NTIA Report 02-392

Video Quality Measurement Techniques

Stephen Wolf
Margaret Pinson

report series

U.S. DEPARTMENT OF COMMERCE • National Telecommunications and Information Administration

NTIA Report 02-392

Video Quality Measurement Techniques

Stephen Wolf
Margaret Pinson

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary

Nancy J. Victory, Assistant Secretary
for Communications and Information

June 2002

CONTENTS

FIGURES

TABLES

TERMS AND DEFINITIONS

4:2:2 - A Y, Cb, Cr image sampling format where chrominance planes (Cb and Cr) are sampled horizontally at half the luminance (Y) plane's sampling rate. See Rec. 601 [13].

Absolute Temporal Information (ATI) – A feature derived from the absolute value of temporal information images that are computed as the difference between successive frames in a video clip. ATI quantifies the amount of motion in a video scene. See section [4.5] for the precise mathematical definition.

American National Standards Institute (ANSI) - Serves as administrator and coordinator of the United States private sector voluntary standardization system.

Alliance for Telecommunications Industry Solutions (ATIS) - A North American standards body that develops telecommunications standards, operating procedures, and guidelines through its sponsored committees and forums.

Big YUV - The binary file format used for storing clips that have been sampled according to Rec. 601. In the Big YUV format, all the video frames for a scene are stored in one large binary file, where each individual frame conforms to Rec. 601 sampling. The Y represents the luminance channel information, the U represents the blue color difference channel (i.e., C_B in Rec. 601), and the V represents the red color difference channel (i.e., C_R in Rec. 601). The pixel ordering in the binary file is the same as that specified in SMPTE 125M [23]. The full specification of the Big YUV file format is given in section 2 and software routines for reading and displaying Big YUV files are given in [35].

Clip - Digital representation of a scene that is stored on computer media.

Clip VQM - The VQM of a single clip of processed video.

Chrominance (C, C_B, C_R) - The portion of the video signal that predominantly carries the color information (C), perhaps separated further into a blue color difference signal (C_B) and a red color difference signal (C_R).

Codec - Abbreviation for a coder/decoder or compressor/decompressor.

Common Intermediate Format (CIF) - A video sampling structure used for video teleconferencing where the luminance channel is sampled at 352 pixels by 288 lines [14].

Feature - A quantity of information associated with, or extracted from, a spatial-temporal sub-region of a video stream (either an original video stream or a processed video stream).

Field - One half of a frame, containing all of the odd or even lines.

Frame – One complete television picture.

Frames per Second (FPS) - The number of original frames per second transmitted by the video system under test. For instance, an NTSC video system transmits approximately 30 FPS.

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

H.261 - Abbreviation for ITU-T Recommendation H.261 [14].

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

HRC VQM - The VQM of an HRC computed as the average of all the individual Clip VQMs.

Input Video - Video before being processed or distorted by an HRC (see Figure 1). Input video may also be referred to as Original Video.

Institute for Telecommunication Sciences (ITS) - The research and engineering laboratory of the National Telecommunications and Information Administration, U.S. Department of Commerce.

Institute for Radio Engineers (IRE) Unit - A unit of voltage commonly used for measuring video signals. One IRE is equivalent to 1/140 of a volt.

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

Mean Opinion Score (MOS) - The average subjective quality judgment assigned by a panel of viewers to a processed video clip.

Moving Picture Experts Group (MPEG) - A working group of ISO/IEC in charge of the development of standards for coded representation of digital audio and video (e.g., MPEG-1, MPEG-2, MPEG-4).

National Television Systems Committee (NTSC) - The 525-line analog color video composite system adopted by the US and most other countries (excluding Europe) [24].

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). Offset of the luminance signal is commonly known as brightness.

Original Region of Interest (OROI) - A Region of Interest (ROI) extracted from the original video, specified in Rectangle Coordinates.

Original Video - Video before being processed or distorted by an HRC (see Figure 1). Original video may also be referred to as input video since this is the video input to the digital video transmission system.

Original Valid Region (OVR) - The Valid Region of an original video clip, specified in Rectangle Coordinates.

Output Video - Video that has been processed or distorted by an HRC (see Figure 1). Output video may also be referred to as Processed Video.

Over-scan - The portion of the video that is not normally visible on a standard television monitor.

Peak Signal-to-Noise Ratio (PSNR) - Peak signal to noise ratio as defined by ANSI T1.801.03-1996 [3], or alternatively, by an ANSI Accredited Committee T1 Technical Report [8].

Phase-Altering Line (PAL) - The 625-line analog color video composite system adopted predominantly in Europe with the exception of a few other countries around the world.

Parameter - A measure of video distortion that is the result of comparing two parallel streams of features, one stream from the original video and the corresponding stream from the processed video.

Processed Region of Interest (PROI) - A Region of Interest (ROI) extracted from the processed video and corrected for spatial shifts of the HRC, specified in Rectangle Coordinates.

Processed Video - Video that has been processed or distorted by an HRC (see Figure 1). Processed video may also be referred to as output video since this is the video output from the digital video transmission system.

Processed Valid Region (PVR) - The Valid Region of a processed video clip from an HRC, specified in Rectangle Coordinates. The PVR is always referenced to the original video so it is necessary to correct for any spatial shifts of the video by the HRC before computing PVR. Thus, PVR is always contained within the Original Valid Region (OVR). The region between the PVR and the OVR is that portion of the video that was blanked or corrupted by the HRC.

Production Aperture - The image lattice that represents the maximum possible image extent in a given standard. The Production Aperture represents the desirable extent for image acquisition, generation, and processing, prior to blanking. For Rec. 601 sampled video, the Production Aperture is 720 pixels x 486 lines for 525-line systems and 720 pixels x 576 lines for 625-line systems [25].

Quarter Common Intermediate Format (QCIF) - A video sampling structure used for video teleconferencing where the luminance channel is sampled at 176 pixels by 144 lines [14].

Rec. 601 - Abbreviation for ITU-R Recommendation BT.601 [13], a common 8-bit video sampling standard that samples the luminance (Y) channel at 13.5 MHz, and the blue and red color difference channels (C_B and C_R) at 6.75 MHz. See section 2 for more information.

Rectangle Coordinates - A rectangular shaped image sub-region that is completely contained within the production aperture and that is specified by four coordinates (top, left, bottom, right). Numbering starts from zero so that the (top, left) corner of the sampled image is (0, 0). See section 2.3.

Reduced-Reference - 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 [15]. 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.

Reframing - The process of reordering two consecutively sampled interlaced fields of processed video into a frame of video. Reframing is necessary when HRCs do not preserve standard interlace field types (e.g., an NTSC field type one is output as an NTSC field type two and vice versa). See section 3.1.2.

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. Also see SROI.

Root Cause Analysis (RCA) - Root cause analysis is objective or subjective analyses used to determine the presence or absence of specific video artifacts (e.g., blurring, tiling, or dropped frames) in the processed video. Root Cause Analysis provides the user with detailed information on the likely cause of quality degradations measured by VQM. RCA lists a percentage for several possible impairments (e.g., jerky motion, blurring, and error blocks), where 100% indicates that all viewers perceived the impairment as a primary artifact, 50% indicates that viewers perceived the impairment as a secondary artifact, and 0% indicates that the artifact was not perceived. RCA gives an estimate of the impairment *type*, as opposed to the *amount* of perceived impairment that is estimated by a VQM model.

Root Mean Square Error (RMSE) - The estimated root mean square error between objective VQM and subjective mean opinion score data, scaled to VQM's nominal output range of zero to one [6].

Scene - A sequence of video frames.

Spatial Information (SI) - A feature based on statistics that are extracted from the spatial gradients (i.e., edges) of an image or video scene. References [3] and [16] provide a definition of SI based on statistics extracted from 3 x 3 Sobel-filtered images [19] while section 4.2.2 of this report provides a definition of SI based on statistics extracted from much larger 13 x 13 edge-filtered images (Figure 27).

Spatial Region of Interest (SROI) - The specific image lattice (specified in Rectangle Coordinates) that is used to calculate the VQM of a video clip. The SROI is a rectangular subset that lies completely inside the Processed Valid Region. For Rec. 601 sampled video, the recommended SROI is 672 pixels x 448 lines for 525-line systems and 672 pixels x 544 lines for 625-line systems, centered within the

Production Aperture. This recommended SROI corresponds to approximately the portion of the video picture that is visible on a monitor, excluding the over-scan area. Also see ROI.

Spatial Registration - The process that is used to estimate and correct for spatial shifts of the processed video sequence with respect to the original video sequence.

Spatial-Temporal (S-T) Sub-Region - A block of image pixels in an original or processed video stream that includes a vertical extent (number of rows), a horizontal extent (number of columns), and a time extent (number of frames). See Figure 25.

Society of Motion Picture and Television Engineers (SMPTE) - An industry-leading society for the motion picture and television industries devoted to advancing theory and application in motion imaging, including film, television, video, computer imaging, and telecommunications. The industry relies on SMPTE to generate standards, engineering guidelines, and recommended practices to be followed by respective field professionals.

Temporal Information (TI) - A feature based on statistics that are extracted from the temporal gradients (i.e., motion) of a video scene. References [3], [16] and section 4.5 of this report all provide definitions of TI based on statistics extracted from simple frame differences.

Temporal Region of Interest (TROI) - The specific time segment, sequence, or subset of frames that is used to calculate a clip's VQM. The TROI is a contiguous segment of frames that lies completely inside the Temporal Valid Region. The maximum possible TROI is the fully registered time segment and contains all temporally registered frames within the TVR. If reframing is required, the processed clip is always reframed, not the original clip.

Temporal Registration - The process that is used to estimate and correct for the temporal shift (i.e., video delay) of the processed video sequence with respect to the original video sequence. Two different algorithms for performing temporal registration are described in this document. One is sequence-based (see section 3.4.1) and the other is frame-based (see section 3.4.2).

Temporal Valid Region (TVR) - The maximum time segment, sequence, or subset of video frames that may be used for calibration and VQM model calculation. Frames outside of this time segment will always be considered invalid.

Uncertainty (U) - The estimated error (plus or minus) in the temporal registration after allowance is made for the best guess of the HRC video delay. See section 3.4.

Valid Region (VR) - The rectangular portion of an image lattice (specified in Rectangle Coordinates) that is not blanked or corrupted due to processing. The Valid Region is a subset of the production aperture of the video standard and includes only those image pixels that contain picture information that has not been blanked or corrupted. See Original Valid Region and Processed Valid Region.

Video Quality Metric (VQM) - An overall measure of video impairment reported by a particular VQM model, either for an individual video clip (Clip VQM), or for an HRC (HRC VQM). 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.

VQM Model - A particular algorithm that is used to compute VQM and that has been specifically optimized to achieve maximum objective to subjective correlation based upon certain optimization criteria, including the range of quality over which the model applies and the speed of computation. This document defines five VQM models: (1) General - optimized using a wide range of video quality and bit rates, (2) Developer - optimized using a wide range of video quality and bit rates with the added constraint of fast computation, (3) Television - optimized for television (e.g., MPEG-2), (4) Videoconferencing - optimized for videoconferencing (e.g., H.261), and (5) PSNR - Peak signal to noise ratio.

VIDEO QUALITY MEASUREMENT TECHNIQUES

Stephen Wolf and Margaret Pinson[*]

Objective metrics for measuring digital video performance are required by Government and industry for specification of system performance requirements, comparison of competing service offerings, service level agreements, network maintenance, and optimization of the use of limited network resources such as transmission bandwidth. To be accurate, digital video quality measurements must be based on the perceived quality of the actual video being received by the users of the digital video system rather than the measured quality of traditional video test signals (e.g., color bar). This is because the performance of digital video systems is variable and depends upon the dynamic characteristics of both the original video (e.g., spatial detail, motion) and the digital transmission system (e.g., bit rate, error rate). The goal of this report is to provide a complete description of the ITS video quality metric (VQM) algorithms and techniques. The ITS automated objective measurement algorithms provide close approximations to the overall quality impressions, or mean opinion scores, of digital video impairments that have been graded by panels of viewers.

Key words: Video, quality, models, metrics, features, parameters, objective, subjective, correlation, reduced-reference, television, videoconferencing, root cause analysis, spatial information (SI), temporal information (TI), impairments, blocking, blurring, frame dropping, peak-signal-to-noise ratio (PSNR), video calibration, spatial registration, temporal registration, gain, contrast, level offset, brightness

[*] The authors are with the Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce, 325 Broadway, Boulder, CO 80305.

1. INTRODUCTION AND OVERVIEW

Digital video systems are replacing existing analog video systems and making possible the creation of many new telecommunication services (e.g., direct broadcast satellite, digital television, high definition television, video teleconferencing, telemedicine, e-commerce) that are becoming an essential part of the U.S. and world economy. Objective metrics for measuring the video performance of these systems are required by Government and industry for specification of system performance requirements, comparison of competing service offerings, service level agreements, network maintenance, and optimization of the use of limited network resources such as transmission bandwidth. To be accurate, digital video quality measurements must be based on the perceived quality of the actual video being received by the users of the digital video system rather than the measured quality of traditional video test signals (e.g., color bar). This is because the performance of digital video systems is variable and depends upon the dynamic characteristics of both the original video (e.g., spatial detail, motion) and the digital transmission system (e.g., bit rate, error rate).

The goal of this report is to provide a complete description of the ITS video quality metric (VQM) algorithms and techniques. These automated objective measurement algorithms provide close approximations to the overall quality impressions, or mean opinion scores, of digital video impairments that have been graded by panels of viewers. Figure 1 gives an overview diagram of the processes required to compute VQM. These processes include sampling of the original and processed video streams (section 2), calibration of the original and processed video streams (section 3), extraction of perception-based features (section 4), computation of video quality parameters (section 5), and calculation of VQM models (section 6). VQM tracks the perceptual changes in quality due to distortions in any component of the digital video transmission system (e.g., encoder, errors in digital channel, decoder).

