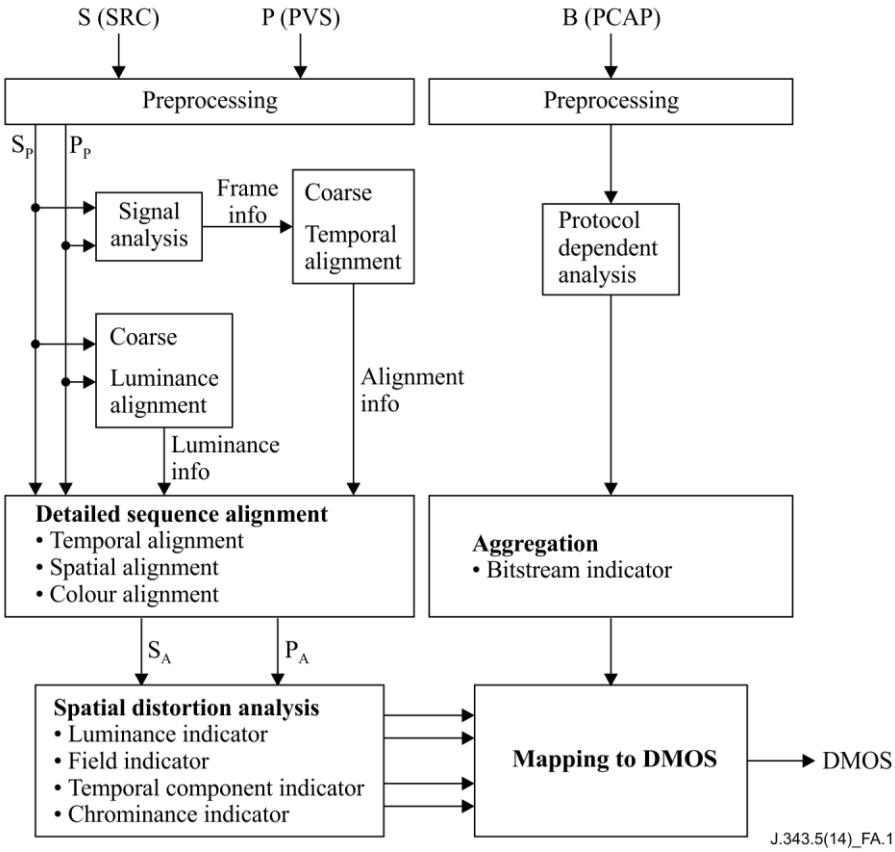


Figure A.1 – Overview of PEVQ-S (e)

In the last step "Mapping to DMOS" in Figure A.1 of PEVQ-S (e) the five indicators that were derived above are weighted by logistic functions and combined to form the final PEVQ-S (e) score, which correlates highly with a MOS obtained from the subjective tests.

A.2.2 Bitstream analysis

The task of the bitstream analysis is to produce an indicator that reflects the distortion in the video signal caused by packet loss (PL) during the transmission. This indicator is used within the OPTICOM model, together with the indicators from the pixel analysis, to predict the overall video quality of a given video stream.

The input into the bitstream analysis module is the bitstream B, which first is segmented in a list of packets $P(i)$, $i \in [0..I-1]$, where I denotes the total number of packets arrived at the measurement point. This means the arrival times of the packets increase with increasing i.

The bitstream analysis module supports ITU-T H.264 as video codec and two different protocol stacks:

- protocol stack S_1: Ethernet (ETH)/IPv4/UDP/RTP;
- protocol stack S_2: Ethernet (ETH)/IPv4/UDP/RTP/MPEG2-TS.

This means the packets $P(i)$ containing the video data have the structure shown in Figure A.2 (a) and (b) for stack S_1 and S_2 respectively.

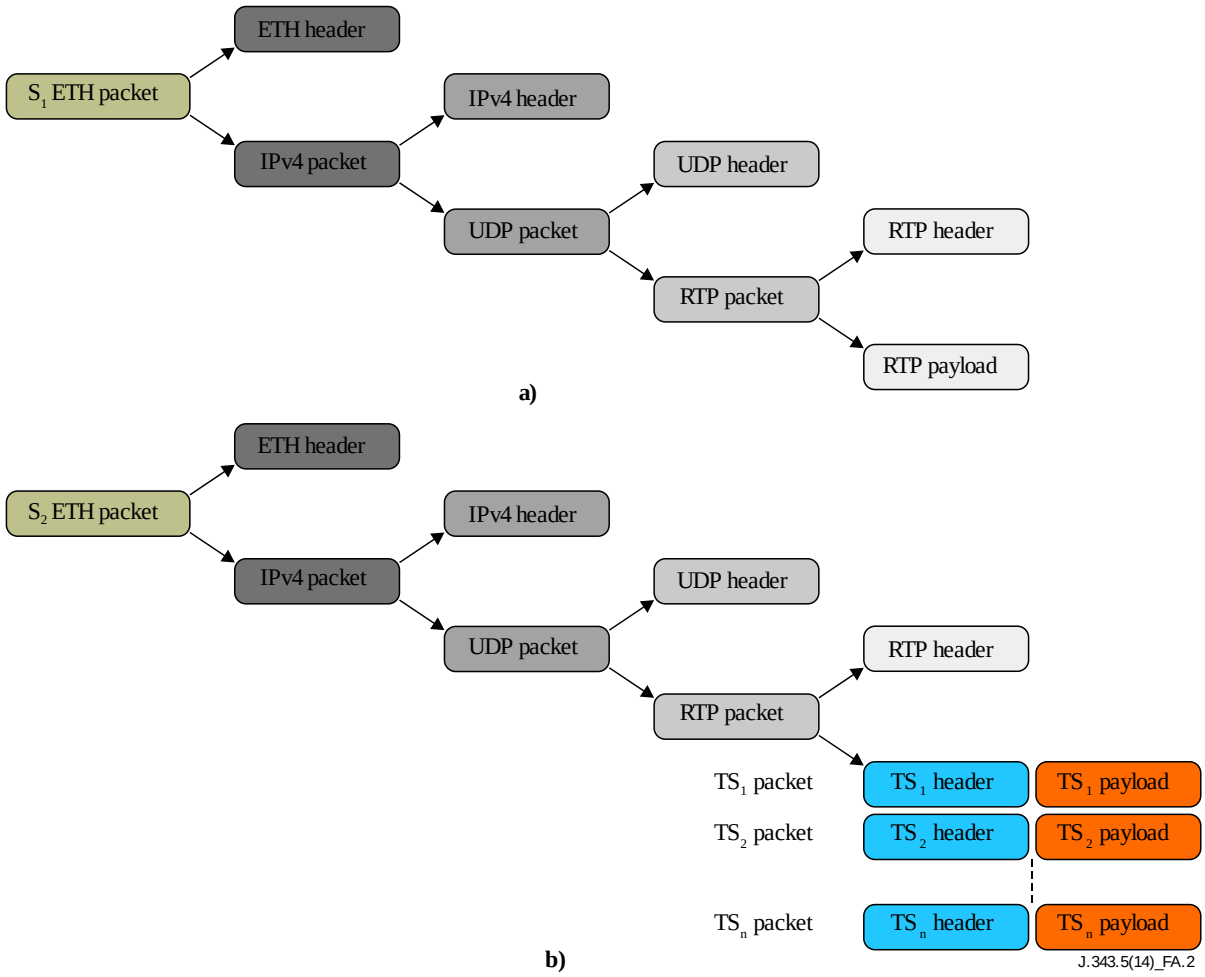


Figure A.2 – Payload structure with the Ethernet packets (a) Protocol stack S_1 (b) Protocol stack S_2

The bitstream analysis can handle specific bitstream encryption cases. For stack S_1 the RTP payload may be encrypted, which means that the RTP header is still accessible to the model.

For stack S_2, two encryption modes are supported: transport stream (TS) payload encryption and PES payload encryption. For PES payload encryption, the PES headers are accessible to the model; for TS payload encryption, only the RTP and TS headers are available to the model.

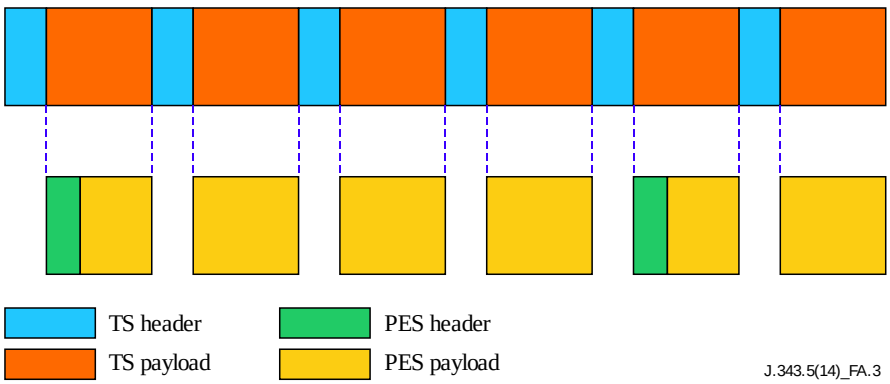


Figure A.3 – PES payload arrangement within one or more TS packets in Protocol Stack S_2

An audio stream may be present in the bitstream but is not taken into account for the prediction of the perceived video quality.

A.2.2.1 Bitstream analysis module

The bitstream analysis module runs through the following three main functions sequentially:

1) pre-processing (common for both supported protocol stacks);
2) protocol dependent bitstream analysis;
3) aggregation of results from previous steps to a single indicator.

A.2.2.1.1 Pre-processing

The first step of the bitstream analysis is the pre-processing of the received packets. As depicted by the flow diagram in Figure A.4, the pre-processing of the received bitstream is done in two phases, which are explained below.

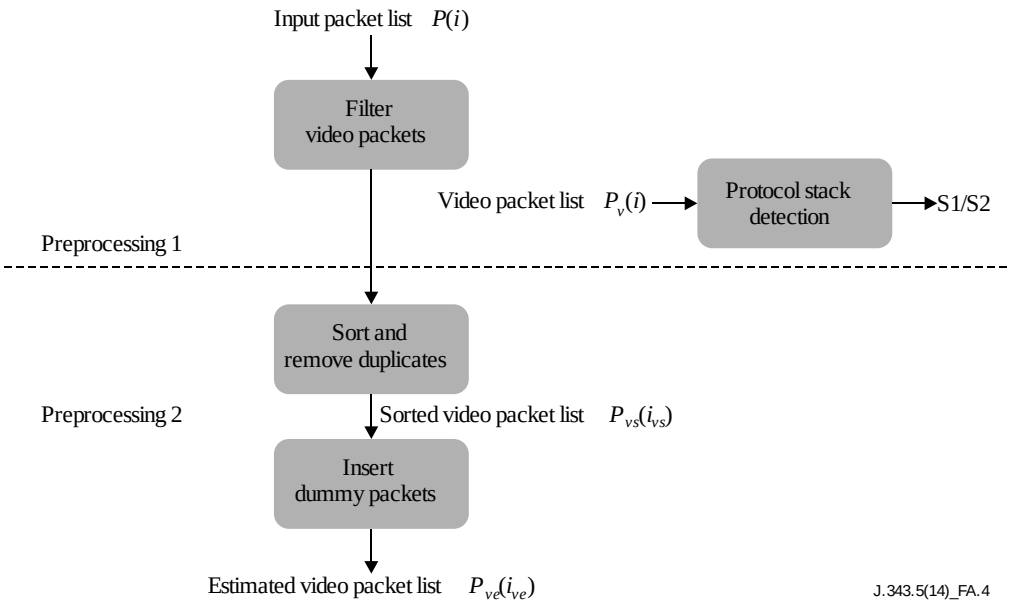


Figure A.4 – Flow diagram of the bitstream pre-processing step

A.2.2.1.2 Filtering and protocol stack detection

The input stream may contain packets that are irrelevant for the analysis and therefore need to be removed. Packets are deemed relevant for the analysis if they belong to the RTP stream containing the video packets.

In general, video packets represent the majority of the input stream. Therefore, the UDP port of the RTP video stream at the client side $clientUdpVideoPort$ is equal to the destination UDP port that is present in the highest number of packets in the input stream.

Consequently, the input stream is filtered for UDP packets with destination port $clientUdpVideoPort$. The list of packets that fulfills these requirements is called $P_v(i_v)$ with $i_v \in [0..\mathrm{I_v} - 1]$; all other packets are disregarded for further analysis.

Based on the video stream packets P_v the complete protocol stack is determined by probing the RTP packets. If the RTP payload consists of MPEG2-TS packets the protocol stack of the video stream is considered to be S_2 otherwise it is S_1.

A RTP packet is classified as carrying MPEG2-TS packets if size of the payload of the RTP packet is a multiple of 188 bytes and if every potential TS packet in the RTP payload starts with a byte value of 0x47.

A.2.2.1.3 RTP SEQ unwrapping and packet reordering

As the packets may have arrived out of order due to travelling over different network paths, the next step restores the sending order of the packets.

For S_1 the RTP header carries a sequence index (SEQ) field for this purpose. The SEQ field is a counter that is increased for every sent RTP packet. Therefore, the sending order can be restored by sorting the packets by ascending SEQ number.

The word length of the SEQ field is 16 bits; this means the highest RTP SEQ number that can be represented is 65'535. As the counter overflows if the stream contains more packets the SEQ number must be unwrapped prior to sorting.

After unwrapping the SEQ number and sorting the packets, the sending order is restored. It is possible that packets got duplicated while travelling over the network. These duplicates can be found by searching for packets with the same SEQ number. Removing the duplicates yields the list of ordered, unique packets $P_{vs}(i_{vs})$.

If no packet loss has occurred while transmitting the video stream the SEQ number of each packet $P_{vs}(x)$ fulfills:

$$P_{vs}(x).RTP_Header.SEQ == P_{vs}(x-1).RTP_Header.SEQ + 1 \qquad \text{(A.2-1)}$$

Otherwise one or more packets have been lost during transmission. The number of lost packets $PL(x)$ between two packets $P_{vs}(x)$ and $P_{vs}(x+1)$ can be determined through comparison of their SEQ numbers.

$$PL(x) = P_{vs}(x+1).RTP_Header.SEQ - P_{vs}(x).RTP_Header.SEQ - 1 \qquad \text{(A.2-2)}$$

For every lost packet a dummy packet is inserted into $P_{vs}(i_{vs})$ which yields the estimated list of video packets $P_{ve}(i_{ve})$ with $i_{ve} \in [0..I_{ve}-1]$ that have been sent by the streaming server.

The video packet lists P_{vs} and P_{ve} will be used in the following steps to extract features from which the MOV of the bitstream module can be created.

A.2.2.2 Protocol dependent analysis

The input to the protocol dependent analysis is the packet lists P_{vs} and P_{ve}. The result is the video frame rate and the number of video frames in the original streamed video (i.e., $nrOrigFrames$).

A.2.2.2.1 Specific analysis for stack S1

It is assumed that in case of protocol stack S_1 the time stamp in the RTP header can be set according to two schemes:

- decoding time stamp (DTS);
- presentation time stamp (PTS).

In both cases the RTP time stamp is the same for all packets carrying bytes of the same coded picture. For the DTS scheme the timestamp represents a clock that runs at 90 kHz and reflects the relative time when a coded picture needs to be decoded. For the PTS scheme the timestamp represents a clock that runs at 90 kHz and reflects the relative time when a coded picture needs to be displayed on the screen.

As the coded pictures are stored in decoding order in the bitstream, the RTP timestamp is monotonically increasing with the coded picture number for DTS scheme.

The table below gives an example for DTS and PTS scheme for a video with a frame rate of 30 fps.

Coded Pic Type	I	P	B	P	B
Coded Pic Number	0	1	2	3	4
RTP timestamp (DTS)	0	3'000	6'000	9'000	12'000
RTP timestamp (PTS)	0	6'000	3'000	12'000	9'000

This means it is possible to derive the video frame rate $videoFps$ from the minimal absolute RTP time stamp difference of packets of two consecutive frames, see below:

$$videoFps = \frac{90'000}{minAbsTimeStampDiff} \left[\frac{1}{s}\right] \tag{A.2-3}$$

To increase the robustness, the calculation of $minAbsTimeStampDiff$ is carried out on sections $SEC_{x,y}$ of the packet list P_{vs}. The subscripts x and y refer to the first and the last packet number of the packets belonging to a respective section.

The sections $SEC_{x,y}$ are derived in such a way that all packets $P_{vs}(z)$ with $z \in [x, x+1, .., y]$ fulfill equation (A.2-1). This means that every section contains packets without packet loss in between.

The three sections that contain the most packets, i.e., where $y - x$ is largest, are analysed to gather information about the RTP time stamp scheme. The differences in RTP time stamps between consecutive packets inside $SEC_{x,y}$ are calculated to get $minAbsTimeStampDiff$ and are also used to build a histogram of the time stamp gradients. The histogram is not necessary for the creation of the MOV but for additional information about the analysed bitstream.

Algorithmic description

```
minAbsTimeStampDiff = 99999999
negGrad = 0
zeroGrad = 0
posGrad = 0
for p in [x+1..y]:
       timeStampDiff = P_vs(p).RTP_Header.TimeStamp - P_vs(p-1).RTP_Header.TimeStamp
       if timeStampDiff == 0:
              zeroGrad + = 1
       if timeStampDiff > 0:
              posGrad += 1
              if minAbsTimeStampDiff > timeStampDiff:
                     minAbsTimeStampDiff = timeStampDiff
       if timeStampDiff < 0:
              negGrad += 1
              if minAbsTimeStampDiff > -timeStampDiff:
                     minAbsTimeStampDiff = -timeStampDiff

if negGrad >= 2:
       timeStampScheme = PTS

else:

       timeStampScheme = DTS
```

The threshold for detected PTS scheme is two instead of zero for robustness reasons.

Using the minimum time stamp difference and the RTP time stamp of the first and the last received packet it is possible to estimate the number of frames $nrOrigFrames$ in the *sent* video stream.

$$\begin{aligned} nrOrigFrame = (&P_{vs}(I_{vs} - 1).RTP_Header.TimeStamp \\ &- P_{vs}(0).RTP_Header.TimeStamp) \,/\, minAbsTimeStampDiff + 1 \end{aligned} \tag{A.2-4}$$

The described analysis for protocol stack S_1 is exactly the same for non-encrypted and RTP payload encrypted bitstreams.

A.2.2.2.2 Specific analysis for stack S_2

In case of TS-level encryption the PES header information is not accessible by the bitstream analysis module. As a fallback solution the algorithm assumes a fixed duration of 14 s and a frame rate of 25 fps.

A.2.2.3 Aggregation of results/generation of bitstream indicator

In the last step, the detected packet loss is assigned to the coded video frames. Therefore, a bitstream damage over time vector $DMG(f)$ with $f \in [0..F-1]$ where $F = nrOrigFrames$ is created. This vector only contains the values zero and one where zero means the respective frame was not damaged by packet loss and one means the frame was damaged by the packet loss.

The association of packets to frames is only estimated by assuming equal number packets per frames. This is of course not optimal but seems to give a good enough estimation.

Algorithmic description:

```
for i in [0..I_ve-1]:
      DMG(f) = 0

nrPacketsPerFrame = I_ve / nrOrigFrames

for i in [0..I_ve-1]:
      if P_ve(i).IsDummy:
            f = floor(i / nrPacketsPerFrame)
            DMG(f) = 1
```

To simulate the effect of error propagation from one damaged frame to other frames in the decoded video due to inter-frame dependencies, the damage over time vector is convoluted with a non-symmetric window function. The window size depends on the video frame rate to simulate error propagation on time basis instead of frame basis.

$$propWinSize = ceil(videoFps * 0.5) \tag{A.2-5}$$

The window $PropWin(w)$ with $w \in [0..propWinSize - 1]$ is defined as:

$$PropWin(w) = 1 - \frac{w}{propWinSize} \tag{A.2-6}$$

The damage over time vector is convoluted with the window to spread out single packet loss events over multiple consecutive frames in the bitstream. The elements of resulting vector are clipped to a maximum value of 1.0 to limit the effect of packet loss in multiple consecutive frames.

Additionally, the vector is weighted with a window $WeightWin(f)$ to reduce the influence of packet loss at the very beginning and very end of the video stream. The idea is that packet loss shortly after start and just before end of the video does not impact the quality as much as in the rest of the video stream.

$$weightWinSize = floor(videoFps * 0.5 + 0.5) \tag{A.2-7}$$

$$WeightWin(f) = \begin{cases} 1 - \left(\frac{f - weightWinSize}{weightWinSize}\right)^2 & if \quad f \in [0..weightWinSize - 1] \\ 1 - \left(\frac{f + weightWinSize - F + 1)}{weightWinSize}\right)^2 & if \quad f \in [F - weightWinSize..F - 1] \\ 1 & else \end{cases} \tag{A.2-8}$$

The bitstream indicator from the bitstream analysis module is calculated as following:

$$BitstreamIndicator = \frac{1}{F}\sum_{f=0}^{F-1} DMG(f) * WeightWin(f) \quad \text{(A.2-9)}$$

A.2.3 Pixel based video stream analysis

The task of the pixel-based analysis is to produce indicators that reflect the distortion in the video signal. This indicator is used within the PEVQ-S (e) model, together with the indicators from the bitstream analysis, to predict the overall video quality of a given video stream.

As input the pixel based analysis module uses the uncompressed reference and degraded video sequences for a detailed distortion analysis. In the pre-processing step of the module a spatial region of interest (ROI) is extracted from the reference and test signal. All subsequent calculations are performed on this ROI only which is represented by the cropped signals S_P and P_P. The pre-processing is followed by a coarse alignment (registration) of the input sequences in the temporal and luminance domain. The "luminance and alignment information", obtained by these modules is used in the subsequent "detailed sequence alignment" process which performs the temporal alignment frame by frame of the two video sequences, a compensation for spatial shifts, and a compensation for differences in colour and brightness based on histogram evaluations. The results of the "detailed sequence alignment" are the "matching info", which are used to determine the perceptual impact of temporal degradations, as well as the cropped and aligned sequences S_A and P_A.

The spatial distortions are further analysed by the "spatial distortion analysis" block, which calculates the perceptual differences between the sequences in the spatial domain, resulting in four distortion indicators.

A.2.3.1 Size conversion

To minimize the processing time some of the operations are processed on reduced image size. Size reduction is done using cubic interpolation algorithm.