VQM has been extensively tested on the subjective data sets described in section 7 and the results of these objective to subjective comparisons are provided in section 8. Finally, section 9 describes a set of root cause analysis (RCA) models that can be used to determine the presence of specific digital video artifacts (e.g., blurring, tiling, or dropped frames) [2] in the processed video.

If an ancillary data channel is available between the original and processed ends as shown in Figure 1, the VQM techniques presented here can also be adapted for continuous in-service quality monitoring ([9], [26]-[28], [30]-[34]). Video quality monitoring systems of this type are known as reduced-reference systems [15], since they utilize only a small portion of the reference video stream (i.e., the original video stream for downstream monitoring, the processed video stream for upstream monitoring) to make the quality measurement. The ancillary data channel can be used to transmit certain calibration information as well as the extracted quality features. Using the techniques presented in this document, VQM models can be readily designed to accommodate a range of ancillary data channel bandwidths. The performance of these VQM models (i.e., objective to subjective correlation) will depend upon the bandwidth of the ancillary data channel. VQM models designed for ancillary data channels with very low bandwidths can achieve quite impressive performance [34].

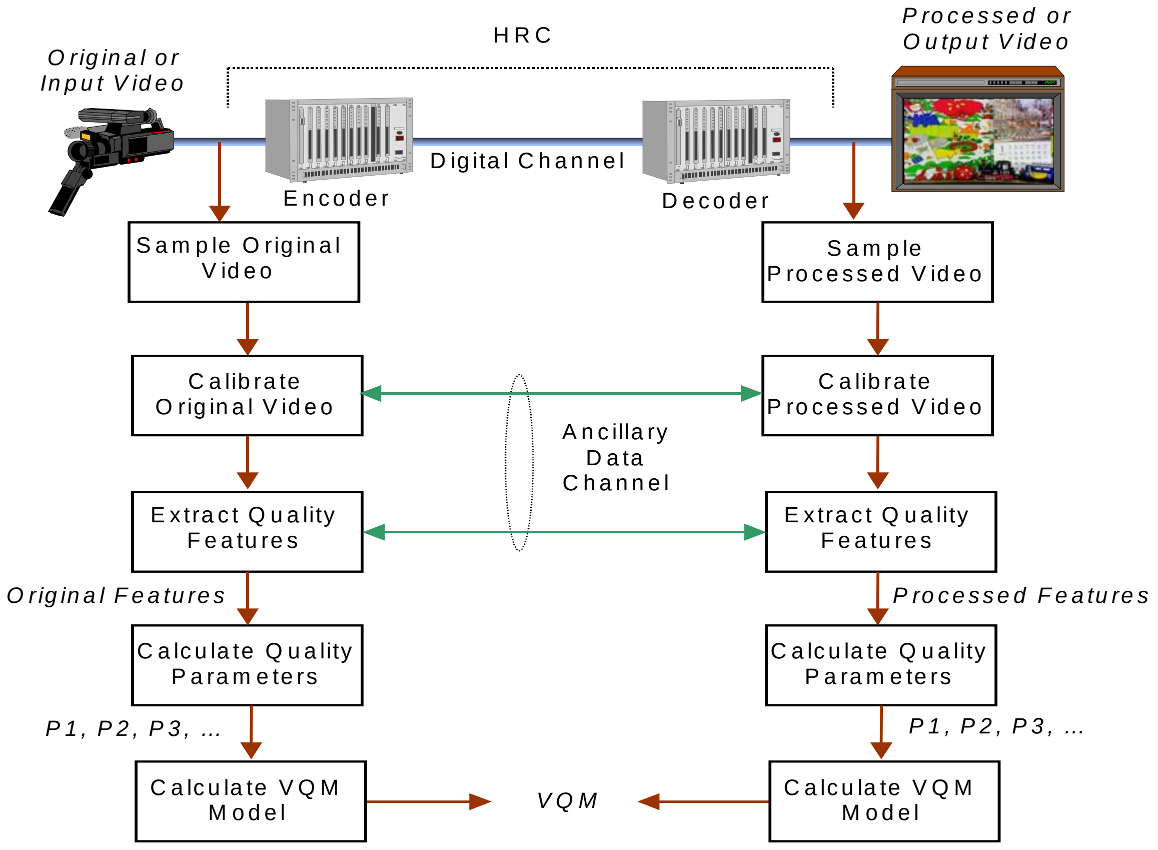

Figure 1. Steps required to compute VQM.

2. SAMPLING

The computer-based algorithms in this document assume that the original and processed video streams are available as digital representations stored on computer media (referred to as a clip in this document). If the video is analog format, one of the most widely used digital sampling standards is ITU-R Recommendation BT.601 [13], which will henceforth be denoted as Rec. 601 for brevity. Composite video such as NTSC must first be converted into component video that contains the following three signals: luminance (Y), blue color difference (C_B), and red color difference (C_R) [13]. Rec. 601 sampling is commonly known as 4:2:2 sampling since the Y channel is sampled at full rate while the C_B and C_R channels are sampled at half rate. Rec. 601 specifies a 13.5 MHz sample rate that produces 720 Y samples per video line. Since there are 486 lines that contain picture information in the NTSC standard, the complete Rec. 601 sampled Y video frame will be 720 pixels by 486 lines. If 8 bits are used to uniformly sample the Y signal, Rec. 601 specifies that reference black (i.e., 7.5 IRE units) be sampled as a “16” and reference white (i.e., 100 IRE units) be sampled as a “235.” Thus, a working margin is available for video signals that exceed the reference black and white levels before they are clipped by the analog to digital converter. The chrominance channels (C_B and C_R) are each sampled at 6.75 MHz such that the first pair of chrominance samples (C_B, C_R) is associated with the first Y luminance sample, the second pair of chrominance samples is associated with the third luminance sample, and so forth. Since the chrominance channels are bipolar, zero signal is sampled as a “128.”

2.1 Temporal Indexing of Original and Processed Video Files

A luminance video frame that results from Rec. 601 sampling will be denoted as **Y**(*t*). The variable *t* is being used here as an index for addressing the sampled frames within the original and processed Big YUV files; it does not denote actual time. If the Big YUV file contains N frames, as shown in Figure 1, *t* = 0 denotes the first frame that was sampled and *t* = (N-1) denotes the last frame that was sampled.

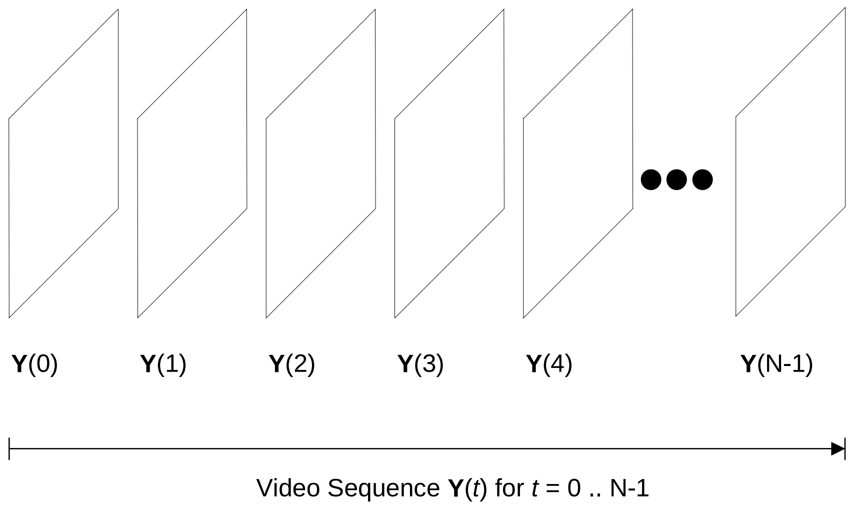

Figure 2. Temporal indexing of frames in Big YUV files.

All the algorithms are written and described from the viewpoint of operation on sampled file pairs: one original video sequence and an associated processed video sequence. To avoid confusion, both files are assumed to be the same length. Furthermore, an initial assumption will be made that the first frame of the original file aligns temporally to the first frame of the processed file, within plus or minus some temporal uncertainty.

For real-time, in-service implementations, this balanced uncertainty presumption can be replaced with a one-sided uncertainty. Causality constrains the range of temporal uncertainty. For example, a processed frame occurring at time *t* = n must come from original frames occurring at or before time *t* = n.

The above assumption regarding original and processed video files (i.e., that the first frames align) is equivalent to selecting the best guess for the temporal delay of the HRC shown in Figure 1. Therefore, the uncertainty that remains in the video delay estimate will be denoted as plus or minus **U**.

2.2 Spatial Indexing of Original and Processed Video Frames

The coordinate system used for the sampled luminance frames is shown in Figure 3. The horizontal and vertical coordinates of the upper left corner of the luminance frames are defined to be (v = 0, h = 0), where the horizontal axis (h) coordinate values increase to the right and the vertical axis (v) coordinate values increase down. Horizontal axis coordinates range from 0 to one less than the number of pixels in a line. Vertical axis coordinates range from 0 to one less than the number of lines in the image, which will be specified in frame lines for progressive systems and either field lines or frame lines for interlace systems. The amplitude of a sampled pixel in **Y**(*t*) at row *i* (i.e., v = *i*), column *j* (i.e., h = *j*), and time *t* is denoted as *Y*(*i*, *j*, *t*).

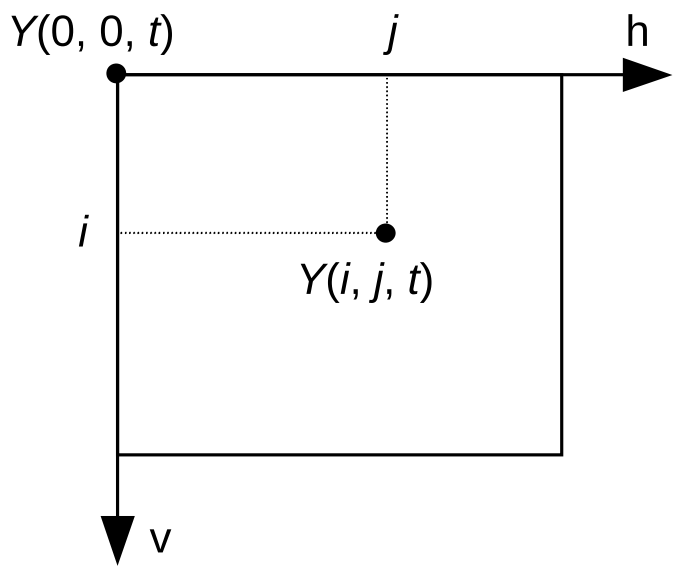

Figure 3. Coordinate system used for sampled luminance Y frames.

A clip of video sampled according to Rec. 601 is stored in “Big YUV” file format, where the Y denotes the Rec. 601 luminance information, the U denotes the blue color-difference information (i.e., C_B in Rec. 601), and the V denotes the red color-difference information (i.e., C_R in Rec. 601). In the Big YUV file format, all the frames are stored sequentially in one large continuous binary file. The image pixels are stored sequentially by video scan line as bytes in the following order: C_{B0}, Y_0, C_{R0}, Y_1, C_{B2}, Y_2, C_{R2}, Y_3, etc., where the numerical subscript denotes the pixel number (pixel replication or interpolation must be used to find the C_B and C_R chrominance samples for Y_1, Y_3, …). This byte ordering is equivalent to that specified in SMPTE 125M [23].

2.3 Specifying Rectangular Sub-Regions

Rectangular sub-regions of a sampled image are used to control the computation of VQM. For instance, VQM may be computed over the valid region of the sampled image or over a user-specified spatial region of interest that is smaller than the valid region. Specification of rectangular sub-regions will use rectangle coordinates defined by the four quantities top, left, bottom, and right. Figure 4 illustrates the specification of a rectangular sub-region for a single frame of sampled video. The red image pixels are included in the sub-region but the black image pixels are excluded. In the calculation of VQM, an image is often divided into a large number of smaller sub-regions that abut. The rectangular sub-region definition used in Figure 4 defines the grid used to display these abutted sub-regions and the math used to extract features from each abutted sub-region.

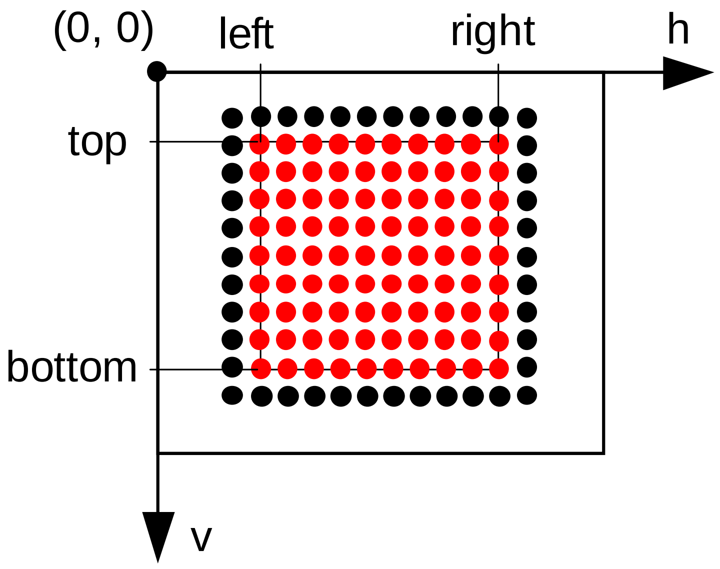

Figure 4. Rectangle coordinates for specifying image sub-regions.

2.4 Considerations for Video Sequences Longer Than 10 Seconds

The video quality measurements in this document were based upon subjective test results that utilized 8 to 10 second video clips (see section 7). When working with longer video sequences, the sequence should be divided into shorter video segments, where each segment is assumed to have its own calibration and quality attributes. Dividing the video stream into overlapping segments and processing each segment independently is one method for emulating continuous quality assessments for long video sequences using the VQM techniques presented herein.