A.2.3.2 Pre-processing

Distortions nearest to the border are often ignored or not really registered by viewers. Therefore, the input sequences are cropped to a region of interest calculated by:

$$S_P[i,j,t] = S[i+c, y+c, t] \quad \forall\, i \in [0..W-1-2c],\ j \in [0..H-1-2c] \quad \text{(A.2-10)}$$

$$P_P[i,j,t] = P[i+c, y+c, t] \quad \forall\, i \in [0..W-1-2c],\ j \in [0..H-1-2c] \quad \text{(A.2-11)}$$

Where W is the width of the input sequence S and H is the height of the input sequence S.

The border value c depends on the size of the input sequence for HD sequences.

c=30, VGA sequences c=12

A.2.3.3 Signal analysis

The task of the signal analysis part consists of the extraction of information and characterisation of the processed signals. Processing is done on a resized image. Resize factor is 0.5 for HD sequences and 1 for VGA sequences.

To avoid side effects, a border around the image of 10 pixels is removed:

$$C_s[i,j,t] = S_p[i+10, y+10, t] \quad \forall\, i \in [0..W_{sp}-21],\ j \in [0..H_{sp}-21] \quad \text{(A.2-12)}$$

$$C_P[i,j,t] = P_P[i+10, y+10, t] \quad \forall\, i \in [0..W_{pp}-21],\ j \in [0..H_{pp}-21] \quad \text{(A.2-13)}$$

In the next step, six different aspects of the spatial domain are extracted from the reference and degraded signal. Later stored in the matrix D_s respective D_p.

These are:

- mean and standard deviations of the cropped signal:

$$D_S[0,t] = \frac{1}{W_{cp} H_{cs}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} C_s[i,j,t] \tag{A.2-14}$$

$$D_p[0,t] = \frac{1}{W_{cp} H_{cp}} \sum_{i=0}^{W_{cp}-1} \sum_{j=0}^{H_{cp}-1} C_p[i,j,t] \tag{A.2-15}$$

$$D_s[1,t] = \sqrt{\frac{1}{W_{cs} H_{cs}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} \left(C_s[i,j,t] - D_s[0,t]\right)^2} \tag{A.2-16}$$

$$D_p[1,t] = \sqrt{\frac{1}{W_{cp} H_{cp}} \sum_{i=0}^{W_{cp}-1} \sum_{j=0}^{H_{cp}-1} \left(C_p[i,j,t] - D_P[0,t]\right)^2} \tag{A.2-17}$$

- mean and standard deviations of the difference between the cropped signal and the cropped signal shifted by three pixels in the horizontal and vertical direction:

$$D_S[2,t] = \frac{1}{(W_{cp}-3)(H_{cs}-3)} \sum_{i=0}^{W_{cs}-4} \sum_{j=0}^{H_{cs}-4} \left(C_s[i,j,t] - C_s[i+3,j+3,t]\right) \tag{A.2-18}$$

$$D_P[2,t] = \frac{1}{(W_{cp}-3)(H_{cs}-3)} \sum_{i=0}^{W_{cs}-4} \sum_{j=0}^{H_{cs}-4} \left(C_P[i,j,t] - C_P[i+3,j+3,t]\right) \tag{A.2-19}$$

$$D_s[3,t] = \sqrt{\frac{1}{(W_{cs}-3)(H_{cs}-3)} \sum_{i=0}^{W_{cs}-4} \sum_{j=0}^{H_{cs}-4} \left(C_s[i,j,t] - C_s[i+3,j+3,t] - D_s[2,t]\right)^2} \tag{A.2-20}$$

$$D_p[3,t] = \sqrt{\frac{1}{(W_{cs}-3)(H_{cs}-3)} \sum_{i=0}^{W_{cp}-4} \sum_{j=0}^{H_{cp}-4} \left(C_p[i,j,t] - C_p[i+3,j+3,t] - D_P[2,t]\right)^2} \tag{A.2-21}$$

For the aspects 4 and 5, edge images $E_s[t]$ and $E_p[t]$ are created by filtering the cropped images according to the following rule:

$$E_s[t] = \sqrt{\left(C_s[t] * K_h\right)^2 + \left(C_s[t] * K_v\right)^2} \tag{A.2-22}$$

$$E_p[t] = \sqrt{\left(C_p[t] * K_h\right)^2 + \left(C_p[t] * K_v\right)^2} \tag{A.2-23}$$

Where K_h and K_v are horizontal and vertical Sobel filter kernels:

$$K_h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad K_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{A.2-24}$$

- the mean and standard deviation of the edge images $E_s[t]$ and $E_p[t]$ are chosen as informational aspects.

$$D_S[4,t] = \frac{1}{W_{cp} H_{cs}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} E_s[i,j,t] \tag{A.2-25}$$

$$D_p[4,t] = \frac{1}{W_{cp} H_{cp}} \sum_{i=0}^{W_{cp}-1} \sum_{j=0}^{H_{cp}-1} E_p[i,j,t] \tag{A.2-26}$$

$$D_s[5,t] = \sqrt{\frac{1}{W_{cs}H_{cs}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} \left(E_s[i,j,t] - D_s[4,t]\right)^2} \quad \text{(A.2-27)}$$

$$D_p[5,t] = \sqrt{\frac{1}{W_{cp}H_{cp}} \sum_{i=0}^{W_{cp}-1} \sum_{j=0}^{H_{cp}-1} \left(E_p[i,j,t] - D_P[4,t]\right)^2} \quad \text{(A.2-28)}$$

- in the time domain the standard deviation of the difference between two consecutive frames is used in the later functions:

$$\overline{T_S}[t] = \frac{1}{W_{cs}H_{cs}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} \left(C_s[i,j,t] - C_s[i,j,t-1]\right) \quad \text{(A.2-29)}$$

$$\overline{T_P}[t] = \frac{1}{W_{cp}H_{cp}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} \left(C_P[i,j,t] - C_P[i,j,t-1]\right) \quad \text{(A.2-30)}$$

$$T_s[t] = \sqrt{\frac{1}{W_{cs}H_{cs}} \sum_{i=0}^{W_{cs}-1} \sum_{j=0}^{H_{cs}-1} \left(C_s[i,j,t] - C_s[i,j,t-1] - \overline{T_s}[t]\right)^2} \quad \text{(A.2-31)}$$

$$T_p[t] = \sqrt{\frac{1}{W_{cp}H_{cp}} \sum_{i=0}^{W_{cp}-1} \sum_{j=0}^{H_{cp}-1} \left(C_p[i,j,t] - C_P[i,j,t-1] - \overline{T_P}[t]\right)^2} \quad \text{(A.2-32)}$$

A.2.3.4 Monochrome estimate

A.2.3.4.1 Definition of the monochrome estimate vectors

The monochrome estimate is based on the spatial aspects and carried out on reference and test signals separately.

$$ME_s[t] = \begin{cases} 0 & \text{if} \quad D_s[1,t] > 0.1 \cdot mean\left(D_s[1,t]\right) \\ ME_s[t-1]+1 & \text{else} \end{cases} \quad \text{(A.2-33)}$$
$$ME_s[0] = 0$$

$$ME_p[t] = \begin{cases} 0 & \text{if} \quad D_p[1,t] > 0.1 \cdot mean\left(D_p[1,t]\right) \\ ME_p[t-1]+1 & \text{else} \end{cases} \quad \text{(A.2-34)}$$
$$ME_p[0] = 0$$

A.2.3.4.2 Maximum number consecutive monochrome elements

The function *MaxMonochromElements(ME,*t_1*,*t_2*)* returns the maximum number of consecutive nonzero elements of $ME[t]$ which are between t_1 and t_2

A.2.3.5 Fluidity estimate

A.2.3.5.1 Definition of the fluidity estimate vectors

The fluidity estimate is based on the temporal aspects and carried out on reference and test signals separately.

$$FE_s[t] = \begin{cases} 0 & \text{if} \quad D_s[1,t]_s > 0.1 \cdot mean\left(T_s[t]\right) \\ FE_s[t-1]+1 & \text{else} \end{cases} \quad \text{(A.2-35)}$$
$$FE_s[0] = 0$$

$$FE_p[t] = \begin{cases} 0 & if \quad T_p[t] > 0.1 \cdot mean(T_p[t]) \\ FE_p[t-1]+1 & else \end{cases}$$
$$FE_p[0] = 0 \tag{A.2-36}$$

A.2.3.5.2 Maximum number consecutive fluidity elements

The function *MaxFluidityElements*(FE,t_1,t_2) returns the maximum number of consecutive nonzero elements of $FE[t]$ which are between t_1 and t_2.

A.2.3.6 Definition of the distortion map

The distortion map is a combination of the fluidity estimate vector and monochrome vector.

$$DE_s[t] = \begin{cases} 0 & if \quad ME_s[t] = 0 \quad and \quad FE_s[t] = 0 \\ DE_s[t-1]+1 & else \end{cases}$$
$$DE_s[0] = 0 \tag{A.2-37}$$

$$DE_p[t] = \begin{cases} 0 & if \quad ME_p[t] = 0 \quad and \quad FE_p[t] = 0 \\ DE_p[t-1]+1 & else \end{cases}$$
$$DE_p[0] = 0 \tag{A.2-38}$$

A.2.3.6.1 Maximum number consecutive elements in distortion map

The function *MaxDistortionElements*(DE,t_1,t_2) returns the maximum number of consecutive nonzero elements of $DE[t]$ which are between t_1 and t_2.

A.2.3.6.2 Segment of interest

A segment of interest α is defined as the region of frames $t_\alpha < t_{\alpha+1}$ with following characteristics.

$$SI(\alpha) \in \begin{cases} MaxDistortionElements(DE_p, t_\alpha, t_{\alpha+1}) \leq 8 \\ DE_p[t_\alpha] = 0 \\ DE_p[t_{\alpha+1}] = 0 \\ t_{\alpha+1} - t_\alpha \leq \dfrac{(Nr\ of\ elements\ in\ DE_p)}{2} \end{cases} \tag{A.2-39}$$

A.2.3.7 Coarse alignment

An important part of a full reference algorithm is the temporal registration of the received video sequence to the given reference sequence. The temporal alignment may be corrupted by monochrome frames, delay and frame skipping, or repetition. In PEVQ-S (e) this alignment procedure is done in two steps. In the first step a coarse alignment together with an estimate of the accuracy of the coarse alignment results is done, followed by a fine alignment module.

PEVQ-S (e) uses a top down strategy based on finding an optimum cross correlation between the spatial information aspects.

A.2.3.7.1 Similarity measure

Let $X[t]$ and $Y[t]$ be two vectors with the length nx and ny>=nx

Then a normalized product-moment correlation vector is given by:

$$\tilde{r}_{xy}[n] = \frac{1}{\sqrt{\sum_{t=0}^{nx-1}\left(X[t]-\bar{\chi}\right)^2 \sum_{t=0}^{ny-1}\left(Y[t+n]-\overline{\xi[n]}\right)^2}} \sum_{t=0}^{nx-1}\left(X[t]-\bar{\chi}\right)\left(Y[t+n]-\overline{\xi[n]}\right) \tag{A.2-40}$$

with:

$$\bar{\chi} = \frac{1}{nx}\sum_{t=0}^{nx-1} X[t]$$

$$\bar{\xi}[n] = \frac{1}{xn}\sum_{t=0}^{xn-1} Y[t+n]$$

and

$$n \in (0..ny - nx)$$

Two or more product-moment correlation vectors of the same size may be combined to a new product-moment correlation vector calculating element wise average.

$$\bar{r}[n] = \frac{1}{i}\sum_{t=0}^{i-1} \tilde{r}_{x_i y_i}[n] \qquad \text{(A.2-41)}$$

The index of the maximum value within the product-moment correlation vector is defined as

$$corrx(X[t], Y[t]) = \arg\max_{n}(\tilde{r}_{xy}[n]) \qquad \text{(A.2-42)}$$

This value can be used to estimate the delay between the two vectors.

The value:

$$corrm(X[t], Y[t]) = \max_{n}(\tilde{r}_{xy}[n]) \qquad \text{(A.2-43)}$$

The similarity measure *'corrm'* represents an estimate of the accuracy of the estimated delay.

A.2.3.7.2 Local maxima calculation

In some cases the delay estimation as defined in clause A.2.3.7.1 leads to uncertain results. Taking higher order maxima into account in delay search may improve the accuracy.

The local maxima calculation of a vector v is done by following the steps below:

1. find global maxima x of the vector v which is a local maxima,
2. goto x-7 bins left to the vector,
3. search left side for the next local minima within the vector v which is the first element with v[x]>v[x-1],
4. goto x+7 bins right to the vector,
5. search right side for the next local minima within the vector v which is the first element with v[x]<v[x+1],
6. create a new vector V by removing all elements between the local left side minima and right side minima and,
7. continue with step 1 for next local maxima calculation.

In the following the index of the local maxima n of a vector v is referenced as LocalMaximaIndex(v,n), the value at the local maxima n of a vector v is referenced as LocalMaximaValue(v,n).

A.2.3.7.3 Definitions of the similarity measure for the spatial information aspects

For the delay estimation PEVQ-S (e) uses the spatial information aspects, stored in the matrices D_s and D_p, as defined in clause A.2.3.3. These types of matrices are referred as the similarity matrices.

A.2.3.7.3.1 Similarity measure

The similarity measure of the similarity matrices is defined as the similarity measure of the combined product-moment correlation vector of the individual information aspects.

$$corrmsim(D_s, D_p) = corrm\left(\frac{1}{6}\sum_{i=0}^{5}\tilde{r}_{D_s[i,t],D_p[i,t]}[n]\right) \quad \text{(A.2-44)}$$

A.2.3.7.3.2 Similarity index

The similarity index of two similarity matrices are defined as the index of the information aspect with the highest similarity measure.

$$corrmx(D_s, D_p) = \arg\max\left(\frac{1}{6}\sum_{i=0}^{5}\tilde{r}_{D_s[i,t],D_p[i,t]}[n]\right) \quad \text{(A.2-45)}$$

A.2.3.7.3.3 Delay between similarity matrices

The delay between two similarity matrices is defined as the delay of the information aspect with the highest similarity measure

$$delaysim(D_s, D_p) = delay\left(D_s\left[corrmx\left(D_s, D_p\right)\right], D_p\left[corrmx\left(D_s, D_p\right)\right]\right) \quad \text{(A.2-46)}$$

A.2.3.7.4 Realignment procedure

The realignment procedure of a similarity matrix *D[j,t]* is defined as follows:

$$realign(D_s, NumFrames) = \begin{cases} \begin{array}{l} shift\ each\ element\ of\ D_s \\ NumFrames\ columns\ to\ the \\ right,\ discarding\ the\ last \\ NumFrrames\ columns. \\ Set\ the\ elements\ in\ the\ first \\ NumFrames\ columns\ of\ D_s \\ to\ zero \end{array} & If\ \ NumFrames >= 0 \\ \\ \begin{array}{l} shift\ each\ element\ of\ D_s \\ NumFrames\ columns\ to\ the \\ left,\ discarding\ the\ first \\ NumFrames\ columns. \\ Set\ the\ elements\ in\ the\ last \\ NumFrames\ columns\ of\ D_s \\ to\ zero \end{array} & If\ \ NumFrames < 0 \end{cases} \quad \text{(A.2-47)}$$

A.2.3.7.5 Extract region of interest

The function $extr(D, first, last)$ creates a new similarity matrix *E* out of the similarity matrix *D* following the rule:

$$E[i,t] = D[i, t + first],\ \forall\ i \in \{0..5\}, t \in \{0, last - first - 1,\} \quad \text{(A.2-48)}$$

A.2.3.7.6 Length of the similarity matrix

The function $length(D)$ returns the number of rows of the similarity matrix D.

A.2.3.8 Coarse temporal alignment

The inputs to the coarse alignment process are the spatial similarity matrices of the degraded and the reference sequences D_s and D_P with *6xL* elements, with *L* being:

L = max(Nr of frames in PVS, Nr of frames in SRC). (A.2-49)

Any missing elements in the shorter vector are zero padded. The outcome of the process is a frame wise estimate of the delay and an estimate of the accuracy of the delay.

Based on segments in the distortion map a set of similarity matrices is created. Upon this, the coarse alignment creates a tree of sub matrices by iteratively splitting each similarity and sub matrix in two halves, until each sub matrix contains between 8 and 16 columns

The *'DelayInfo'* structure consists of frame number *'S'* which marks the start of segment, the numbers of frames in the segment named as *'L'* and the submatrix delay information as *'Del'*.

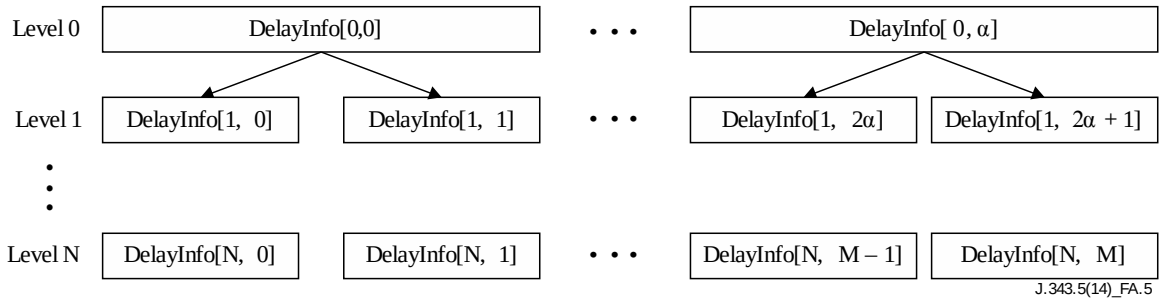


Figure A.5 – Submatrix tree

This procedure is performed using the following steps:

Step 1: raw global alignment

- find all un-overlapped segments $SI(t_\alpha, t_{\alpha+1})$
- estimate the overall delay between the extracted segments of the degraded similarity matrix and the uncut reference similarity matrix. $Del = delaysim(D_p(SI(t_\alpha, t_{\alpha+1})), D_s)$
- assign *DelayInfo[0,α].Del*=Del as delay for the matrix
- assign *DelayInfo[0,α].L=length*($D_p(SI(t_\alpha, t_{\alpha+1}))$) for the matrix
- assign *DelayInfo[0,α].S*=0 for the first column in the matrix
- perform delay correction using local maxima
- perform delay correction for severe outliers

Step 2: refine delay estimate using a tree approach

Apply the following to all submatrices from level 0 to level N as in Figure A.5 from left to right.

- let *i* be the level of the input segment
- let *j* be the index of the input segment
- let *L= DelayInfo[i,j].L* be the length of the input matrices
- let *Del= DelayInfo[i,j].Del* the delay between the input matrices

– let *S= DelayInfo[i,j].S* be the frame number assigned to the first column in the input matrix
– re-align the similarity matrix *Ds*

  Ds'=realign(Ds,Del);
– create two new sub matrices (left matrix and right matrix) by splitting the input matrices in two parts with an overlap of 50% at each side,

For the left side this results in:

Dp'[Left]=extr(Dp,-L/4+S,L/2+L/4+S)

Ds'[Left]=extr(Ds',-L/4+S,L/2+L/4+S)

For the right side this results in:

Dp'[Right]=extr(Dp,-L/4+L/2+S,L+L/2+S)

Ds'[Right]=extr(Ds',-L/4+L/2+S,L+L/2+S)

– calculate the delays between each pair of matrices:

  Del'[Left]=delaysim(Dp'[Left], Ds'[Left])+Del'

  Del'[Right]=delaysim(Dp'[Right], Ds'[Right])+Del'

For the left side matrix assign:

– *DelayInfo[i+1,2j].Del=Del'[Left]* as delay for the matrix
– assign *DelayInfo[i+1,2j].L=floor(L/2)* as length for the matrix
– assign *DelayInfo[i+1,2j].S=S* as the frame number for the first column in the matrix

For the right side matrix assign:

– *DelayInfo[i+1,2j+1].Del=Del'[Right]* as delay for the matrix
– assign *DelayInfo[i+1,2j+1].L=ceil(L/2)* as length for the matrix
– assign *DelayInfo[i+1,2j+1].S=S+ceil(L/2)* as the frame number for the first column in the matrix
– perform delay correction using local maxima
– perform delay correction for severe outliers

Step 3: repeat step 2 until the length of all resulting sub matrices is less than 19

A.2.3.8.1 Delay correction using local maxima

In case the values of the first and second order local maxima of the similarity measure are close to each other the delay estimation based on global maxima may be uncertain.

In this case the delay index and delay values are of the similarity measure are refined as follows:

- identify a segment k within the working level which satisfy the equation

$$\left|LocalMaxiaValue\left(\bar{r}\left[Dp[k], Ds[k]\right], 0\right) - LocalMaxima(\bar{r}[Dp[k], Ds[k]], 1)\right| < 0.1$$

- search for the first segment i left from k with

$$|LocalMaxiaValue(\bar{r}[Dp[i], Ds[i]], 0) - LocalMaxima(\bar{r}[Dp[i], Ds[i]], 1)| \geq 0.1$$

- search for the first segment j right from k with

$$|LocalMaxiaValue(\bar{r}[Dp[j], Ds[j]], 0) - LocalMaxima(\bar{r}[Dp[j], Ds[j]], 1)| \geq 0.1$$

- estimate an expected delay calculating the mean delay between Delay of segment i and segment j

  EstimatedDelay=0.5(DelayInfo[i].Del+ DelayInfo[j].Del)*

- search within segment k for the local maxima $d \in \{0..4\}$ which is close to *EstimatedDelay*

$$|LocalMaximaIndex(\bar{r}[Dp[k], Ds[k]], d) - EstimatedDelay| \rightarrow min$$

- assign the new delay information of local maxima d to the segment info k

A.2.3.8.2 Delay correction for severe outliers

In rare cases especially for image sequences with low change in information the correlation approach may lead to severe outliers. For correction the DelayInfo segments are filtered as follow:

$$\text{If } (\ |DelayInfo[n, i-1].Del - \ DelayInfo[n, i+1].Del| < 4 \ ) \ and$$

$$|DelayInfo[n, i-1].Del - \ DelayInfo[n, i].Del| > 5 \ ) \ and$$

$$|DelayInfo[n, i+1].Del - \ DelayInfo[n, i].Del| > 5 \ ){$$

$$DelayInfo[n, i].Del = \ 0.5 * (DelayInfo[n, i-1].Del + \ DelayInfo[n, i].Del)$$