3. CALIBRATION

Four steps are required to properly calibrate the sampled video in preparation for feature extraction. These steps are (1) spatial registration estimation and correction, (2) valid region estimation to limit the extraction of features to those pixels that contain picture information, (3) gain and level offset estimation and correction (commonly known as contrast and brightness), and (4) temporal registration estimation and correction. Step 2 must be performed on both the original and processed video streams. Steps 1, 3, and 4 must be performed on the processed video stream. Normally, the spatial registration, gain, and level offset are constant for a given video system and hence these quantities only need to be calculated once. However, it is common for the valid region and temporal registration to change depending upon scene content. For instance, full screen and letterbox scenes will have different valid regions, and videoconferencing systems often have variable video delays that depend upon scene content (e.g., talking head versus sports action). In addition to the calibration techniques presented here, the reader may also want to examine [7] and [17] for alternate spatial and temporal registration methods.

Calibrating prior to feature extraction means that VQM will not be sensitive to horizontal and vertical shifts of the image, temporal shifts of the video stream that result from non-zero video delays, and changes in image contrast and brightness that fall within the dynamic range of the video sampling unit. While these calibration quantities can have a significant impact on the overall perceived quality (e.g., low contrast images from a video system with a gain of 0.3), the philosophy taken here is to report calibration information separately from VQM. Spatial shifts, valid regions, gains, and offsets can normally be adjusted using good engineering practices, while temporal delays provide important quality information when evaluating two-way or interactive video systems.

All of the video quality features and parameters (sections 4 and 5) assume that only one video delay will be removed to temporally register the processed video sequence (i.e., constant video delay). Some video systems or HRCs delay individual processed frames by different amounts (i.e., variable video delay). For the purposes of this document, all video systems are treated as having a constant video delay. Variations from this delay are considered degradations that are measured by the features and parameters. This approach appears to yield higher correlations to subjective score than video quality measurements based on processed video sequences where variable video delay has been removed. When working with long video sequences (see section 2.4), the sequence should be divided into shorter video segments, where each segment has its own constant video delay. This allows for some delay variation as a function of time. A more continuous estimation of delay variations may be obtained by dividing the sequence into overlapping time segments.

If the HRC being tested also spatially scales the picture or changes its size (e.g., zoom), then an additional step to estimate and remove this spatial scaling would have to be included in the calibration process. Spatial scaling is beyond the scope of this report.

3.1 Spatial Registration

3.1.1 Overview

The spatial registration process determines the horizontal and vertical spatial shift of the processed video relative to the original video. A positive horizontal shift is associated with a processed image that has been moved to the right by that number of pixels. A positive vertical shift is associated with a processed image that has been moved down that number of lines. Thus, spatial registration of interlace video results in three numbers: the horizontal shift in pixels, the vertical NTSC field one shift in field lines, and the vertical NTSC field two shift in field lines. Spatial registration of progressive video results in two numbers: the horizontal shift and the vertical shift in frame lines. The accuracy of the spatial registration algorithm is to the nearest pixel for horizontal shifts and to the nearest line for vertical shifts. After the spatial registration has been calculated, the spatial shift is removed from the processed video stream (e.g., a processed image that was shifted down is shifted back up). For interlace video, this may include reframing of the processed video stream as implied by comparison of the vertical field one and two shifts.

When operating on interlace video, all operations will consider video from each field separately; when operating on progressive video, all operations will consider the entire video frame simultaneously. For simplicity, the spatial registration algorithm will first be entirely described for interlace video, this being the more complicated case. The modifications needed to operate on progressive video are identified in section 3.1.7.

Spatial registration must be determined before processed valid region (PVR), gain and level offset, and temporal registration. Specifically, each of those quantities must be computed by comparing original and processed video content that has been spatially registered. If the processed video stream were spatially shifted with respect to the original video stream and this spatial shift were not corrected, then these estimates would be corrupted because they would be based on dissimilar video content. Unfortunately, spatial registration cannot be correctly determined unless the PVR, gain and level offset, and temporal registration are also known. The interdependence of these quantities produces a “chicken or egg” measurement problem. Calculation of the spatial registration for one processed field requires that one know the PVR, gain and level offset, and the closest matching original field. However, one cannot determine these quantities until the spatial shift is found. A full exhaustive search over all variables would require a tremendous number of computations if there were wide uncertainties in the above quantities.

The solution presented here performs an iterative search to find the closest matching original field for each processed field. This search includes iteratively updating estimates for PVR, gain and level offset, and temporal registration. For some processed fields, however, the spatial registration algorithm could fail. Usually, when the spatial registration is incorrectly estimated for a processed field, the ambiguity is due to characteristics of the scene. Consider, for example, a digitally created interlace scene containing a pan to the left. Because the pan was computer generated, this scene could have a horizontal pan of exactly one pixel every field. From the spatial registration search algorithm's point of view, it would be impossible to differentiate between the correct spatial registration computed using the matching original field, and a two pixel horizontal shift computed using the field that occurs two fields prior to the matching original field. For another example, consider an image consisting entirely of digitally perfect black and white vertical lines. Because the image contains no horizontal lines, the vertical shift is entirely ambiguous. Because the pattern of vertical lines repeats, the horizontal shift is ambiguous, two or more horizontal shifts being equally likely.

Therefore, the iterative search algorithm should be applied to a sequence of processed fields. The individual estimates of spatial shifts from multiple processed fields can then be used to produce a more robust estimate. Spatial shift estimates from multiple sequences or scenes may be further combined to produce an even more robust estimate for the HRC being tested; assuming that the spatial shift is constant for all scenes passing through the HRC.

3.1.2 Interlace Issues

Vertical spatial registration of interlaced video is a greater challenge than progressive video, since the spatial registration process must differentiate between field one and field two. There are three vertical shift conditions that must be differentiated to obtain the correct vertical shift registration for interlaced systems: vertical field one equals vertical field two, vertical field one is one less than vertical field two, and everything else.

Some HRCs shift field one and field two identically, yielding a vertical field one shift that is equal to the vertical field two shift. For HRCs that do not repeat fields or frames (i.e., HRCs that transmit the full frame rate of the video standard), this condition means that what was a field one in the original video stream is also a field one in the processed video stream, and what was a field two in the original is also a field two in the processed.

Other HRCs reframe the video, shifting the sampled frame by an odd number of frame lines. What used to be field one of the original becomes field two of the processed, and what used to be field two of the original becomes the next frame's field one. Visually, the displayed video appears correct since the human cannot perceive a one-line frame shift of the video.

As shown in Figure 5, NTSC field one occurs earlier in time, starts with frame line one, and contains all odd-numbered frame lines. NTSC field two occurs later in time, starts with frame line zero (topmost frame line), and contains all even-numbered frame lines. When original field two is moved into the next frame's field one, the top line of the field moves from original field-two frame line 0 to processed field-one frame line 1. In field line numbering, the top line stays in field line 0, so processed field one has a zero vertical shift (since vertical shifts are measured for each field using field lines). When original field one is moved to that frame's field two, the top line of the field moves from original field one, frame line 1 to processed field two, frame line 2. In field line numbering, the top line moves from field line 0 to field line 1, so processed field two has a one field line vertical shift. The general rule is that when the field-two vertical shift (in field lines) is one greater than the field-one vertical shift, reframing has occurred.

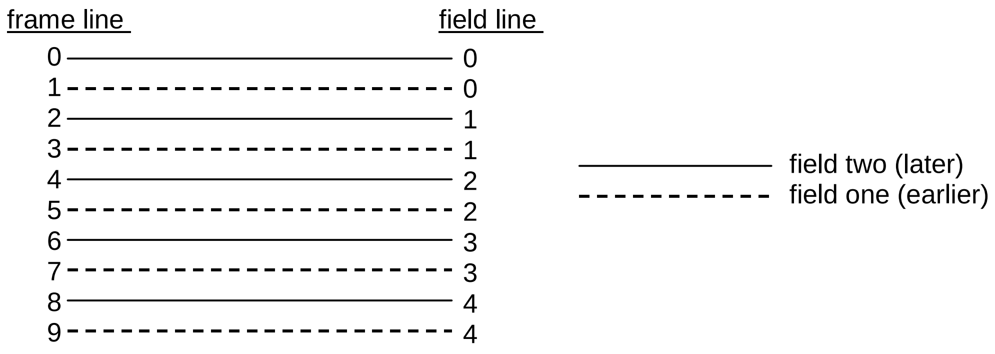

Figure 5. Diagram depicting NTSC interlaced fields and frame/field line numbering scheme.

If the field-two vertical shift is not equal to or one more than the field-one vertical shift, the HRC has corrupted the proper spatial sampling of the two interlaced fields of video and the resulting video will appear to "bob" up and down. Such an impairment is both obvious and annoying to the viewer, and hence seldom occurs in practice since the HRC designer discovers and corrects the error. Therefore, most of the time, spatial registration simplifies into two common patterns. In systems that do not reframe, field-one vertical shift equals field-two vertical shift; and in systems that reframe, field-one vertical shift plus one equals field-two vertical shift.

Additionally, notice that spatial registration includes some temporal registration information, specifically whether the video has been reframed or not. The temporal registration process may or may not be able to detect reframing, but even if it can, reframing is inherent to the spatial registration process. Therefore, spatial registration must be able to determine whether the processed field being examined best aligns with an original field one or field two. The spatial registration for each field can only be correctly computed when the processed field is compared to the original field that created it. Aside from the reframing issue, use of the wrong original field (field one versus field two) can produce spatial registration inaccuracies due to the inherent differences in the spatial content of the two interlaced fields.

3.1.3 Required Inputs to the Spatial Registration Algorithm

This section gives a list of the input variables that are required by the spatial registration algorithm. These inputs specify items such as the range of spatial shifts and temporal fields over which to search. If these ranges are overly generous, the speed of convergence of the iterative search algorithm used to find the spatial shift may be slow and the probability of false spatial registration for scenes with repetitive content is increased (e.g., someone waving their hand). Conversely, if these ranges are too restrictive, the search algorithm will encounter, and *slowly* extend, the search range boundaries with successive iterations. While this built-in search intelligence is useful if the user mis-guesses the search uncertainties by a small amount, the undesirable side effect is to dramatically increase run time when the user mis-guesses by a large amount. Alternatively, the search algorithm may fail to find the correct spatial shift in this case.

3.1.3.1 Expected Range of Spatial Shifts

The expected range of spatial shifts for 525-line video sampled according to Rec. 601 lies between ±20 pixels horizontally and ±12 *field* lines vertically. This range of expected shifts has been determined empirically by processing video data from hundreds of HRCs. The expected range of spatial shifts for video sampled according to other formats smaller than Rec. 601 (e.g., CIF), is presumed to be half of that observed for 525-line systems. This search algorithm should operate correctly, albeit a bit slower, when the processed field has spatial shifts that lie outside of the expected range of spatial shifts. This is because the search algorithm will expand the search beyond the expected range of spatial shifts when warranted. Excursions exceeding 50% of the expected range, however, may report a failure to find the correct spatial registration.

3.1.3.2 Temporal Uncertainty

The user must also specify the temporal registration uncertainty, i.e., the range of original fields to examine for each processed field. This temporal uncertainty is expressed as a number of frames before and after the default temporal registration. If the original and processed video sequences are stored as files, then a reasonable default temporal registration is to assume that the first frames in each file align. The temporal uncertainty that is specified should be large enough to include the actual temporal registration. An uncertainty of plus or minus one second (30 frames for 525-line NTSC video) should be sufficient for most video systems. HRCs with long video delays may require a larger temporal uncertainty. The search algorithm may examine temporal registrations outside of the specified uncertainty range when warranted (e.g., when the farthest original field is chosen as the best temporal registration).

3.1.3.3 Processed Valid Region (PVR) Guess

The processed valid region (PVR) guess specifies the portion of the processed image that has not been blanked or corrupted due to processing, presuming no spatial shift has occurred (since the spatial shift has not yet been measured). Although the PVR guess could be determined empirically, a user-specified PVR guess that excludes the over-scan is a good choice. In most cases this will eliminate invalid video from being used in the spatial registration algorithm. For 525-line / NTSC video sampled according to Rec. 601, the over-scan covers approximately 18 frame lines at the top and bottom of the frame, and 22 pixels at the left and right sides of the frame. For 625-line / PAL video sampled according to Rec. 601, the over-scan covers approximately 14 frame lines at the top and bottom of the frame, and 22 pixels at the left and right sides of the frame. When using other image sizes (e.g., CIF), a reasonable default PVR for these image sizes should be selected.

3.1.4 Sub-Algorithms Used by the Spatial Registration Algorithm

The spatial registration algorithm makes use of a number of sub-algorithms, including estimation of gain and level offset, and the formula used to determine the closest matching original field for a given processed field. These sub-algorithms have been designed to be computationally efficient, since they must be performed many times by the iterative search algorithm.

3.1.4.1 Region of Interest (ROI) Used by All Calculations

All field comparisons made by the algorithm will be between spatially shifted versions of a ROI extracted from the processed video (to compensate for the spatial shifts introduced by the HRC) and the corresponding ROI extracted from the original video. The spatially shifted ROI from the processed video will be denoted as PROI (i.e., processed ROI) and the corresponding ROI from the original video will be denoted as OROI (original ROI). The rectangle coordinates that specify OROI are fixed throughout the algorithm and are chosen to give the largest possible OROI that meets both of the following requirements:

- The OROI must correspond to a PROI that lies within the processed valid region (PVR) for all possible spatial shifts that will be examined.
- The OROI is centered within the original image.

3.1.4.2 Gain and Level Offset

The following algorithm is used to estimate the gain of the processed video. The processed field being examined is shift-corrected using the current estimate for spatial shift. After this shift-correction, a PROI is selected that corresponds to the fixed OROI determined in section 3.1.4.1. Next, the standard deviation of the luminance (Y) pixels from this PROI and the standard deviation of the luminance pixels (Y) from the OROI are calculated. Gain is then estimated as the standard deviation of PROI pixels divided by the standard deviation of OROI pixels.