}

A.2.3.8.3 Specification for raw matching vector

A vector *RawMatch[t]* which for each frame in the PVS sequence assigns a corresponding frame from the SRC sequence may be calculated as described below:

```
For (j=0; j<M-1; j++)
{
    For (k= DelayInfo[N,j].S; k<DelayInfo[N,j+1].S; k++)
    {
        RawMatch[k]=k+DelayInfo[N,j].Del;
    }
}
For (k= DelayInfo[N,M].S;k< DelayInfo[N,M].S +DelayInfo[N,M].L; k++)
{
        RawMatch[k ]= k +DelayInfo[N,M].Del;
}
```

A.2.3.8.4 Specification of first and last index

The index of the first and the last frame in the PVS that can be assigned to the corresponding frame from SRC is calculated as follows:

```
StartMatchIndex=0
While(RawMatch[StartMatchIndex]<0)StartMatchIndex++

StopMatchIndex=Nr frames Degraded-1
While(RawMatch[StopMatchIndex]>=Nr frames Degraded)StopMatchIndex--
```

A.2.3.8.5 Specification of the fine alignment search range vector

For each index of j and level N, a search range value is assigned to a search description vector *'Search'* by the following steps:

Search[j] = 2

Search[j] = max(Search[j], MaxFluidityElements(FE_p*, DelayInfo[N,j].S, DelayInfo[N,j].S+ DelayInfo[N,j].L)*)

```
Search[j] = max(Search[j], MaxFluidityElements(FEs, DelayInfo[N,j].S, DelayInfo[N,j].S+
                                                                    DelayInfo[N,j].L) )

 If (abs((DelayInfo[N,j-1].S+DelayInfo[N,j-1].L)-DelayInfo[N,j].S)))<20
{
    Search[j] = max(Search[j], abs(DelayInfo[N,j].Del- DelayInfo[N,j-1].Del))
     Search[j] = max(Search[j], abs((DelayInfo[N,j-1].S+ DelayInfo[N,j-1].L)-
                                                                    DelayInfo[N,j].S))
  }

 If (abs(DelayInfo[N,j].S -(DelayInfo[N,j-1].S+DelayInfo[N,j+1].L)-)))<20
{
        Search[j]= max(Search[j], abs(DelayInfo[N,j].Del- DelayInfo[N,j+1].Del))
         Search[j] = max(Search[j], abs((DelayInfo[N,j].S+ DelayInfo[N,j].L)-
                                                                    DelayInfo[N,j+1].S))
    }
```

A vector *SearchRange[t]* for each frame may be calculated as described below:

```
    For (j=0; j<M-1; j++)
    {
            For (k= DelayInfo[N,j].S; k<DelayInfo[N,j+1].S  ; k++)
            {
                SearchRange[k ]= Search[j]);
            }
    }
    For (k= DelayInfo[N,M].S; k< DelayInfo[N,M].S+ DelayInfo[N,M].L; k++)
    {
            SearchRange[k ]= Search[j]);
    }
```

A.2.3.9 Coarse luminance alignment

A.2.3.9.1 Determination of the coarse luminance alignment correction curve

A preliminary luminance alignment based on the luminance part of the temporally unaligned SRC and PVS are carried out. Processing is done on a resized image. Resize factor is 0.25 for HD sequences and 0.75 for VGA sequences.

First the histograms of the luminance components are computed by:

$$h_{s,y}[k] = \frac{1}{N \cdot W \cdot H} \sum_{t=0}^{N-1} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \delta\left[k, S_{p,y}[i, j, t]\right] \qquad \text{(A.2-50)}$$

$$h_{p,y}[k] = \frac{1}{N \cdot W \cdot H} \sum_{t=0}^{N-1} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \delta[k, P_{p,y}[i, j, t]] \tag{A.2-51}$$

With

$$\delta[a,b] = \begin{cases} 1 & if \quad a = b \\ 0 & otherwise \end{cases} \tag{A.2-52}$$

In the next step the cumulative histograms are calculated

$$HC_{S,y}[\lambda] = \sum_{k=0}^{\lambda} h_{S,y}[k] \tag{A.2-53}$$

$$HC_{P,y}[\lambda] = \sum_{k=0}^{\lambda} h_{P,y}[k] \tag{A.2-54}$$

The luminance correction $Correction\,Curve_y$, which aligns the luminance part of the temporally unaligned SRC and PVS, is calculated as described in the following:

```
 Int   binS     = 0
 Int   binP     = 128

 Float fracS    = hs,y[binS]
 Float cumFracS = HCs,y[binS]
 Float fracP      = hp,y[binP]
 Float cumFracP = HCp,y[binP]

 Int tarS       = 0
 Int SteepnessS = 0

while((HCs,y[binS]<= cumFracP) && (binS<255)
{
  binS++
}



 for (binP = 128; binP < 256; binP++)
 {
      fracP=hp,y[binP]
      cumFracP = HCp,y[binP]
      if (binS < 255)
      {
           if ((fracS < 0.0008) && (fracP < 0.0008))
           {
                binS++
                fracS = hs,y[binS]
                cumFracS = HCs,y[binS]

                tarS=binS

           }
           else
           {
                SteepnessS=0
                while ((cumFracS < cumFracP)&&
                         (binS < 255) && (SteepnessS <= 50)
                {
                     binS++
                     SteepnessS++
                     fracS = hs,y[binS]
                     cumFracS = HCs,y[binS]
                }
```

```
                    if(cumFracS >= cumFracP)
                    {

                          tarS=(binS-1)*( HCs,y[binS]- cumFracP)+
                                        (binS)*((cumFracP- HCsy[binS-1])/
                                                  ( HCs,y[binS]- HCs,y[binS-1]))

                          tarS=max(tarS,binS-1)
                          tarS=min(tarS,binS)

                    }
                    else
                    {
                      tarS=binS
                    }

              }
        }
        CorrectionCurvey[binP] = round(tarS)
  }

  binS = 255
  fracS = hs,y[binS]
  cumFracS = HCs,y[binS]


  cumFracP = HCp,y[128]
 while((HCs,y[binS]>= cumFracP) && (bins>0)
 {
   binS--
  }

  for (binP = 127; binP >= 0; binP--)
  {
        fracP=hp,y[binP]
        cumFracP = HCp,y[binP]
        if (binS >= 0)
        {
              if ((fracS < 0.0008) && (fracP < 0.0008))
              {
                    binS--
                    fracS = hs,y[binS]
                    cumFracS = HCs,y[binS]
                    tarS=binS
              }
              else
              {
                    Steepness=0
                    while ((cumFracS > cumFracP) &&
                              (binS >= 0)&&
                              (Steepness<=50))
                    {
                          bins--
                          Steepness++
                          fracS = hs,y[binS]
                          cumFracS = HCs,y[binS]
                    }

                    if(cumFracS <= cumFracP)
                    {

                          tarS=(binS)*( HCs,y[bins+1]- cumFracP)+
                                        (binS+1)*((cumFracP- HCs,y[binS])/
                                                  ( HCs,y[binS+1]- HCs,y[binS]))

                          tarS=max(tarS,binS)
                          tarS=min(tarS,bins+1)


                    }
                    else
```

```
                    {
                      tarS=binS
                    }
                }
            }
            CorrectionCurve_y[binP] = round(tarS)
        }
```

A.2.3.9.2 Histogram correction

The histogram correction is carried out by applying the *CorrectionCurve*$_y$ as a table lookup for each PVS component.

$$S_{c,y}[i,j,t] = S_{p,y}[i,j,t] \tag{A.2-55}$$

$$P_{c,y}[i,j,t] = CorrectionCurve_y\left[P_{p,y}[i,j,t]\right] \tag{A.2-56}$$

A.2.3.10 Temporal alignment

A.2.3.10.1 Determination of the temporal registration vector

The temporal registration of the received video sequence is done using a mean square error (MSE) criterion, and carried out on the coarse luminance aligned sequences. Processing is done on a resized image. The resize factor is 0.25 for HD sequences and 0.75 for VGA sequences.

By minimizing the function

$$f\left(\delta_x, \delta_{y,} t, RawMatch[t] + \delta_t[t]\right) -> \min; \tag{A.2-57}$$

Where

$$f(\delta_x, \delta_y, t_p, t_s) = \sqrt{\frac{1}{Norm} \sum_{i=\max(0,\delta_x)}^{\min(W,W+\delta_x)-1} \sum_{j=\max(0,\delta_y)}^{\min(H,H+\delta_y)-1} \left|P_{c,y}(i+\delta_x, j+\delta_y, t_p) - S_{c,y}(i,j,t_s)\right|^2}$$

$$with \tag{A.2-58}$$

$$Norm = (\min(W, W+\delta_x) - \max(0,\delta_x))(\min(W, W+\delta_y) - \max(0,\delta_y))$$

Over the three variables δ_x, δ_y and δ_t

$$\delta_x \in \{-4,...,4\}, \delta_y \in \{-4,...,4\}, \delta_t[t] \in \{-SearchRange[t], \ldots, SearchRange[t]\} \tag{A.2-59}$$

and the frames which satisfy:

$$StartMatchIndex \le t < StopMatchIndex \tag{A.2-60}$$

The values δ_{x0}, δ_{y0} and δ_{t0} are determined. Note that in case that the *SearchRange[t]* is 0.

δ_{t0} is known to be 0 in advance.

A matching vector which contains the best match between the reference and test images is given by:

$$MatchingPictureIndex[t] = RawMatch[t] + \delta_{t0}[t] - DE_s\left[RawMatch[t] + \delta_{t0}[t]\right] \tag{A.2-61}$$

The matching picture index is smoothed according to following recursion:

$$SmoothedMatching[t] = \begin{cases} MatchingPictureIndex[t] & if \quad DE_p[t] = 0 \\ SmoothedMatching[t-1]+1 & if \quad 0 < DE_p[t] < 8 \\ SmoothedMatching[t-1] & else \end{cases} \tag{A.2-62}$$

A.2.3.10.2 Temporal alignment correction

The temporal correction is carried out on the reprocessed sequences

$$S_{t,\mu}[i,j,t] = S_{p,\mu}[i,j,SmoothedMatching[t+StartMatchIndex]] \tag{A.2-63}$$

$$P_{t,\mu}[i,j,t] = P_{p,\mu}[i,j,t+StartMatchIndex] \tag{A.2-64}$$

Where $\mu \in \{Y, Cr, Cb\}$ represents the different colour planes

A.2.3.11 Spatial alignment

A.2.3.11.1 Determination of the spatial offset

The same MSE approach as in the temporal registration is used for determination of the spatial shift between the reference and test signals on a frame-by-frame basis. The alignment is carried out on the temporally aligned signals.

The spatial offsets are determined by minimizing the function:

$$f(\delta_x[t], \delta_y[t], t) -> \min \tag{A.2-65}$$

where:

$$f(\delta_x, \delta_y, t) = \sqrt{\frac{1}{Norm} \sum_{i=\max(0,\delta_x)}^{\min(W,W+\delta_x)-1} \sum_{j=\max(0,\delta_y)}^{\min(H,H+\delta_y)-1} \left| P_t(i+\delta_x, j+\delta_y, t) - S_t(i,j,t) \right|^2} \tag{A.2-66}$$

and

$$Norm = (\min(W, W+\delta_x) - \max(0,\delta_x))(\min(W, W+\delta_y) - \max(0,\delta_y))$$

Over the two variables δ_x, δ_y

$$\delta_x \in \{-16,...,16\}, \delta_y \in \{-16,...,16\}$$

The minimum values $\delta_{x0}[t], \delta_{y0}[t]$ represent the spatial offset between the reference and test sequence.