The reliability of this gain estimate improves as the algorithm iterates toward the correct spatial and temporal shift. A gain of 1.0 (i.e., no gain correction) may be used during the first several iteration cycles. The above gain calculation is sensitive to impairments in the processed video such as blurring. However, for the purposes of spatial registration, this gain estimate is appropriate because it makes the processed video look as much like the original video as possible. To remove gain from the processed field, each luminance pixel in the processed field is divided by the gain.

There is no need to determine or correct for level offset, since the spatial registration algorithm's search criteria is unaffected by level offsets (see section 3.1.4.3).

3.1.4.3 Formulae Used to Compare PROI with OROI

After correcting the PROI for gain[1] (section 3.1.4.2), the standard deviation of the (OROI-PROI) difference image is used to choose between two or more spatial shifts or temporal shifts. The gain estimate from the previous best match is used to correct the PROI gain. To search among several spatial shifts (with temporal shift held constant), compute the standard deviation of the (OROI-PROI) difference image for several PROI generated using different spatial shifts. For a given processed field, the combination of spatial and temporal shifts that produce the smallest standard deviation (i.e., most cancellation with the original) is chosen as the best match.

3.1.5 Spatial Registration Using Arbitrary Scenes

Spatial registration of a processed field from a scene must examine a plurality of original fields and spatial shifts since both the temporal shift (i.e., video delay) and the spatial shift are unknown. As a result, the search algorithm is complex and computationally intense. Furthermore, the scene content is arbitrary, and so the algorithm may find an incorrect spatial registration (see section 3.1.1). Therefore, the prudent course is to compute the spatial registration of several processed fields from several different scenes that have all been passed through the same HRC, and combine the results into one robust estimate of spatial shift. A single HRC should have one constant spatial registration. If not, these time varying spatial shifts would be perceived as an impairment (e.g., the video would bounce up and down or from

[1] Gain compensation can sometimes be omitted to decrease the computational complexity. However, omission of gain correction is only recommended during early stages of the iterative search algorithm, where the goal is to find the approximate spatial registration (e.g., see sections 3.1.5.2 and 3.1.5.3).

side to side). This section describes the spatial registration algorithm from the bottom up, in that the core components of the algorithm are described first, and then their application for spatial registering scenes and HRCs is described.

3.1.5.1 Best Original Field Match in Time

When spatially registering using scene content, the algorithm must find the original field that most closely matches the current processed field. Unfortunately, that original field may not actually exist. For example, a processed field may contain part of two different original fields since it may have been interpolated from other processed fields. The current estimate of the best original field match (i.e., that original field that most closely matches the current processed field) is kept at all stages of the search algorithm.

An initial assumption is made that the first field of the processed Big YUV file aligns with the first field of the original Big YUV file, within plus or minus some temporal uncertainty in frames (denoted here as **U**). For each processed field that is examined by the algorithm, there must be a buffer of **U** original frames before and after this field. Thus, the algorithm starts examining processed fields that are **U** frames into the file, and examines every frequencyth frame thereafter (denoted here as **F**), stopping **U** frames before the end of the file.

The final search results from the previous processed field (gain, vertical and horizontal shift, temporal shift) are used to initialize the search for the current processed field. The best original field match to the current processed field is computed assuming a constant video delay. For example, if processed field **N** was found to best align with original field **M** in the Big YUV files, then processed field **N+F** would be assumed to be best aligned to original field **M+F** at the start of the search.

3.1.5.2 Broad Search for the Temporal Shift

A full search of all possible spatial shifts across the entire temporal uncertainty for each processed field would require a large number of computations. Instead, a multi-step search is used, where the first step is a broad search, over a very limited set of spatial shifts, whose purpose is to get close to the correct matching original field.

For the selected processed frame, this broad search examines NTSC field one of this frame (see Figure 5) and considers only those original fields of NTSC field type one that are spaced two frames apart (i.e., four fields apart) across the entire range of plus and minus the temporal registration uncertainty. The broad search considers the following four spatial shifts of the processed video: no shift, eight pixels to the left, eight pixels to the right, and eight field lines up (see Figure 6). In Figure 6, positive shifts mean the processed video is shifted down and to the right with respect to the original video. The "eight field lines down" shift is not considered because empirical observations have revealed that very few video systems move the video picture down. The previous best estimate for spatial shift (i.e., from a previously processed field) is also included as a fifth possible shift when it is available. The closest matching original field to the selected processed field is found using the comparison technique described in section 3.1.4.3. The temporal shift implied by the closest matching original field becomes the starting point for the next step of the algorithm, a broad search for the spatial shift (section 3.1.5.3). According to the coordinate system in Figure 3, a positive temporal shift means that the processed video has been shifted in the positive time direction (i.e., the processed video is delayed with respect to the original video). With respect to the original and processed Big YUV files, a positive field shift thus means that fields must be discarded from the beginning of the processed Big YUV file while a negative field shift means that fields must be discarded from the beginning of the original Big YUV file.

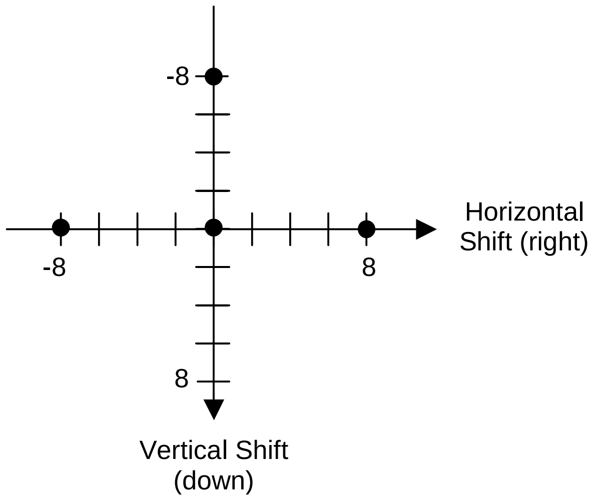

Figure 6. Spatial shifts considered by the broad search for the temporal shift.

3.1.5.3 Broad Search for the Spatial Shift

Using the temporal registration found by the broad search for temporal shift (section 3.1.5.2), a broad search for the correct spatial shift is now performed using a more limited range of original fields. The range of original fields that are considered for this search include the best matching original field of NTSC field type one (from section 3.1.5.2) and the four next closest original fields that are also of NTSC field type one (NTSC field type ones from the 2 frames before and after the best matching original field). The broad search for spatial shift covers the range of spatial shifts given in Figure 7. Notice that fewer downward shifts are considered (as in section 3.1.5.2), since these are less likely to be encountered in practice. The set of spatial shifts and original fields is searched using the comparison technique described in section 3.1.4.3. The resulting best temporal and spatial shifts now become the improved estimates for the next step of the algorithm given in section 3.1.5.4.

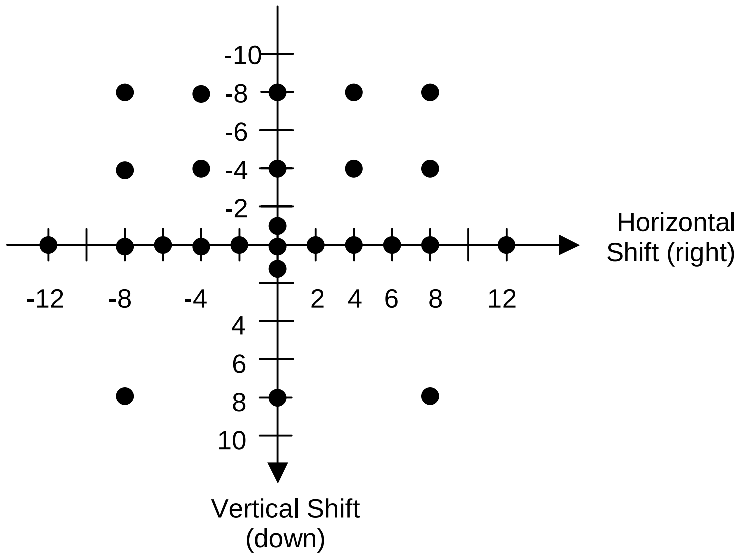

Figure 7. Spatial shifts considered by the broad search for the spatial shift.

3.1.5.4 Fine Search for the Spatial-Temporal Shift

The fine search includes a much smaller set of shifts centered around the current spatial registration estimate and just five fields centered around the current best matching original field. Thus, if the best matching original field were an NTSC field type one, the search would include three NTSC field type ones and the two NTSC field type twos. The spatial shifts that are considered include the current shift estimate, all eight shifts that are within one pixel or one line of the current estimate, eight shifts that are two pixels or two lines from the current shift estimate, and the zero shift condition (see Figure 8). In the example shown in Figure 8, the current spatial shift estimate for the processed video is a shift of 7 field lines up and 12 pixels to the right of the original video. The set of spatial shifts shown in Figure 8 form a near-complete local search of the spatial registrations near the current spatial registration estimate. The zero shift condition is included as a safety check that helps prevent the algorithm from wandering and converging to a local minimum. The set of spatial shifts and original fields is thoroughly searched using the comparison technique described in section 3.1.4.3. The resulting best temporal and spatial shifts now become the improved estimates for the next step of the algorithm given in section 3.1.5.5.

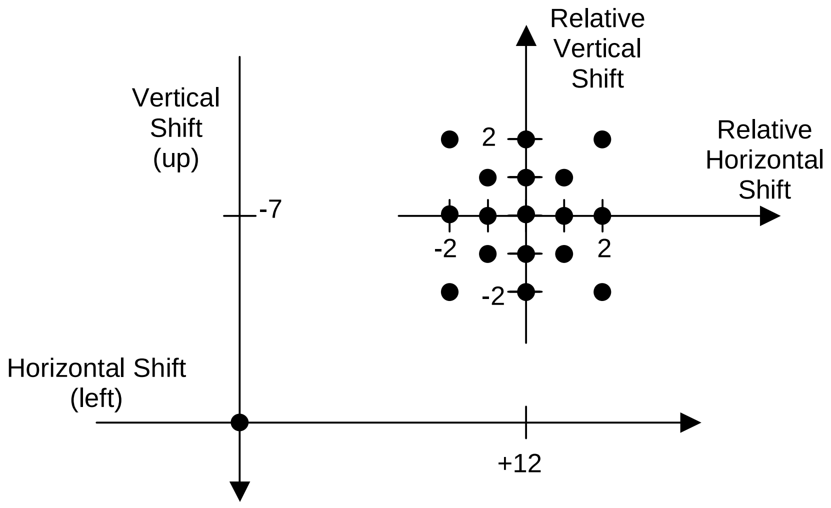

Figure 8. Spatial shifts considered by the fine search for the spatial shift.

3.1.5.5 Repeated Fine Searches

Iteration through the fine search of section 3.1.5.4 will move the current estimate for spatial shift a little closer to either the actual spatial shift or (more rarely) a false minimum. Likewise, one iteration through the fine search will move the current estimate for the best-aligned original field either a little closer to the actual best-aligned original field or (more rarely) a little closer to a false minimum. Thus, each fine search will move these estimates closer to a stable value. Because fine searches examine a very limited area spatially and temporally, they must be performed repetitively to assure that convergence has been reached. When gain compensation is being used, the processed field's gain is estimated anew between each fine search (see section 3.1.4.2).

Repeated fine searches are performed on the processed field (section 3.1.5.4) until the best spatial shift *and* the original field associated with that spatial shift remain unchanged from one search to the next. Repeated fine searches are stopped if the algorithm is alternating between two spatial shifts (e.g., a horizontal shift 3 and then a horizontal shift 4, with everything else remaining the same). This alternation is indicated when the current best estimate for spatial shift *and* the original field associated with that spatial shift, are identical to those found two iterations ago.

Sometimes the repeated search algorithm fails to converge. If the algorithm fails to converge within some requested maximum number of iterations, the iterative search algorithm is terminated and a "failure to find shift" condition is reported for that processed field. This special case does not normally pose a problem because multiple processed fields are examined for each scene (section 3.1.5.6) and multiple scenes are examined for each HRC (section 3.1.5.7).

3.1.5.6 Algorithm for One Scene

An initial baseline (i.e., starting) estimate for vertical shift, horizontal shift, and temporal registration is computed without any gain compensation as follows. The first temporal uncertainty (**U**) processed frames in the Big YUV file are skipped. A broad search for the temporal shift is performed on the next processed field of NTSC field type one (section 3.1.5.2). Notice that this broad search will search the first **U***2 + 1 frames of the original video sequence for an NTSC field type one that best aligns. Then, a broad search for the spatial shift is performed centered on this best-aligned original field (section 3.1.5.3). Next perform up to five fine spatial-temporal searches to fine-tune the spatial and temporal estimates (sections 3.1.5.4 and 3.1.5.5). If these repeated fine searches fail to find a stable result, discard this processed field from consideration. Repeat the above procedure every frequencyth (F) frame until an original field of NTSC field type one is found that produces stable results. The baseline estimate will be updated periodically, as described below.

The spatial shift estimates are calculated for both NTSC field types of a frame in the processed Big YUV file as follows. Using the baseline estimate as a starting point, perform up to three repeated fine searches on the first processed field of NTSC field type one. If the baseline estimate is correct or very nearly correct, the repeated fine searches will yield a stable result. If so, the spatial shift and temporal delay for that processed field are stored in an array that is dedicated to storing the field one results. If a stable result is not found, most likely the spatial shift is correct but the temporal shift estimate is off (i.e., the current estimate of temporal shift is more than two frames away from the true temporal shift). So a broad search for the temporal shift is conducted that includes the current best estimate of spatial shift. This broad search will normally correct the temporal delay estimate. When the broad search for the temporal shift completes, its output is used as the starting point, and up to five repeated fine searches are performed. If this second repeated fine search fails to find a stable result, then report a failed spatial registration for this frame (i.e., both NTSC field type one and NTSC field type two). If a stable result is found from this second search, then the spatial shift and temporal delay for that field are stored in the field one array. Also, the spatial shift and temporal delay used as the starting point for the next processed field of NTSC field type one are updated (i.e., for the first processed field, the baseline results are used and after that, the last stable result is used). After the spatial shift has been estimated for the first processed field of NTSC field type one, the spatial shift for the first processed field of NTSC field type two is estimated. Using the field one spatial results as the starting point, the same steps are used to find the field-two spatial shift (i.e., the three fine searches, and if needed a broad search for the temporal shift followed by five repeated fine searches). If a stable result is found for field two, store the vertical and horizontal shift for field two in a different array that is dedicated to storing field-two results.