A.2.3.11.2 Spatial offset correction

The spatial corrected sequences are calculated by:

$$S_{s,\mu}[i,j,t] = S_{t,\mu}[i,j,t] \tag{A.2-67}$$

$$P_{s,\mu}[i,j,t] = \begin{cases} P_{t,\mu}[i+\delta_{x0}[t], j+\delta_{y0}[t], t] & if \;\; 0 \le (i+\delta_{x0}[t]) < W \;\; and \;\; 0 \le (j+\delta_{y0}[t]) < H \\ S_{s,\mu}[i,j,t] & else \end{cases} \tag{A.2-68}$$

Where $\mu \in \{Y, Cr, Cb\}$ represents the different colour planes.

A.2.3.12 Luminance – chrominance alignment

A.2.3.12.1 Determination of the correction curves

The first step is to compute the histograms of the reference and distorted image for the luminance and the chrominance components. Processing is done on a resized image. Resize factor is 0.5 for HD sequences and 1 for VGA sequences.

Let S_s represent the resized temporal and spatial aligned reference image, and P_s the resized temporal and spatial aligned distorted image. Then the histogram is calculated as follows:

$$h_{s,\mu}[k] = \frac{1}{N \cdot W \cdot H} \sum_{t=0}^{N-1} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \delta[k, S_s[i, j, t]] \tag{A.2-69}$$

$$h_{P,\mu}[k] = \frac{1}{N \cdot W \cdot H} \sum_{t=0}^{N-1} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \delta[k, P_\mu[i, j, t]] \tag{A.2-70}$$

$$\text{With } \delta[a,b] = \begin{cases} 1 & if \quad a = b \\ 0 & otherwise \end{cases} \tag{A.2-71}$$

Where $\mu \in \{Y, Cr, Cb\}$ represents the different colour planes.

In the next step the cumulative histograms of the reference and distorted signals are calculated:

$$HC_{s,\mu}[\lambda] = \sum_{k=0}^{\lambda} h_{s,\mu}[k] \tag{A.2-72}$$

$$HC_{P,\mu}[\lambda] = \sum_{k=0}^{\lambda} h_{P,\mu}[k] \tag{A.2-73}$$

The aim of the histogram correction is to align the histograms of the processed video sequence as close as possible to those of the source sequence. Assuming that the cumulative histograms of source sequence and the distorted sequence are strictly increasing so that their inverse exists, the transformation function may be found as:

$$Z_\mu[k] = HC_{S,\mu}^{-1}\left(HC_{P,\mu}^{-1}[k]\right) \tag{A.2-74}$$

In practice, the values at the upper and lower range are clipped to a certain threshold. In addition e.g., due to coding artifacts, some histogram values may be zero causing the cumulative histogram to be locally non invertible. Therefore, the transformation formula Z may not be directly applied.

The approach used in PEVQ-S (e) is a combination of using the transformation formula for Z whenever there is enough empirical probability mass and use a straight line computation elsewhere.

The following algorithmic description illustrates the computation of the luminance correction curve *CorrectionCurve*$_y$

```
Int   binS     = 0
Int   binP     = 128

Float fracS    = hs,y[binS]
Float cumFracS = HCs,y[binS]
Float fracP       = hp,y[binP]
Float cumFracP = HCp,y[binP]

Int tarS       = 0
Int SteepnessS = 0

while((HCsy[binS]<= cumFracP) && (binS<255)
{
```

```
        binS++
    }

    for (binP = 128; binP < 256; binP++)
    {
        fracP=h_{p,y}[binP]
        cumFracP = HC_{p,y}[binP]
        if (binS < 255)
        {
            if ((fracS < 0.0008) && (fracP < 0.0008))
            {
                binS++
                fracS = h_{s,y}[binS]
                cumFracS = HC_{s,y}[binS]

                tarS=binS

            }
            else
            {
                SteepnessS=0
                while ((cumFracS < cumFracP)&&
                        (binS < 255) && SteepnessS <= 50)
                {
                    binS++
                    SteepnessS++
                    fracS = h_{s,y}[binS]
                    cumFracS = HC_{s,y}[binS]
                }
                if(cumFracS >= cumFracP)
                {

                    tarS=(binS-1)*( HC_{s,y}[binS]- cumFracP)+
                                (binS)*((cumFracP- HC_{sy}[binS-1])/
                                        ( HC_{s,y}[binS]- HC_{s,y}[binS-1]))

                    tarS=max(tarS,binS-1)
                    tarS=min(tarS,binS)

                }
                else
                {
                  tarS=binS
                }

            }
        }
        CorrectionCurve_y[binP] = round(tarS)
    }

    binS = 255
    fracS = h_{s,y}[binS]
    cumFracS = HC_{s,y}[binS]


    cumFracP = HC_{p,y}[128]
   while((HC_{s,y}[binS]>= cumFracP) && (bins>0)
   {
       binS--
    }

    for (binP = 127; binP >= 0; binP--)
    {
        fracP=h_{p,y}[binP]
        cumFracP = HC_{p,y}[binP]
        if (binS >= 0)
        {
            if ((fracS < 0.0008) && (fracP < 0.0008))
            {
                binS--
                fracS = h_{s,y}[binS]
```

```
                cumFracS = HCs,y[binS]
                tarS=binS
            }
            else
            {
                Steepness=0
                while ((cumFracS > cumFracP) &&
                        (binS >= 0)&&
                        (Steepness<=50))
                {
                    bins--
                    Steepness++
                    fracS = hs,y[binS]
                    cumFracS = HCs,y[binS]
                }

                if(cumFracS <= cumFracP)
                {

                    tarS=(binS)*( HCs,y[bins+1]- cumFracP)+
                                (binS+1)*((cumFracP- HCs,y[binS])/
                                        ( HCs,y[binS+1]- HCs,y[binS]))

                    tarS=max(tarS,binS)
                    tarS=min(tarS,bins+1)


                }
                else
                {
                  tarS=binS
                }
            }
        }
        CorrectionCurvey[binP] = round(tarS)
    }
```

Due to the fact that the source sequence may be colourless, whereas the processed sequence contains cross colour artifacts, a minimum slope of the correction curve is required. Therefore the calculation of the chrominance correction curves is slightly different to that of the luminance curve.

This is presented in the following algorithmic description, where:

$$h_{sc} \in \{h_{s,Cr}, h_{s,Cb}\} HC_{sc} \in \{HC_{s,Cr}, HC_{s,Cb}\} \quad h_{pc} \in \{h_{P,Cr}, h_{P,Cb}\} HC_{pc} \in \{HC_{P,Cr}, HC_{P,Cb}\}$$

and

$$CorrectionCurve_c \in \{CorrectionCurve_{Cb}, CorrectionCurve_{Cr}\}$$

Should be set according to the processed chrominance plane

```
    Int   binS     = 0
    Int   binP     = 128
    Float fracS    = hs,c[binS]
    Float cumFracS = HCs,c[binS]
    Float fracP    = hs,c[binP]
    Float cumFracP = HCs,c[binP]
    Float tarS     = 0
    Int   oldBinS  = binS
    Float steps    = 0

   while((HCs,c[binS]<= cumFracP) && (binS<255)
   {
     binS++
    }

    for (binP = 128; binP < 256; binP++)
    {
        Steps = Steps+0.5
```

```
        fracP=h_{p,c}[binP]
        cumFracP = HC_{p,c}[binP]

        if (binS < 255)
        {
              if ((fracS < 0.0008) && (fracP < 0.0008))
              {
                    binS++
                    fracS = h_{sc}[binS]
                    cumFracS = HC_{s,c}[binS]
                    tarS=binS
              }
              else
              {
                    while ((cumFracS < cumFracP)&& (binS < 255))
                     {
                          binS++
                          fracS = h_{sc}[binS]
                          cumFracS = HC_{s,c}[binS]
                    }
                    if(cumFracS >= cumFracP)
                    {

                          tarS=(binS-1)*( HC_{s,c}[binS]- cumFracP)+
                                         (binS)*((cumFracP- HC_{s,c}[binS-1])/
                                                   ( HC_{s,c}[binS]- HC_{s,c}[binS-1]))

                          tarS=max(tarS,binS-1)
                          tarS=min(tarS,binS)

                    }
                    else
                    {
                      tarS=binS
                    }
              }
        }

        if (steps >= 1)
        {
              if ((binS - oldBinS)/steps < 1)
              {
                    binS++
                    tars++
              }
              steps = 0
              oldBinS = binS
        }
        CorrectionCurve_c[binP] = round(tarS)
  }

  binS = 255
  fracS = h_{sc}[binS]
  cumFracS = HC_{s,c}[binS]
  oldBinS = binS
  steps=0

  cumFracP = HC_{p,c}[128]
 while((HC_{s,c}[binS]>= cumFracP) && (binS>0) binS--

  for (binP = 127; binP >= 0; binP--)
  {
        Steps=Steps+0.5
        fracP=h_{pc}[binP]
        cumFracP = HC_{p,c}[binP]
        if (binS >= 0)
        {
              if ((fracS < 0.0008) && (fracP < 0.0008))
              {
                    binS--
```

```
                fracS = hsc[c,binS]
                cumFracS = HCs,c[c,binS]
            }
            else
            {
                while ((cumFracS > cumFracP) && (binS >= 0))
                {
                    binS--
                    fracS = hsc[binS]
                    cumFracS = HCs,c[binS]
                }

                if(cumFracS <= cumFracP)
                {

                    tarS=(binS)*( HCs,c[bins+1]- cumFracP)+
                                  (binS+1)*((cumFracP- HCs,c[binS])/
                                            ( HCs,c[binS+1]- HCs,c[binS]))

                    tarS=max(tarS,binS)
                    tarS=min(tarS,binS+1)


                }
                else
                {
                  tarS=binS
                }

            }
        }
        if (steps >= 1)
        {
            if ((oldBinS-binS)/steps < 1)
            {
                binS++
                tarS++
            }
            steps = 0
            oldBinS = binS
        }
        CorrectionCurvec[binP] =round(tarS)
    }
```

A.2.3.12.2 Histogram correction

To carry out the histogram correction on the raw processed sequence is computationally simple. It involves a simple table lookup for each component.

$$S_{A,\mu}[i,j,t] = S_{s,\mu}[i,y,t] \tag{A.2-75}$$

$$P_{A,\mu}[i,j,t] = CorrectionCurve_{\mu}\left[P_s[i,y,t]\right] \tag{A.2-76}$$

Where $\mu \in \{Y, Cr, Cb\}$ represents the different colour planes.

A.2.3.13 Spatial distortions analysis

In the spatial domain four different indicators are calculated.

These are:

- one describing the overall distortions in image fields;
- one describing distortions found in the edge of the image components;
- one describing distortions within colour fields;
- one that describes temporal variability within the sequences.

A.2.3.13.1 Field indicator

The field indicator is designed to evaluate the severity of distortions within image regions. The calculation is based on the luminance part of the images.

In a first step the temporal, spatial and histogram corrected reference image $S_{A,Y}[t]$, along with corrected test images $P_{A,Y}[t]$ are segmented in 41 rows and 41 columns resulting in n=168 sub blocks with the block with BW=$\frac{W}{41}$ and block height BH=$\frac{H}{41}$.