The procedure described in the above paragraph is applied to estimate the spatial shift of both NTSC field types of each frequencyth (**F**) frame in the Big YUV file that contains the processed video. The first temporal uncertainty (**U**) processed frames in the Big YUV file are skipped. This sequence of estimates is then used to compute robust estimates of the spatial shift for each NTSC field type for the scene being examined. The vertical field-one shift results from each frame are sorted, and the 50th percentile retained as the overall vertical field-one shift. Likewise, the vertical field-two shift results from each frame are sorted, and the 50th percentile retained as the overall vertical field-two shift. The horizontal field-one shift results from each frame are sorted, and the 50th percentile retained as the overall horizontal shift. Any difference between field-one and field-two horizontal shift is most likely due to a sub-pixel horizontal shift (e.g., a horizontal shift of 0.5 pixels). Sub-pixel horizontal shifts will produce estimates that include both of the two closest shifts. Using the 50th percentile point allows the most likely horizontal shift to be chosen, which produces a spatial registration accuracy that is good to the nearest 0.5 pixels.[2]

3.1.5.7 Algorithm for One HRC

If several scenes have been passed through the same HRC, the spatial registration results for each scene should be identical. Thus, filtering results obtained from multiple scenes can increase the robustness and accuracy of the spatial shift measurements. The overall HRC spatial registration results can then be used to compensate all of the processed video for that HRC.

3.1.5.8 Comments on Algorithm

Some video scenes are simply not well suited for estimating spatial registration. The described algorithm will sometimes locate a false minimum. Other times, the algorithm will wander between multiple solutions and never reach a stable result. For these reasons, it is advisable to examine multiple images within the same scene and to median filter (i.e., sort results from low to high and select the 50th percentile point) these results across different scenes. The spatial registration by scenes algorithm is a heuristic algorithm that utilizes patterns of spatial shifts that have been observed from a sampling of video systems. These assumptions may be incorrect for some systems, causing the algorithm to find an incorrect spatial shift. However, failure of the algorithm tends to produce spatial shifts that are inconsistent from frame to frame and from scene to scene (i.e., when the algorithm fails, it normally produces a scattering of results). When the algorithm outputs the same or very similar spatial shifts for each scene, a high degree of confidence is indicated. When the individual field results for a scene wander, a low degree of confidence is indicated.

3.1.6 Spatial Registration Using Special SMPTE Color Bars

When an artificial test scene can be introduced into the HRC, computation of spatial registration can be simplified considerably. For instance, a static or motionless test scene eliminates the need to consider multiple original frames, because all original frames are identical. This substantially improves the speed of spatial registration, since one no longer has to perform a temporal search. In addition, the artificial scene content can be developed specifically for the purpose of spatial registration; this eliminates ambiguous spatial registrations that might result from using arbitrary scenes (see section 3.1.1).

Figure 9 demonstrates a special SMPTE color bar pattern that can be used for spatial registration of Rec. 601 video scenes.[3] Appendix A includes the MATLAB[4] code that was used to generate the special SMPTE color bar in Rec. 601 format. This test signal can be passed through the HRC along with the actual video scenes. The special SMPTE color bar was designed with several desirable characteristics

[2] Spatial registration to the nearest 0.5 pixels is sufficient for the video quality measurements described in this document. The one exception to this is the peak-signal-to-noise (PSNR) metric for high quality video systems (see section 6.5). Sub-pixel spatial registration techniques are beyond the scope of this document.

[3] The special SMPTE color bar described in this report is U.S. patent pending.

[4] MATLAB is a registered trademark of MathWorks, Inc.

that facilitate estimation of spatial shift. First, the image is fairly simple and entirely still. Therefore, most HRCs will transmit the signal with high quality. Second, the standard deviation of the (OROI-PROI) difference image increases steadily in value as the current spatial shift deviates further from the true spatial shift. In other words, the error function used in the search slants downward toward the true minimum. Third, the diagonal pattern may be used to differentiate between NTSC field one and NTSC field two of the original, which is required to detect reframing of the processed video (section 3.1.2). This section presents a simplified version of the spatial registration algorithm of section 3.1.5 that has been optimized for the special SMPTE color bar.

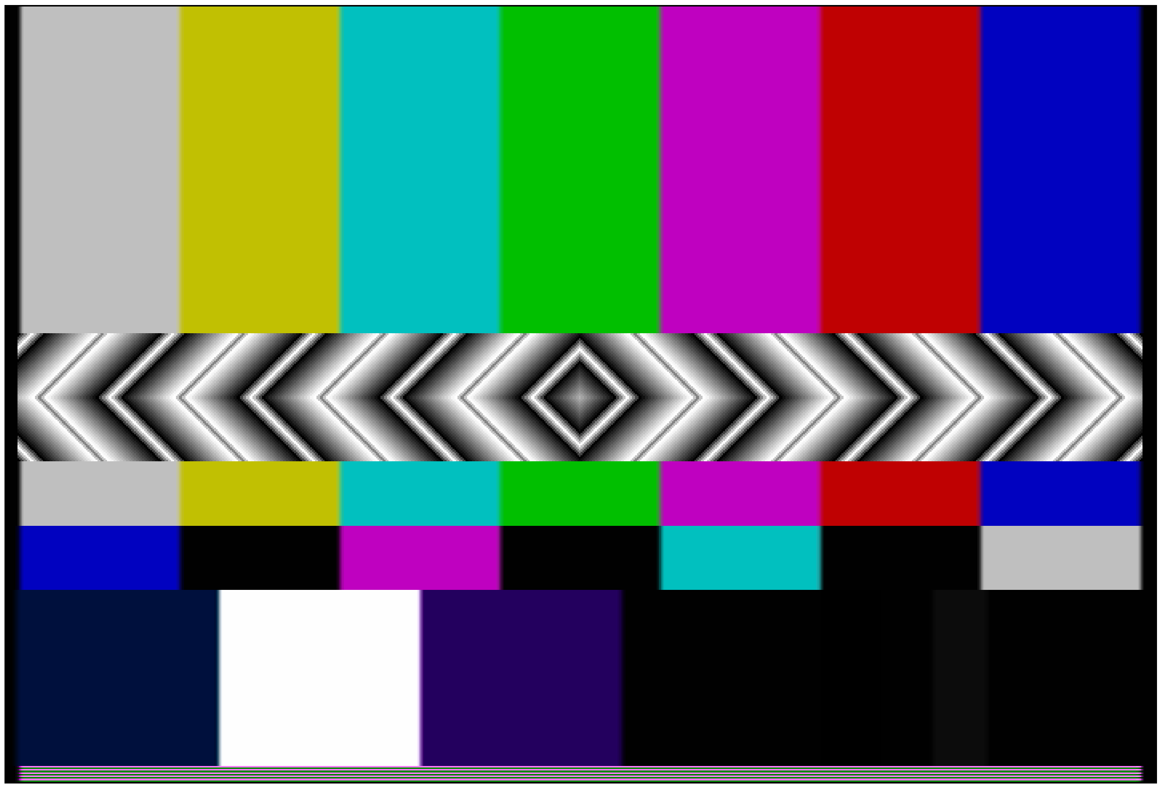

Figure 9. Special SMPTE color bar for spatial registration.

3.1.6.1 Broad Search for the Spatial Shift

The following broad search for the spatial shift takes advantage of the known qualities of the static test image in Figure 9. Given a new processed field, the first step of the algorithm is to do a broad search within the expected range of spatial shifts. The output from this broad search will be a spatial shift that is close to the correct spatial shift. The broad search must consider both original NTSC field one and original NTSC field two. For 525-line and 625-line video sampled according to Rec. 601, twenty-eight spatial shifts are applied to the processed video as shown in Figure 10. These spatial shifts cover the expected range of spatial shifts, with an emphasis on the more commonly observed ones. The shape of the error function for the static test signal also determines the placement of the spatial shifts in Figure 10. The formula described in section 3.1.4.3 is used to determine the best spatial shift. After a single broad search for the spatial shift, the current estimate for spatial shift is close to the true spatial shift.

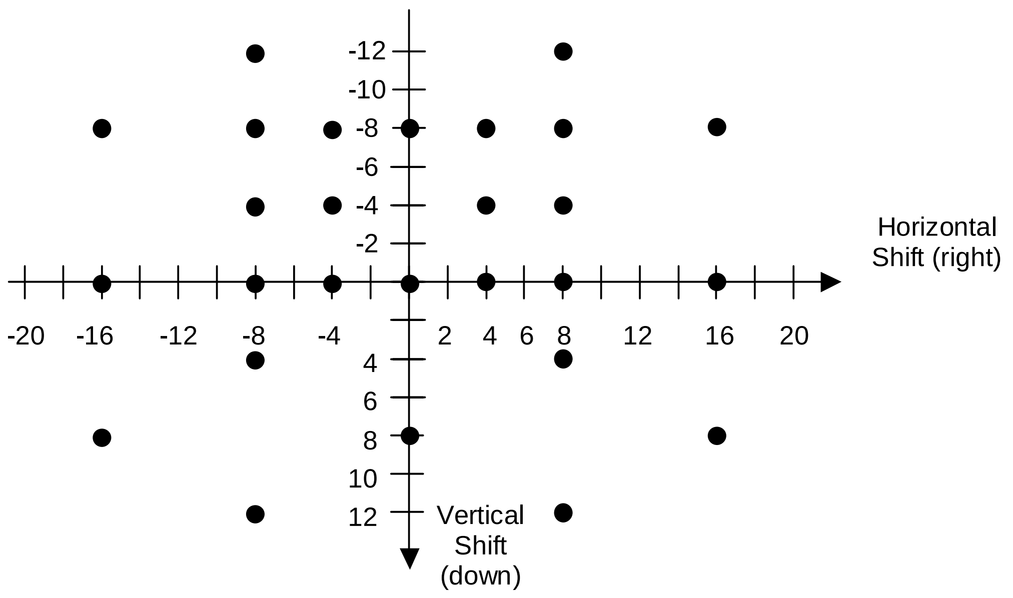

Figure 10. Spatial shifts considered by the broad search for the spatial shift.

3.1.6.2 Fine Search for the Spatial Shift

When the broad search for the spatial shift is complete (section 3.1.6.1), the algorithm has an estimate for spatial registration that is reasonably close to the correct spatial shift. The search space for the fine search includes the following dimensions: (1) vertical shift, (2) horizontal shift, and (3) NTSC field one versus NTSC field two. Due to characteristics of the static test image (section 3.1.6) and the search criterion (section 3.1.4.3), the algorithm only needs to consider shifts close to the current estimate. Therefore, the fine search for spatial shift includes a much smaller set of spatial shifts that are centered on the current spatial registration estimate. These shifts include the current shift estimate, all eight shifts that are within one pixel or one line, and eight shifts that are two pixels or two lines from the current shift estimate. This forms a sufficient local search of the spatial registrations that are near the current spatial registration estimate. As before, the zero shift condition is included as a safety precaution. Figure 11 illustrates the search space when the current spatial registration estimate for the processed video is 12 pixels to the right and 7 pixels up from the original video. The set of spatial shifts and original fields is thoroughly searched using the comparison technique described in section 3.1.4.3. The resulting spatial shifts now become the improved estimates for the next step of the algorithm given in section 3.1.6.3.

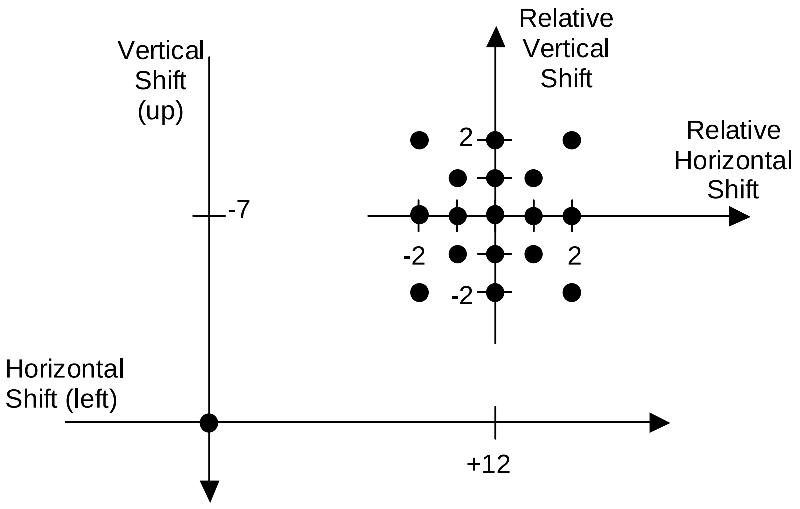

Figure 11. Spatial shifts considered by the fine search for the spatial shift.

3.1.6.3 Repeated Fine Searches

Given a starting spatial shift from section 3.1.6.2, alternately perform fine searches from section 3.1.6.2 and update the gain estimation from section 3.1.4.2. The processed video's gain estimate is removed during fine searches. Stop when the spatial shift *and* the original field associated with that spatial shift remain the same from one search to the next. Additionally, stop if the algorithm is alternating between two spatial shifts. This alternation is indicated when the current best estimate for spatial shift *and* the original field associated with that spatial shift are identical to that found two iterations ago. Accept that final spatial shift as the spatial registration for this processed field. Otherwise, stop if seven fine searches fail to reach a stable result (i.e., the spatial shift cannot be found for this field).

3.1.6.4 Algorithm for One Color Bar Scene

The spatial registration search algorithm for special color bars from an interlaced system is as follows. Choose an NTSC field one of the processed video sequence that is close to the middle of the scene. This image is unlikely to contain transitional effects from the scene cut before or after the static test image. Conduct a broad search for the spatial shift on this processed field (section 3.1.6.1). Then, repeatedly perform up to seven fine searches (sections 3.1.6.2 and 3.1.6.3). If a stable result cannot be found, repeat the above for another processed field of NTSC field type one until a stable result is found. Record the following information as the baseline for processed field one: the spatial shift and whether this spatial shift was associated with original field one or original field two. Choose the processed field two immediately following the field one used above. Using the baseline spatial shift for field one as a starting point, repeatedly perform fine searches. If a stable result cannot be found, repeat this for another processed field two until a stable result is found. Record the following information as the baseline for processed field two: the spatial shift and whether this spatial shift was associated with original field one or original field two.

By applying the algorithm to several different frames of the special color bar image, a more robust measurement of spatial shift can be obtained. The field one and field two baseline measurements are used as an initial starting point for the fine spatial searches. Examine the first processed frame in the video sequence, and all processed frames that fall at a constant interval thereafter (e.g., every frequencyth frame). For each field one and field two of these frames, repeatedly perform up to seven fine searches.

Record all the stable results found by the repeated fine searches. As in section 3.1.5.6, robust measures of field one vertical shift, field two vertical shift, and horizontal shift are found by taking the 50th percentile of all sorted results.

3.1.6.5 Comments on Algorithm

The broad search for the spatial shift considers 28 shifts for each of two fields for a total of 56 combinations. The fine search considers 17 or 18 shifts for each of two fields, for a total of 34 or 36 combinations. Performing the broad search only once means that, once the baseline shifts have been computed, most fields will locate the minimum shift in at best one iteration and at worst seven iterations. If the baseline spatial shift is absolutely correct, all subsequent searches will very likely require only one fine search.

3.1.7 Spatial Registration of Progressive Video

Spatial registration of progressive video follows the same steps as the interlace algorithms, with minor modifications. Where the interlace algorithms operate on field one and field two separately, the progressive algorithm operates on frames. Thus, all mentions of field two are ignored and, with the exception of the fine searches, the range of vertical shifts is doubled.

The modification of the vertical shift range is most important for the broad spatial shift. When doing a broad search for spatial shift (section 3.1.5.3 and section 3.1.6.1) the numbers on the vertical axis in Figure 7 and Figure 10 must be doubled (e.g., +8 becoming +16 and –4 becoming –8).[5] In addition, for progressive CIF and QCIF images, the horizontal and vertical broad spatial search ranges are halved due to the smaller shifts that are typically encountered with these image sizes. For example, using CIF images in Figure 7, the horizontal axis would stretch from –6 to +6 pixels and the vertical axis would stretch from –8 to +8 frame lines.

The temporal search range, being stated in frames, is largely unchanged. For the broad temporal search in section 3.1.5.2, instead of matching one processed field one to every second original field one, the progressive algorithm compares one processed frame to every second original frame. For the colorbar algorithm, the search examines spatial shifts for one processed frame and one original frame (i.e., no temporal searching).

The only steps requiring more complicated changes are the fine searches from section 3.1.5.4 and section 3.1.6.2. Here, the vertical shifts remain unchanged, lying between -2 frame lines and +2 frame lines. Thus, the vertical axis of Figure 8 and Figure 11 is interpreted as referring to frame lines. The temporal extent of these fine searches may be set to five original frames centered on the current aligned original frame, instead of the three original frames otherwise implied. A five-frame search extent may improve the speed and efficiency of the fine search when compared to the interlace version of the algorithm, since progressive HRCs are more likely to contain varying video delay than non-zero spatial shifts.

When considering the algorithmic changes for progressive video systems, many of the spatial shift search parameters can be modified without harming the integrity of the algorithm. As an example, consider spatial shifts other than zero pixels and zero lines for the broad temporal search. The spatial shift at zero pixels horizontally and 8 field lines vertically for interlace video could be moved to 16 frame lines for progressive video, as recommended above, or placed at 8 frame lines, under the assumption that progressive video sequences are unlikely to contain 16 frame lines of vertical shift. Likewise, spatial

[5] In one possible exception to this doubling, the spatial shift associated with zero pixels horizontally and plus or minus one field line vertically could be left at plus or minus one frame line vertically. Spatial shifts very close to (zero, zero) are commonly encountered.

shift at zero lines vertically and 8 pixels horizontally could be moved to 9 or 10 pixels horizontally without any detrimental effects. As another example, the exact number of repeated fine searches performed could be increased or decreased for specific applications. The exact values recommended here are significantly less important than the actual structure of the search algorithm.

3.2 Valid Region

NTSC (525-line) and PAL (625-line) video sampled according to Rec. 601 may have a border of pixels and lines that do not contain a valid picture. The original video from the camera may only fill a portion of the Rec. 601 frame. A digital video system that utilizes compression may further reduce the area of the picture in order to save transmission bits. If the non-transmitted pixels and lines occur in the over-scan area of the television picture, the typical end-user should not notice the missing lines and pixels. If these non-transmitted pixels and lines exceed the over-scan area, the viewer may notice a black border around the picture, since the system will normally insert black into this non-transmitted picture area. Video systems (particularly those that perform low-pass filtering) may exhibit a ramping up from the black border to the picture area. These transitional effects most often occur at the left and right sides of the image but can also occur at the top or bottom. Occasionally, the processed video may also contain several lines of corrupted video at the top or bottom of the picture that may not be visible to the viewer (e.g., VHS tape recorders corrupt several lines at the bottom of the picture in the over-scan area). To prevent non-picture areas from influencing the VQM measurements, these areas should be excluded from the VQM measurement. The automated valid region algorithm presented here estimates the valid region of the original and processed video streams so that subsequent computations do not consider corrupted lines at the top and bottom of the Rec. 601 frame, black border pixels, or transitional effects where the black border meets the picture area.

3.2.1 Core Valid Region Algorithm

This section describes the core valid region algorithm that is applied to a single original or processed image. This algorithm requires three input arguments: an image, a maximum valid region, and the current valid region estimate.

- **Image**. The core algorithm uses the Rec. 601 luminance image of a single video frame. When measuring the valid region of a *processed* video sequence, any spatial shift imposed by the video system must have been removed from the luminance image before applying the core algorithm (see spatial registration section 3.1).
- **Maximum Valid Region**. The core algorithm will not consider pixels and lines outside of a maximum valid video region. This provides a mechanism for the user to specify a maximum valid region that is smaller than the entire image area if *a priori* knowledge indicates that the sampled image has corrupted pixels or lines as discussed in section 3.2.
- **Current Valid Region**. The current valid region is an estimate of the valid region and lies entirely within the maximum valid region. All pixels inside the current valid region are known to contain valid video; pixels outside the current valid region may or may not contain valid video content. Initially, the current valid region is set to the smallest possible area located at the exact center of the image.

The core algorithm examines the area of video between the maximum valid region and the current valid region. If some of those pixels appear to contain valid video, the current valid region estimate is enlarged. The algorithm will now be described in detail for the left edge of the image.

1. Compute the mean of the left-most column of pixels in the maximum valid region. The left-most column of pixels will be denoted as column "J-1" and the mean will be represented by "M_{J-1}".

2. Take the mean of the next column of pixels, “M_J”.
3. Column J is declared invalid video if it is black, ($M_J < 20$) or if the average pixel level of the mean value for successive columns indicates a ramp up from black border to valid picture ($M_J - 2 > M_{J-1}$). If either of these conditions are true, increment J and repeat steps (2) and (3). Otherwise, go to step (4).
4. If final column J is within the current valid region, then no new information has been obtained. Otherwise, update the current valid region with J as the left coordinate.

The algorithm for finding the top edge of the image is similar to that given above for the left edge. For the bottom and right edges, J is decremented instead of incremented; otherwise the algorithm is the same. The values produced for top, left, bottom, and right indicate the last valid pixel or line.

The stopping conditions identified in step (3) can be fooled by scene content. For example, an image that contains genuine black at the left side (i.e., black that is part of the scene) will cause the core algorithm to conclude that the left-most valid column of video is farther toward the middle of the image than it ought to be. For that reason, the core algorithm is applied to multiple images from a video sequence, thereby increasing the accuracy of the valid region estimate.

3.2.2 Applying the Core Valid Region Algorithm to a Video Sequence

3.2.2.1 Original Video

The core algorithm is first applied to the original sequence of images. For NTSC video sampled according to Rec. 601 (see section 2), the recommended setting for the maximum valid region is top = 6, left = 6, bottom = 482, right = 714. For PAL video sampled according to Rec. 601, the recommended setting for the maximum valid region is top = 6, left = 16, bottom = 570, right = 704. The core algorithm is run on the first image in the video sequence, and every frequencyth image thereafter. For example, if the specified frequency were 15, the core algorithm would examine sequence image numbers 0, 15, 30, 45, and so forth. When all images in the sequence have been examined, the current valid region will contain the largest valid area implied by any examined image in the video sequence. Pixels and lines between this final current valid region and the maximum valid region are considered to contain either black or a transitional ramp up from black.

The final valid region must contain an even number of lines and an even number of pixels. Any odd top or left coordinates are incremented by one. Then, if the region contains an odd number of lines, bottom is decremented; likewise, if the region contains an odd number of pixels (e.g., horizontally), right is decremented. This simplifies color processing for video sampled in accordance with Rec. 601, since the color channels are sub-sampled by 2 when compared to the luminance channel. Also, each interlaced field of video will contain the same number of video lines. This will assure that spatial-temporal sub-regions (from which features will be extracted) always contain valid video with equal contributions from both interlaced fields. The resulting valid region is returned as the original valid region.

3.2.2.2 Processed Video

When computing the valid region of the processed video sequence, the maximum valid region setting for the core algorithm is first set equal to the corresponding original valid region found for that scene. This maximum valid region is then reduced in size by any pixels and lines that are considered invalid due to spatially shift correcting the processed video frames. The core algorithm is then run on the first image in the processed video sequence, and every frequencyth image thereafter (i.e., if frequency = F, use images $\mathbf{Y}(0)$, $\mathbf{Y}(F)$, $\mathbf{Y}(2F)$, $\mathbf{Y}(3F)$, and so forth).

After the core algorithm has been applied to the processed video sequence, the valid region found by the core algorithm is reduced inward by a safety margin. The recommended safety margin discards one line off the top and bottom, and five pixels off the left and right. The large left and right safety margins ensure that any ramp up or down from black is excluded from the processed valid region.

The final processed valid region must contain an even number of lines and an even number of pixels. Any odd top or left coordinates are incremented by one. Then, if the region contains an odd number of lines, bottom is decremented; likewise, if the region contains an odd number of pixels (e.g., horizontally), right is decremented. The resulting valid region is returned as the processed valid region.

3.2.3 Comments on Valid Region Algorithm

This automated valid region algorithm will work well to estimate the valid region of most scenes. Due to the nearly infinite possibilities for scene content, the algorithm described herein takes a conservative approach to estimation of the valid region. A manual examination of valid region would quite likely choose a larger region. Conservative valid region estimates are more suitable for an automated video quality measurement system, because discarding a small amount of video will have little impact on the quality estimate and in any case this video usually occurs in the over-scan part of the video. On the other hand, including corrupted video in the video quality calculations may have a large impact on the quality estimate.

This algorithm does not contain sufficient artificial intelligence to distinguish between corrupted pixels and lines at the edge of an image and true scene content. A rule of thumb is used instead, stating that such invalid video generally occurs at the extreme edges of the image. Specification of a conservative user-definable maximum valid video region (i.e., the starting point for the automated algorithm) provides a mechanism to exclude these possibly corrupt image edges from consideration.

When the valid region algorithm is applied to video that is not sampled according to Rec. 601 (e.g., the common intermediate format, or CIF, used by ITU-T Recommendation H.261), the recommended setting for maximum valid region when examining the original video is the entire image. In these cases, the sampled video does not normally include any corrupted over-scan, so a maximum valid region smaller than the entire image is unnecessary.

3.3 Gain and Offset

3.3.1 Core Gain and Level Offset Algorithm

This section explains the method for performing gain and level offset calibration. A prerequisite before applying this algorithm is that the original and processed images be spatially registered (see section 3.1). The original and processed images must also be temporally registered, which will be addressed later in section 3.4. Gain and level offset calibration can be performed on either fields or frames as appropriate.

The method presented here makes the assumption that the Rec. 601 Y, C_B, and C_R signals each have an independent gain and level offset. This assumption will in general be sufficient for calibrating component video systems (e.g., Y, R-Y, B-Y). However, in composite or S-video systems, it is possible to have a phase rotation of the chrominance information since the two chrominance components are multiplexed into a complex signal vector with amplitude and phase. The algorithm presented here will not properly calibrate video systems that introduce a phase rotation of the chrominance information (e.g., the hue adjustment on a television set).

As previously noted, this calibration model assumes that there is no cross coupling between any of the three video components. With this assumption, the core calibration algorithm is applied independently to each of the three channels: Y, C_B, and C_R.

The valid region of the original and processed image plane is first divided into N sub-regions. For each of the sub-regions, the mean *original* and *processed* values are computed (i.e., mean over space). Next, these *original* and *processed* values are represented as N-element column vectors $\underline{O}$ and $\underline{P}$, respectively:

$$\underline{O}_{N \times 1} = \begin{bmatrix} original_1 \\ . \\ . \\ . \\ original_N \end{bmatrix}, \ \underline{P}_{N \times 1} = \begin{bmatrix} processed_1 \\ . \\ . \\ . \\ processed_N \end{bmatrix}.$$

Calibration involves computing the gain (g) and level offset (l) according to the following model:

$$\underline{P} = g\underline{O} + l \ .$$

Since there are only two unknowns (i.e., g and l) but N equations (i.e., N sub-regions), we must solve the over-determined system of linear equations given by:

$$\hat{\underline{P}} = A\begin{bmatrix} l \\ g \end{bmatrix},$$

where A is an N x 2 matrix given by $A_{N \times 2} = \begin{bmatrix} \underline{1} & \underline{O} \end{bmatrix}$, and $\underline{1}$ is an N-element column vector of '1s' given by

$$\underline{1}_{N \times 1} = \begin{bmatrix} 1_1 \\ . \\ . \\ . \\ 1_N \end{bmatrix}.$$

$\hat{P}$ is the estimate of the processed samples if the gain and level offset correction were applied to the original samples. The least squares solution to this over-determined problem (provided N > 2) is given by

$$\begin{bmatrix} l \\ g \end{bmatrix} = \left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}} P \, ,$$

where the superscript "T" denotes matrix transpose and the superscript "-1" denotes matrix inverse.