For each sub block BS(n) and BP(n) with n$\in\{0..167\}$ a correlation estimate is carried out by following:

$$c[n,t]=\frac{10^{-10}+\sum_{i=0}^{BW-1}\sum_{j=0}^{BH-1}\left(BS[n,i,j,t]-\overline{\chi}\right)\left(BP[n,i,j,t]-\overline{\xi}\right)}{10^{-10}+\sqrt{\sum_{i=0}^{BW-1}\sum_{j=0}^{BH-1}\left(BS[n,i,j,t]-\overline{\chi}\right)^2\sum_{i=0}^{BW-1}\sum_{j=0}^{BH-1}\left(BP[n,i,j,t]-\overline{\xi}\right)^2}} \quad \text{(A.2-77)}$$

with:

$$\overline{\chi}=\frac{1}{BW\cdot BH}\sum_{i=0}^{BW-1}\sum_{j=0}^{BH-1}BS[n,i,j,t] \quad \text{(A.2-78)}$$

$$\overline{\xi}=\frac{1}{BW\cdot BH}\sum_{i=0}^{BW-1}\sum_{j=0}^{BH-1}BP[n,i,j,t] \quad \text{(A.2-79)}$$

The aggregation over the sub blocks is done using a logistic mapping:

$$F[t]=\frac{6}{1+e^{\left(a\cdot\left(1-\frac{1}{n}\sum_{n}100\left(1-c[n,t]\right)\right)+b\right)}} \quad \text{(A.2-80)}$$

With the constants:

a=0.13793102994685238

b=-1.8482757486713099

the field indicator is then calculated using a L2 Norm over the frame wise field distortions over time.

$$FieldIndicator=\sqrt{\frac{1}{N}\sum_{t=0}^{N-1}F^2[t]}$$

The field indicator is then calculated by averaging the frame wise field distortions over time:

$$FieldIndicator=\frac{1}{N}\sum_{t=0}^{N-1}F[t] \quad \text{(A.2-81)}$$

A.2.3.13.2 Edge of image

The edge of the reference and degraded image is computed as an approximation to the local gradient of the luminance and chrominance components of the aligned signals.

$$S_{edge,\mu}[t]=MeanFilter_{xy}(\sqrt{\left(S_{A,\mu}[t]*K_v\right)^2+\left(S_{A,\mu}[t]*K_h\right)^2} \quad \text{(A.2-82)}$$

$$P_{edge,\mu}[t] = MeanFilter_{xy}(\sqrt{\left(P_{A,\mu}[t] * K_v\right)^2 + \left(P_{A,\mu}[t] * K_h\right)^2} \quad \text{(A.2-83)}$$

With the filter kernel is set as follows:

For HD1080p sequences:

$$K_v = [0.33, 0.33, 0.33, 0, 0, 0, -0.33, -0.33, -0.33]$$

$$K_h = [0.33, 0.33, 0.33, 0, 0, 0, -0.33, -0.33, -0.33]$$

For HD 1080i sequences:

$$K_v = [0.33, 0.33, 0.33, 0, 0, 0, -0.33, -0.33, -0.33]$$

$$K_h = [0.5, 0.5, 0, -0.5, -0.5]$$

For VGA sequences:

$$K_v = [0.5, 0.5, 0, -0.5, -0.5]$$

$$K_h = [0.5, 0.5, 0, -0.5, -0.5]$$

$\mu \in \{Y, Cr, Cb\}$ represents the different colour planes.

$MeanFilter_{xy}()$ performs a nonlinear filtering operation on the image. The function sets each pixel in the destination image to the mean value of the image pixel values in the neighbourhood of the processed pixel. For VGA video sequences the kernel size is set to 5 × 5. For 1080p sequences the kernel size if set to 11 × 11 and for 1080i sequences the kernel size is set to 11 × 7.

A.2.3.13.3 Luminance indicator

The luminance indicator is calculated based on the luminance part of the edge images.

Subjective experiments are normally carried out with a background illumination that corresponds to the mid-range grey level of the display. The HVS of the test persons adapts to this background illumination. As a result, the visual qualities of dark and light areas in the picture are of lesser importance to the quality judgement. In order to reflect this effect, the deviation of the luminance from 100 is computed for both the source video signal and the processed video signal. It is the maximum of these deviations that is used in the edge frame indicator.

$$dev[i, j, t] = \max\left(\left|S_{A,y}[i, j, t] - 100\right|, \left|P_{A,y}[i, j, t] - 100\right|\right) \quad \text{(A.2-84)}$$

Wherever S_{edge} is nonzero and P_{edge} is smaller than S_{edge} the distortion is perceived as a loss of sharpness. Some source sequences have overall higher edges than others. It turns out that the *relative* decrease of P_{edge}' with respect to S_{edge}' is a better indicator of the loss of sharpness than the absolute difference. The indicator computed in this section not only indicates loss of sharpness (P_{edge} *[i,j,t]* minus S_{edge} *[i,j,t]* negative). Also, the introduction of sharpness is registered as a distortion (P_{edge} *[i,j,t]* minus S_{edge} *[i,j,t]* positive). The relativeness of the effect also manifests itself locally for introduced edge. The introduction of edge in areas with a lot of edges is less disturbing than the introduction of sharpness where little edge is originally present.

$$e_Y[i, j, t] = \frac{P_{edge,Y}[i, j, t] - S_{edge,Y}[i, j, t]}{S_{edge,Y}[i, j, t] + 80 + dev[i, j, t]} \quad \text{(A.2-85)}$$

The normalized change in edge e is locally clipped to the range [−40,40].

$$e_{cliped,Y}[i,j,t] = \begin{cases} -40 & if \quad e_Y[i,j,t] \le -40 \\ 40 & if \quad e_Y[i,j,t] \ge 40 \\ e_Y[i,j,t] & otherwise \end{cases} \tag{A.2-86}$$

The aggregation over space is done using a weighted L5 Norm of the clipped change in edge image:

$$e_Y[t] = \sqrt[5]{\frac{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}\left|e_{clipped,Y}[i,j,t]\right|^5 w[i,j]}{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1} w[i,j]}} \tag{A.2-87}$$

$$w[i,j] = \left|\sin(\pi\frac{i}{W})\cdot\sin(\pi\frac{j}{H})\right| \tag{A.2-88}$$

The luminance indicator is then calculated by averaging the frame wise edge distortions over time:

$$LumIndicator = \frac{1}{N}\sum_{t=0}^{N-1} e_Y[t] \tag{A.2-89}$$

A.2.3.13.4 Chrominance indicator

The chrominance indicator uses a similar approach as the luminance indicator.

The HVS is less sensitive to errors in the colour difference components that occur in areas that have saturated colours. This is even emphasized in bright areas. To reflect this, the colour saturation is computed as follows. As for the deviation signal, the maximum of the colour saturation of the reference signal and the degraded signal is taken:

$$Mx[i,j,t] = \sqrt{\left(S_{A,cb}[i,j,t]-128\right)^2 + \left(S_{A,cr}[i,j,t]-128\right)^2} \tag{A.2-90}$$

$$My[i,j,t] = \sqrt{\left(P_{A,cb}[i,j,t]-128\right)^2 + \left(P_{A,cr}[i,j,t]-128\right)^2} \tag{A.2-91}$$

$$dev_{cb;cr}[i,j,t] = \max\left(Mx[i,j,t];My[i,j,t]\right) \tag{A.2-92}$$

Then the normalized change of both colour components are evaluated,

$$e_{cb}[i,j,t] = \frac{P_{A,cb}[i,j,t] - S_{A,cr}[i,j,t]}{S_{A,cb}[i,j,t] + 40 + 0.8\cdot dev[i,j,t]} \tag{A.2-93}$$

$$e_{cr}[i,j,t] = \frac{P_{A,cr}[i,j,t] - S_{A,cr}[i,j,t]}{S_{A,cr}[i,j,t] + 40 + 0.8\cdot dev_{cb;cr}[i,j,t]} \tag{A.2-94}$$

And clipped locally to the range [−40,40].

$$e_{clipped,cb}[i,j,t] = \begin{cases} -40 & if \quad e_{cb}[i,j,t] \le -40 \\ 40 & if \quad e_{cb}[i,j,t] \ge 40 \\ e_{cb}[i,j,t] & otherwise \end{cases} \tag{A.2-95}$$

The aggregation over space is done using a weighted sum of the clipped change in edge image:

$$e_{cb}[t] = \frac{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1} e_{clipped,cb}[i,j,t]w[i,j]}{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1} w[i,j]} \quad \text{(A.2-96)}$$

$$e_{cr}[t] = \frac{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1} e_{clipped,cr}[i,j,t]w[i,j]}{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1} w[i,j]} \quad \text{(A.2-97)}$$

$$w[i,j] = \left|\sin(\pi\frac{i}{W})\cdot\sin(\pi\frac{j}{H})\right| \quad \text{(A.2-98)}$$

The chrominance indicator is then calculated by averaging the frame wise edge distortions over time:

$$ChromIndicator = \frac{0.5}{N}\sum_{t=0}^{N-1}\left(e_{cb}[t] + e_{cb}[t]\right) \quad \text{(A.2-99)}$$

A.2.3.13.5 Temporal variability indicator

The edge indicator is a pure spatial indicator. However, the spatial content of a sequence is judged more critically in case of still images than for images with fast motion and rapid changes. To reflect this, contributions to the $DMOS_P$ from *e* (and *n*) indicator should be compensated by a contribution from an indicator that measures the temporal variability of the source or processed video sequence. The (peak) temporal variability of the processed video signal is also influenced by transmission errors and the presence or absence of frame repeats. As a result, the temporal variability is best measured on the luminance of the source sequence.

$$d[i,j,t] = \left|S_{A,Y}[i,j,t] - S_{A,Y}[i,j,t-1]\right| - \left|P_{A,Y}[i,j,t] - P_{A,Y}[i,j,t-1]\right| \quad \text{(A.2-100)}$$

To measure the temporal component, the negative part of the variability measure is taken onto account.

$$d_C[i,j,t] = \begin{cases} 0 & if\ \ d[i,j,t] \le 0 \\ -d[i,j,t] & otherwise \end{cases} \quad \text{(A.2-101)}$$

The aggregation over space is done using an L norm of the temporal component

$$d_C[t] = \frac{1}{H\cdot W}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} d_C[i,j,t] \quad \text{(A.2-102)}$$

The temporal component indicator is then calculated by averaging the frame wise omitted distortions over time

$$TempComponentIndicator = \frac{1}{N}\sum_{t=0}^{N-1} d_C[t] \quad \text{(A.2-103)}$$

A.2.4 Estimation of the perceived quality – MOS

The perceived video quality is estimated by mapping the indicators to a single number using a sigmoid approach. Let $I[i]$ represent the indicators.

Then the mapping function may be defined by a set of input scaling factors $I_{\min}[i]$, $I_{\max}[i]$, a set of scaling factors $w_x[i]$, and a set of output scaling factors

$$I_{\lim}[i] = \begin{cases} I_{\min}[i] & if \quad I[i] < I_{\min}[i] \\ I_{\max}[i] & if \quad I[i] > I_{\max}[i] \\ I\ [i] & otherwise \end{cases} \tag{A.2-104}$$

$$Score = LinearOffset + \sum_{i=0}^{4} \frac{w[i]}{1 + e^{\alpha[i] \cdot I_{\lim}[i] + \beta[i]}} \tag{A.2-105}$$

$$OMOS = \begin{cases} 1 & if \quad Score < 1 \\ 5.5 & if \quad Score > 5.5 \\ Score & otherwise \end{cases} \tag{A.2-106}$$

Different mappings were used for HD and VGA resolution as presented in the tables below.

Mapping coefficients used for HD resolution

Index (i)	Indicator ($I[i]$)	$I_{\min}[i]$	$I_{\max}[i]$	$w[i]$	$\alpha[i]$	$\beta[i]$
0	LumIndicator	0.262699	24.1489181	-4.3532315	0.47263432	-1.19331214
1	FieldIndicator	0.251399	5.1255761	-3.0247624	-0.4103259	-0.51779709
2	TempComponent Indicator	16.12980	1274.99499	-0.7244337	-0.0194964	15.86824720
3	Bitstream Indicator	0.0000000	1.000000	0.25423694	-32.483560	16.76871352
4	ChromIndicator	0.288121	5.3126651	-0.2814391	6.83161757	-16.2944602
LinearOffset		7.8037367056				

Mapping coefficients used for VGA resolution

Index (i)	Indicator ($I[i]$)	$I_{\min}[i]$	$I_{\max}[i]$	$w[i]$	$\alpha[i]$	$\beta[i]$
0	LumIndicator	0.80835890	18.7453101	-2.5550374	0.68643158	-3.49210506
1	FieldIndicator	0.3219549	4.8410591	-2.3232843	-0.9333235	0.640197228
2	TempComponent Indicator	4.97828939	1730.8341	-499.09468	0.02358843	-20.8281465
3	Bitstream Indicator	0.000000	1.000000	-19.628381	15.6420289	4.357766893
4	ChromIndicator	0.4326929	4.1263161	2.90752969	10.8294678	-4.08860985
LinearOffset		24.784428392				

Annex B

YHyFRe

(Hybrid-FRe model)

(This annex forms an integral part of this Recommendation.)

B.1 Introduction

The YHyFRe model first computes a video quality metrics (VQM) value using the total number of packets and number of packet loss using a predefined look-up table (LUT). Then, the EPSNR value is computed using [ITU-T J.342] (256k side channel). Finally, post-processing is applied to reflect various impairments due to transmission errors.

B.2 Hybrid-FRe VQM computation

B.2.1 Feature computation

B.2.1.1 Total number of packets and number of packet loss

The total number of packets (*TotalPacket*) is computed as follows:

$$TotalPacket = \begin{cases} TS\ packet\ number & if\ TS\ protocol \\ \frac{1}{180} RTP\ PayloadSize & otherwise \end{cases}$$

The number of packet loss ($TotalPacket_{loss}$) is computed as follows:

$$TotalPacket_{Loss} = \begin{cases} TS\ loss\ packet\ number & if\ TS\ protocol \\ \frac{1}{180}(RTP\ loss\ packet\ number \times Average\ RTP : packet\ size) & otherwise \end{cases}$$

Then two features (X_{enc}, Y_{enc}) are computed as follows:

$$X_{enc} = \log_{10}(TotalPacket)$$
$$Y_{end} = \log_{10}(TotalPacket_{LOSS} + 1)$$

B.2.1.2 Green block feature

Some videos may contain mono-colour blocks due to severe transmission errors. A feature (*Greenblk*) reflecting this impairment is computed as follows:

$$Uzero_{pixel}(i, j, k) = \begin{cases} 1 & U(i, j, k) = 0 \\ 0 & otherwise \end{cases}$$

$$Vzero_{pixel}(i, j, k) = \begin{cases} 1 & V(i, j, k) = 0 \\ 0 & otherwise \end{cases}$$

$$Uzero_{line}(j, k) = \sum_{i=1}^{width} Uzero_{pixel}(i, j, k)$$

$$Vzero_{line}(j, k) = \sum_{i=1}^{width} Vzero_{pixel}(i, j, k)$$

$$Uzero_{flag}(j,k) = \begin{cases} 1 & Uzero_{line}(j,k) > width/8 \\ 0 & otherwise \end{cases}$$

$$Vzero_{flag}(j,k) = \begin{cases} 1 & Vzero_{line}(j,k) > width/8 \\ 0 & otherwise \end{cases}$$

$$Uzero = \sum_{k=1}^{NumFrame} \sum_{j=1}^{height} Uzero_{flag}(j,k)$$

$$Vzero = \sum_{k=1}^{NumFrame} \sum_{j=1}^{height} Vzero_{flag}(j,k)$$

$$Greenblk = \frac{1}{NumFrame}(Uzero + Vzero)$$

Here, *U* is the *u* channel and *V* is the *v* channel in the yuv video format.

B.2.1.3 Freeze features

To compute a freeze feature, the frame difference is calculated using the luminance channel:

$$FrameDiff(k) = \frac{1}{FramePixelSize} \sum_{(i,j)} |Y(i,j,k) - Y(i,j,k-1)|$$

$$FreezeFlag(k) = \begin{cases} 1 & if \quad FrameDiff(k) < Th_{frz} \\ 0 & otherwise \end{cases}$$

$$FRZ_{total} = \sum_{k} FreezeFlag(k)$$

$$FreezeFlag_{occur}(k) = \begin{cases} 1 & if \quad FreezeFlag(k) = 1 \ and \ FreezeFlag(k-1) = 0 \\ 0 & otherwise \end{cases}$$

$$FRZ_{num} = \sum_{k} FreezeFlag_{occur}(k)$$

The maximum freeze length " $FRZ_{\max}$ " is also calculated, which is the longest freeze interval. Thus three freeze features are calculated: FRZ_{total}, FRZ_{num}, $FRZ_{\max}$.

B.2.2 VQM computation

B.2.2.1 VQM computation using encrypted bitstream data and LUT

Using a LUT, $HFR1_{enc}$ is computed as follows:

$$HNR1_{enc} = LUT_{enc}(X_{enc}, Y_{enc})$$

The function (*LUTenc*) uses the bilinear interpolation. The LUT for HD and the LUT for VGA/WVGA are provided electronically in the Excel file attached to this Recommendation.

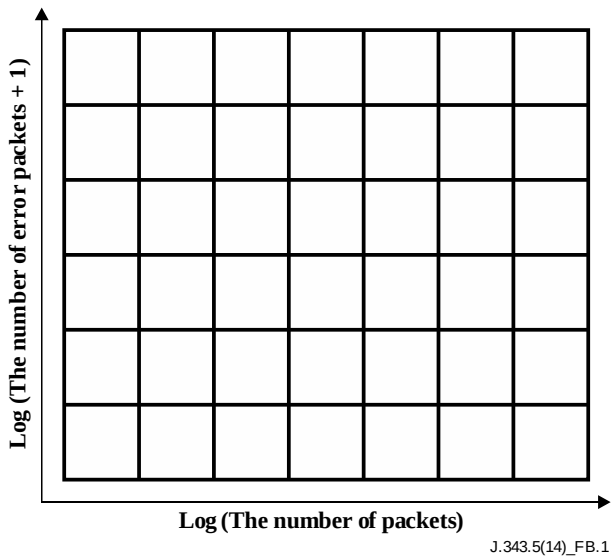


Figure B.1 – A look-up-table for VQM computation

B.2.2.2 EPSNR

The EPSFR value ($EPSNR$) is computed without the post-processing using [ITU-T J.342] (256k side channel).

B.2.3 Post-processing

B.2.3.1 Post-processing for the VQM value computed using encrypted bitstream data

The VQM value computed using encrypted bitstream data and LUT ($HNR1_{enc}$) is used as input (vqm_{in}) for the post-processing in this section.

First, the green block impairment is reflected as follows:

$$vqm_1 = \begin{cases} MIN(vqm_{in}, 1.6) & if \quad Greenblk > 1.0 \\ MIN(vqm_{in}, 2.5) & if \quad Greenblk > 0.0 \\ vqm_{in} & otherwise \end{cases}$$

Second, the frame rate is considered for VGA/WVGA as follows:

$$vqm_2 = \begin{cases} MIN(vqm_1, 3.2) & if \quad fps < 6 \\ MIN(vqm_1, 3.5) & if \quad fps < 10 \\ vqm_1 & otherwise \end{cases}.$$

Finally, the freeze impairment is reflected as follows:

$$FRZ_{temp} = \begin{cases} FRZ_{total} - \left(1 - \dfrac{fps}{fps_{original}}\right) \times fps_{original} \times VideoSec & if \quad Resolution = VGA \ or \ Resolution = WVGA \\ FRZ_{total} & otherwise \end{cases}$$

$$FRZ_{\log} = MIN(\log_{10}(FRZ_{temp} + 1.0),\ 2.3)$$

$$vqm_{out} = \begin{cases} MIN(vqm_2, 4 - \log_{10}(FRZ_{\log} - 0.3) \times 3.8) & if \quad FRZ_{\log} > 1.3 \\ vqm_2 & otherwise \end{cases}$$

B.2.3.2 Post-processing for the EPSNR value

The EPSNR value ($EPSNR$) is used as input (vqm_{in}) for the post-processing in this section.

First, upper and lower bounds are applied with scaling as follows:

$$vqm_1 = \begin{cases} 1.4 & if \;\; vqm_{in} < 23 \\ 4.4 & if \;\; vqm_{in} > 42 \\ \frac{3}{19}(vqm_{in} - 23) + 1.4 & otherwise \end{cases}$$

Second, the error rate is considered as follows:

$$ErrorRate = \frac{TotalPacket_{Loss}}{TotalPacket + TotalPacket_{Loss}}$$

$$vqm_2 = \begin{cases} vqm_1 - 1 & if \;\; ErrorRate > 0.01 \\ vqm_1 & otherwise \end{cases}$$

Third, the error area is considered as follows:

$$vqm_3 = \begin{cases} MIN(vqm_2, 1.5) & if \;\; ErrorArea > 20 \\ MIN(vqm_2, 2.0) & if \;\; ErrorArea > 10 \\ MIN(vqm_2, 3.0) & if \;\; ErrorArea > 5 \\ vqm_2 & otherwise \end{cases}$$

Finally, the number of freeze, maximum freeze length and the total freeze are considered as follows:

$$vqm_4 = \begin{cases} MIN(vqm_3, 2.2) & if \;\; FRZ_{\max} > 100 \\ MIN(vqm_3, 2.5) & if \;\; FRZ_{\max} > 50 \\ vqm_3 & otherwise \end{cases}$$

$$vqm_5 = \begin{cases} MIN(vqm_4, 2.5) & if \;\; FRZ_{total} > 100 \\ MIN(vqm_4, 3.0) & if \;\; FRZ_{total} > 60 \\ vqm_4 & otherwise \end{cases}$$

$$EPSNR_{out} = \begin{cases} MIN(vqm_5, 2.5) & if \;\; FRZ_{num} > 12 \\ vqm_5 & otherwise \end{cases}$$

B.2.3.3 Final VQM

The final VQM is computed by averaging the EPSFR value ($EPSNR_{out}$) and vqm_{out} as follows:

$$VQM_{final} = \frac{1}{2}(EPSNR_{out} + vqm_{out})$$

Bibliography

[b-VQEG Hybrid] Video Quality Experts Group (2014), *Hybrid Perceptual/Bitstream Validation Test Final Report*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	**Cable networks and transmission of television, sound programme and other multimedia signals**
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2015