When the core gain and offset algorithm is independently applied to each of the three channels, six estimates result: Y gain, Y offset, C_B gain, C_B offset, C_R gain, and C_R offset.

3.3.2 Using Special SMPTE Color Bars

This section explains the method for performing gain and offset calibration using the EIA color bar portion of the special SMPTE color bars shown in Figure 9. Temporal registration is unnecessary since this is a static scene.

Table 1 specifies the [Y, C_B, C_R] values for the various sub-regions in the EIA color bar that are part of the SMPTE Rec. 601 color bar test pattern shown in Figure 9. Here, 128 has been subtracted from the C_B and C_R components to center them about zero. The [Y, C_B, C_R] intensity levels are uniform within the specified rectangles. For the sub-regions specified in Table 1, the image top left is at (0, 0) and the image bottom right is at (485,719) (see Figure 4). Using values that span only 75% of the amplitude scale provides plenty of headroom for systems with large gains and/or offsets.

Table 1. Full Sub-Region Y, CB, CR Values

Top	**Left**	**Bottom**	**Right**	**Y**	**C_B**	**C_R**
1	10	204	109	180	0	0
1	110	204	209	162	-84	14
1	210	204	309	131	28	-84
1	310	204	409	112	-56	-70
1	410	204	509	84	56	70
1	510	204	609	65	-28	84
1	610	204	709	35	84	-14

A buffer of 20 pixels and lines all the way around each of the sub-regions should be sufficient to exclude any transitional effects that low bit-rate codecs might introduce. The recommended sub-regions with the 20 pixel/line buffer are given in Table 2. The core gain and level offset algorithm given in section 3.3.1 is used together with these seven sub-regions to estimate the gain and level offset of the Y, C_B, and C_R video components.

Table 2. Buffered Sub-Region Y, CB, CR Values

Top	**Left**	**Bottom**	**Right**	**Y**	**C_B**	**C_R**
21	30	184	89	180	0	0
21	130	184	189	162	-84	14
21	230	184	289	131	28	-84
21	330	184	389	112	-56	-70
21	430	184	489	84	56	70
21	530	184	589	65	-28	84
21	630	184	689	35	84	-14

3.3.3 Using Scenes

The basic algorithm given in section 3.3.1 can be applied to original and processed video streams provided they have been spatially and temporally registered. Whereas the color bar technique presented in section 3.3.2 divides the image into seven blocks with evenly distributed intensity levels, the more general scene-based technique divides the image into abutting blocks with unknown intensity levels. A sub-region size of 16 lines x 16 pixels is recommended for frames (i.e., 8 lines x 16 pixels for one Y NTSC field; 8 lines x 8 pixels for C_B and C_R due to sub-sampling of the color image planes). The mean over space of the [Y, C_B, C_R] samples is computed for each corresponding original and processed sub-region, or block, to form a spatially sub-sampled image. All the selected blocks must lie within the processed valid region (PVR). Here, the number of sub-regions (N) will be much larger than in section 3.3.2.

3.3.3.1 Registering the Processed Images

For simplicity, we will assume that the best spatial registration has already been found using one of the techniques presented in section 3.1. Before gain and level offset are estimated, each processed image must also be temporally registered. The original image that best aligns with the processed image must be used for the gain and level offset calculation. If the video delay is variable, this temporal registration must be performed for each processed image. If the video delay is constant for the scene, the temporal registration only needs to be performed once.

To temporally register a processed image, first create the spatially sub-sampled original and processed fields (or frames for progressive video) as specified in section 3.3.3, after correcting for the spatial shift of the processed video. Using the sub-sampled Y images, apply the search function given in section 3.1.4.3, except perform this search using all the original images that are within the temporal registration uncertainty (**U**). Use the best resulting temporal registration for all three image planes, Y, C_B, and C_R.

3.3.3.2 Gain & Level Offset of Registered Images

An iterative least squares solution with a cost function is used to help minimize the weight of outliers in the fit. This is because outliers are normally due to distortions rather than pure level offset and gain changes, and assigning equal weight to these outliers would distort the fit.

The following algorithm is applied separately to the N matching original and processed pixels from each of the three spatially sub-sampled images [Y, C_B, C_R].

1. Use the normal least squares solution from section 3.3.1 to generate the initial estimate of the level offset and gain: $\begin{bmatrix} l \\ g \end{bmatrix} = \left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}} \underline{P}$.
2. Generate an error vector ($\underline{E}$) that is equal to the absolute value of the difference between the true processed samples and the fitted processed samples: $\underline{E} = \left|\underline{P} - \underline{\hat{P}}\right|$.
3. Generate a cost vector ($\underline{C}$) that is the element-by-element reciprocal of the error vector ($\underline{E}$) plus a small epsilon (ε): $\underline{C} = \dfrac{1}{\underline{E} + \varepsilon}$. The ε prevents division by zero and sets the relative weight of a point that is on the fitted line versus the weight of a point that is off the fitted line. An ε of 0.1 is recommended.

4. Normalize the cost vector $\underline{C}$ for unity norm (i.e., each element of $\underline{C}$ is divided by the square root of the sum of the squares of all the elements of $\underline{C}$).
5. Generate the cost vector $\underline{C}^{2}$ that is the element-by-element square of the cost vector $\underline{C}$ from step 4.
6. Generate an N x N diagonal cost matrix (C^{2}) that contains the cost vector's elements ($\underline{C}^{2}$) arranged on the diagonal, with zeros everywhere else.
7. Using the diagonal cost matrix (C^{2}) from step 6, perform cost-weighted least squares fitting to determine the next estimate of the level offset and gain: $\begin{bmatrix} l \\ g \end{bmatrix} = \left(A^{\mathrm{T}} C^{2} A\right)^{-1} A^{\mathrm{T}} C^{2} \underline{P}$.
8. Repeat steps 2 through 7 until the level offset and gain estimates converge to four decimal places.

These steps are applied separately to processed field one and processed field two, to obtain two estimates for *g* and *l*. Field one and two must be examined separately, because the temporally registered original fields need not correspond to one frame within the original video sequence. For progressive video, the above steps are applied to the entire processed frame at once.

3.3.3.3 Estimating Gain and Level Offset for a Video Sequence and HRC

The algorithm described above is applied to multiple matching original and processed field pairs distributed every frequencyth frame throughout the scene (for progressive video, original and processed frame pairs). A median filter is then applied to the six time histories of the level offsets and gains to produce average estimates for the scene.

If several scenes have been passed through the same HRC, the level offset and gain for each scene will be considered to be identical. Thus, filtering results obtained from multiple scenes can increase the robustness and accuracy of the level offset and gain measurements. The overall HRC level offset and gain results can then be used to compensate all of the processed video for that HRC.

3.3.4 Applying Gain and Level Offset Corrections

The temporal registration algorithms (see section 3.4) and most quality features (section 4) will specify that the gain calculated herein should be removed. To remove gain and level offset from the Y plane, apply the following formula to each processed pixel:

New Y(i,j,t) = [Y(i,j,t) – *l*] / *g*

Gain and level offset correction is not performed on the color planes (i.e., C_B and C_R). Perceptual chrominance errors are instead captured by the color metrics. The C_B and C_R image planes may be gain and level offset corrected for display purposes.

3.4 Temporal Registration

Modern digital video communication systems typically require several tenths of a second to process and transmit the video from the sending camera onto the receiving display. Excessive video delays impede effective two-way communication. Therefore, objective methods for measuring end-to-end video communications delay are important to end-users for specification and comparison of services and to equipment / service providers to optimize and maintain their product offerings. Video delay can depend upon dynamic attributes of the original scene (e.g., spatial detail, motion) and video system (e.g., bit-rate). For instance, scenes with large amounts of motion can suffer more video delay than scenes with small amounts of motion. Thus, video delay measurements should be made in-service to be truly representative and accurate. Estimates of video delay are required to temporally align the original and processed video features shown in Figure 1 before making quality measurements.

Some video transmission systems may provide time synchronization information (e.g., original and processed frames may be labeled with some kind of a frame numbering scheme). In general, however, time synchronization between the original and processed video streams must be measured. This section presents two techniques for estimating video delay based upon the original and processed video frames. The first technique uses low-bandwidth features (e.g., several features per frame) that are extracted from the original and processed video streams (section 3.4.1). This method is called "sequence-based" since it works by correlating a time history of features extracted from the processed video with the same time history of features extracted from the original video. The algorithm estimates one overall video delay, or temporal registration, for a selected sequence of video frames. The second technique is "frame-based" in that it works by correlating lower resolution images, sub-sampled in space and extracted from the original and processed video streams (section 3.4.2). This frame-based technique estimates the delay of each frame or field (for interlaced video systems). These individual estimates are combined to estimate the average delay of the video sequence.

3.4.1 Sequence-Based Algorithm for Estimating Constant Temporal Delays between Original and Processed Video Sequences

The temporal registration algorithm presented in this section is a reduced-reference method [15] that requires a very low bandwidth ancillary data channel with a data rate of 1 to 5 kb/s (see Figure 1). The low bandwidth ancillary data channel is used to transmit features for temporal registration that have been extracted from the original and processed video streams. The low bandwidth requirement, together with the computational simplicity of extracting the features, makes this method ideally suited for in-service real-time estimates of video delay. Thus, this measurement method is well-suited for fully automated video quality monitoring systems.

The video delay measurement method builds upon the ideas presented in section 6.4.1 of ANSI T1.801.03-1996 [3], which describes a system for computing a "constant alignment" offset, or delay, between a processed and corresponding original video feature stream.[6] This measurement method is superior to the one presented in ANSI T1.801.03-1996 for systems that perform field and/or frame repetition.

A set of low bandwidth features track changes in scene motion and image intensity. Specifically, four features are extracted that quantify changes in scene motion and image intensity. Video delay is estimated by cross-correlating, or aligning, the information in the processed feature streams with the corresponding information in the original feature streams. A prerequisite for applying this temporal registration algorithm is that the original and processed images be spatially registered (see section 3.1) and gain and level offset calibrated (see section 3.3). In addition, temporal registration algorithms should only be applied to the processed valid region (PVR) of the original and processed video streams (see section 3.2).

[6] The term "constant alignment" as used in ANSI T1.801.03 refers to a temporal registration process that estimates one temporal offset, or video delay, for all the output video frames. This is contrasted with "variable alignment" techniques mentioned in section 6.4.2 of ANSI T1.801.03 and detailed in ITU-T Recommendation P.931 [17] where each output video frame can have a unique video delay. The terms "alignment" and "registration" are used interchangeably in this document.

Examples and descriptions throughout the following sections presume that this algorithm is applied to interlaced NTSC video streams. This algorithm may be applied to other video streams with minor modifications.[7]

3.4.1.1 Description of Features

The constant alignment algorithm uses four low-bandwidth features in succession, trying to achieve temporal registration at each stage before proceeding to the next feature. The first feature is a field-based implementation of the frame-based temporal information (TI) feature used in section 6.4.1 of ANSI T1.801.03 [3] and ITU-T Recommendation P.910 [16]. This TI feature uses temporal field differences spaced 2 fields apart in time (i.e., 1 frame). Two other TI features are used that are based on temporal field differences spaced 4 fields apart (2 frames) and 10 fields apart (5 frames). The TI features quantify the amount of scene motion by summing the squared pixel differences between two sampled images. If the frame rate in units of frames per second (FPS) decreases and the digital video system fills in the non-transmitted frames with repetitions of prior frames, processed scene motion is perceived as unnatural or discontinuous. To extract motion features that smooth over these times of no motion in the processed video (i.e., repeated frames or fields), it is necessary to base motion calculations on images that are spaced far enough apart in time (e.g., 2, 4, 10 fields). By varying the temporal spacing for the TI calculation, a wide range of FPS rates can be accommodated (e.g., 30, 15, 6 FPS). Another feature is used that is based on the mean of the field. This feature quantifies the intensity, or brightness, of the video scene by measuring the average luminance level.

Each of the four features characterizes the video differently. Thus, each works best for different amounts of frame repeating, varying video delay, and scene intensity fluctuations; and one feature may succeed where another one fails. Features are examined sequentially by the algorithm, where later features measure increasingly coarser temporal variations (to accommodate lower frame rate systems). The standard deviation of the feature time history, and the magnitude of the correlation function for the feature's best temporal offset are used as indicators for the reliability of that feature's temporal registration. The first reliable feature that is found is used to estimate the constant video delay.

In the following feature definitions, the luminance field is noted as Y, and the time when this field occurs is denoted as t. Pixels of Y are further subscripted by row and column, i and j, respectively, so that an individual pixel is denoted as Y(i,j,t). This notation agrees with Figure 3.

<u>TI2 Feature: Two Field Difference Temporal Information</u>

The first feature used will be denoted as TI2 and is calculated as shown in Figure 12. The TI2 feature is essentially the same feature as that used by the ANSI T1.801.03 constant alignment algorithm, but computed on fields instead of frames. To compute TI2 at time t, consider field $\mathbf{Y}(t)$ and the previous field of the same type $\mathbf{Y}(t\text{-}2)$, and compute

$$\mathrm{TI2}(i,j,t) = \mathrm{Y}(i,j,t) - \mathrm{Y}(i,j,t-2))$$

for each pixel, and then compute

$$\mathrm{TI2}(t) = rms_{\mathrm{space}}\left[\mathrm{TI2}(i,j,t)\right] \equiv \sqrt{\frac{1}{\mathrm{R}}\sum_i \sum_j \mathrm{TI2}(i,j,t)^2} \;\;,$$

[7] The sequence-based temporal registration algorithm for progressive video will not be described in this document. The sequence-based algorithm may be used with progressive video by simply operating on frames instead of fields. For example, TI10 (see section 3.4.1.1) becomes a five frame wide temporal difference.

where rms_{space} is the root mean square function over space defined by the above equation, *i* and *j* are within the processed valid region (PVR), and R is the total number of pixels in the image sub-region (i.e., the number of pixels in the double summation).

Next, compute the standard deviation of the time history (std_{time}) of TI2(*t*) as

$$std_{\text{time}}[\mathrm{TI2}(t)] \equiv \sqrt{\frac{1}{\mathrm{M}}\sum_t \mathrm{TI2}(t)^2 - \left[\frac{1}{\mathrm{M}}\sum_t \mathrm{TI2}(t)\right]^2} \;,$$

where M is the total number of time samples in the TI2(*t*) waveform. We have found that if

$$std_{\text{time}}[\mathrm{TI2}(t)] \le \mathrm{TI\_THRESHOLD} \;,$$

where TI_THRESHOLD = 0.05, then temporal registration estimates based on this feature become unreliable. In this case, the TI2 waveform detects insufficient time changes in the amount of scene motion to be useful for temporal registration (e.g., still scene, scene with a constant amount of motion).

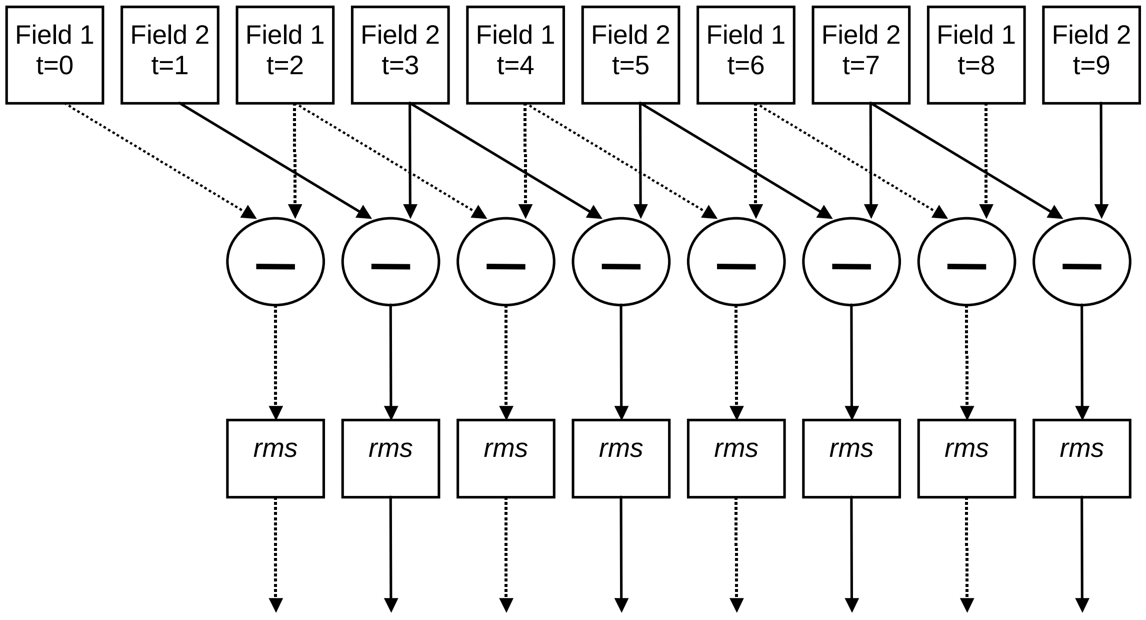

Figure 12. Diagram depicting the method of calculating TI2(***t***).

TI4 Feature: Four Field Difference Temporal Information

The second feature used will be denoted as TI4 and is calculated as shown in Figure 13. The TI4 feature is based on a temporal difference spaced four fields apart (i.e., two frames apart). This feature smoothes the temporal information using a wider filter than TI2, and eliminates frame repeats in the TI waveform for systems that have one or fewer consecutive frame repeats. To compute TI4 at time *t*, consider field **Y**(*t*) and the field of the same type two frames ago **Y**(*t*-4), and compute

$\mathrm{TI4}(i,j,t) = \mathrm{Y}(i,j,t) - \mathrm{Y}(i,j,t-4)$

for each pixel, and then compute

$\mathrm{TI4}(t) = rms_{\text{space}}[\mathrm{TI4}(i,j,t)]$.

As was the case with TI2(t), if the standard deviation of the time history of TI4(t) is 0.05 or less, then temporal registration estimates based on this feature become unreliable.

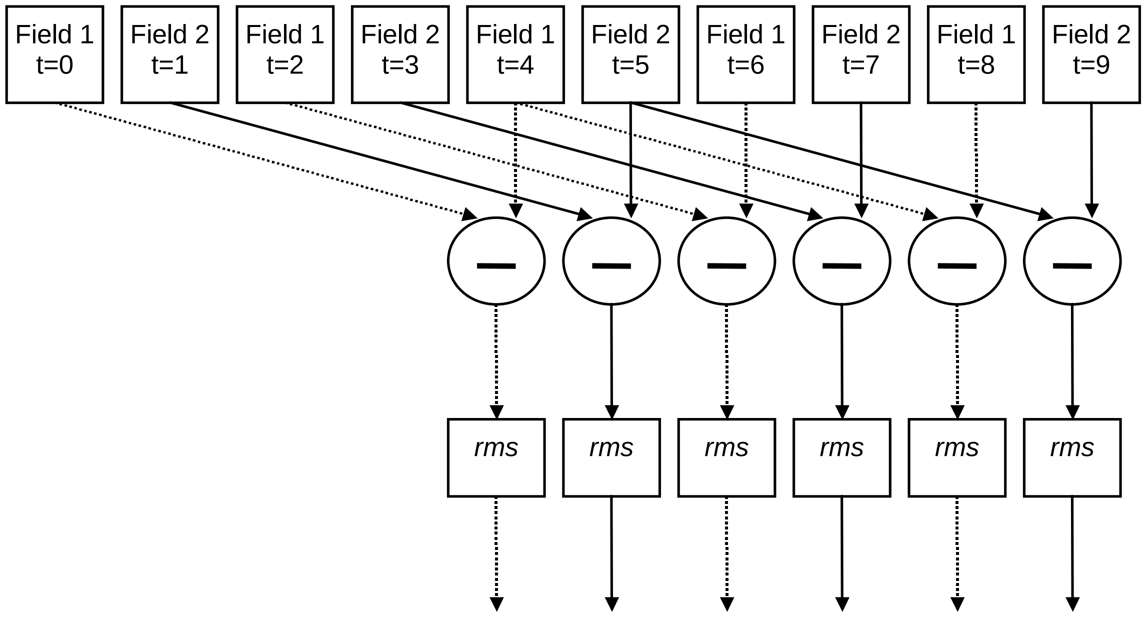

Figure 13. Diagram depicting the method of calculating TI4(***t***).

<u>Ymean Feature: Average Luminance Level</u>

The third feature used will be denoted as Ymean and is calculated as the average luminance level of a field. To compute Ymean at time t, consider field Y(t) and compute

$$\mathrm{Ymean}(t) = mean_{\mathrm{space}}\left[\mathrm{Y}(i, j, t)\right] \equiv \frac{1}{\mathrm{R}} \sum_{i} \sum_{j} Y(i, j, t)$$

where $mean_{\mathrm{space}}$ is the mean function over space defined by the above equation, i and j are within the processed valid region (PVR), and R is the total number of pixels in the image sub-region (i.e., the number of pixels in the double summation). We have found that if

$$std_{\mathrm{time}}\left[\mathrm{Ymean}(t)\right] \leq \mathrm{Y\_THRESHOLD}\ ,$$

where Y_THRESHOLD = 0.5, then temporal registration estimates based on this feature become unreliable. In this case, the Ymean waveform detects insufficient time changes in the amount of scene brightness to be useful for temporal registration (e.g., still scene, scene with a constant brightness level).

<u>TI10 Feature: Ten Field Difference Temporal Information</u>

The fourth feature used will be denoted as TI10. The TI10 feature is based on a temporal difference spaced ten fields apart (i.e., five frames apart). This feature smoothes the temporal information using a wider filter than TI4 and eliminates frame repeats in the TI waveform for systems that have four or fewer consecutive frame repeats. To compute TI10 at time t, consider field Y(t) and the field of the same type five frames ago Y(t-10), and compute

TI10(i,j,t) = Y(i,j,t) - Y(i,j,t-10) for each pixel, and then compute

TI10(t) = rms_{space}[TI10(i,j,t)].

As was the case with TI2(t) and TI4(t), if the standard deviation of the time history of TI10(t) is 0.05 or less, then temporal registration estimates based on this feature become unreliable.

3.4.1.2 Feature Sequence Correlation[8]

Figure 14 and the following steps describe the process that is used to estimate the proper time alignment between two corresponding feature streams extracted from the original and the processed video (section 3.4.1.1). The original feature stream will be referred to as a_o and the processed feature stream as a_p. For example, when using the two field difference temporal information, a(t) = TI2(t). Let M be the length of the feature streams, a_o and a_p.[9] The feature stream a_o, given by {$a_o(0)$, $a_o(1)$, $a_o(2)$, …, $a_o(M-1)$} will be abbreviated as $\{a_o(t)\}_{t=0}^{M-1}$. The temporal registration uncertainty, U, will be specified in *fields*.

1. Given a sequence of processed video features, $\{a_p(t)\}_{t=U}^{M-1-U}$; normalize (divide) each element in the sequence by the standard deviation of that sequence to form the normalized sequence $\{n_p(t)\}_{t=0}^{M-1-2*U}$. For convenience in describing the algorithm, the normalized sequence indexing starts at zero.
2. For each alignment delay guess, d, compute a sequence of original video features $\{a_o(t)\}_{t=U+d}^{M-1-U+d}$; normalize (divide) each element in the sequence by the standard deviation of that sequence to form the normalized sequences $\{n_o(d,t)\}_{t=0}^{M-1-2*U}$ for all $-U \le d \le U$. This original video normalization is essentially the same as the processed video normalization in step one, except computed for each delay guess, d.
3. Take the resulting normalized processed and original sequences and compute the difference between those sequences: $\text{Diff}(d,t) = \{n_o(d,t) - n_p(t)\}_{t=0}^{M-1-2*U}$.
4. Compute the standard deviation over time of the difference sequence for each delay offset d, namely S(d) = std_{time}(Diff(d,t)).
5. The minimum S(d) (denoted S_{min}) and its offset d_{min} is the best alignment indicated by this feature. Figure 15 gives an example plot of the correlation function S(d). In this plot, the best alignment occurs for the delay d = 0 fields (i.e., d_{min} = 0 fields).
6. If $S_{min} \le$ CORRELATION_THRESHOLD, where CORRELATION_THRESHOLD = 0.8, then the correlation between the selected processed sequence and the original video stream is probably reliable. Otherwise, the results of this correlation are suspect (the processed feature waveform is too dissimilar from the original feature waveform for a reliable estimate of constant temporal delay). The threshold of 0.8 was arrived at empirically (see section 3.4.1.4).

[8] The correlation function described in this section is based on minimization of the standard deviation of the difference between the original and processed feature streams, not the maximization of the energy in the cross-product of the two feature streams.

[9] M is not necessarily equal to N, the length of the Big YUV file, from Figure 2. Recall that t is a discrete variable that is used for indexing the frames in the Big YUV file.

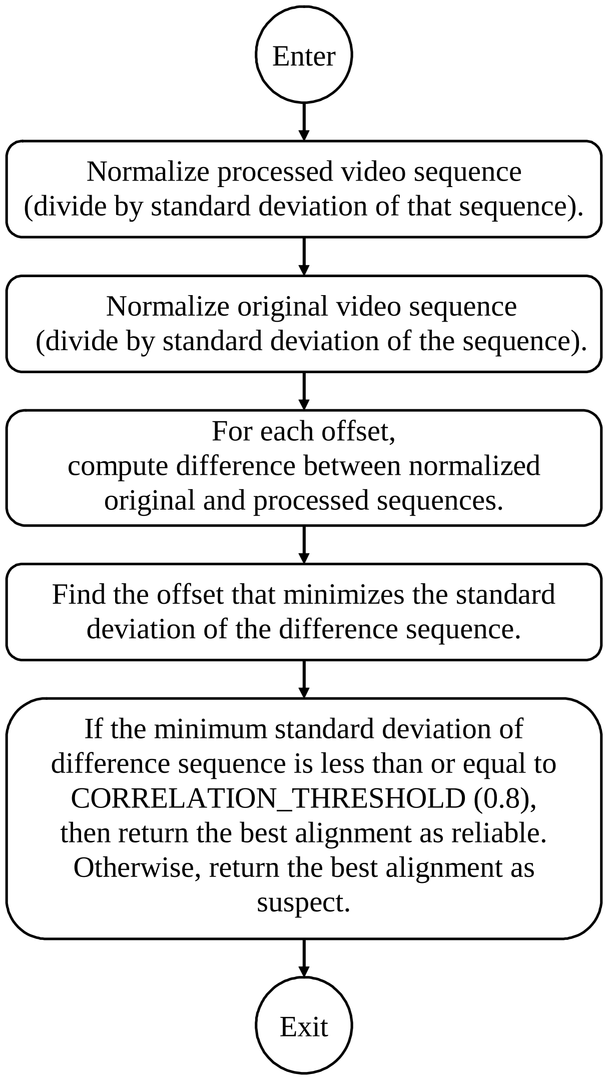
Enter
Normalize processed video sequence (divide by standard deviation of that sequence).
Normalize original video sequence (divide by standard deviation of the sequence).
For each offset, compute difference between normalized original and processed sequences.
Find the offset that minimizes the standard deviation of the difference sequence.
If the minimum standard deviation of difference sequence is less than or equal to CORRELATION_THRESHOLD (0.8), then return the best alignment as reliable. Otherwise, return the best alignment as suspect.
Exit

Figure 14. Correlation algorithm description.

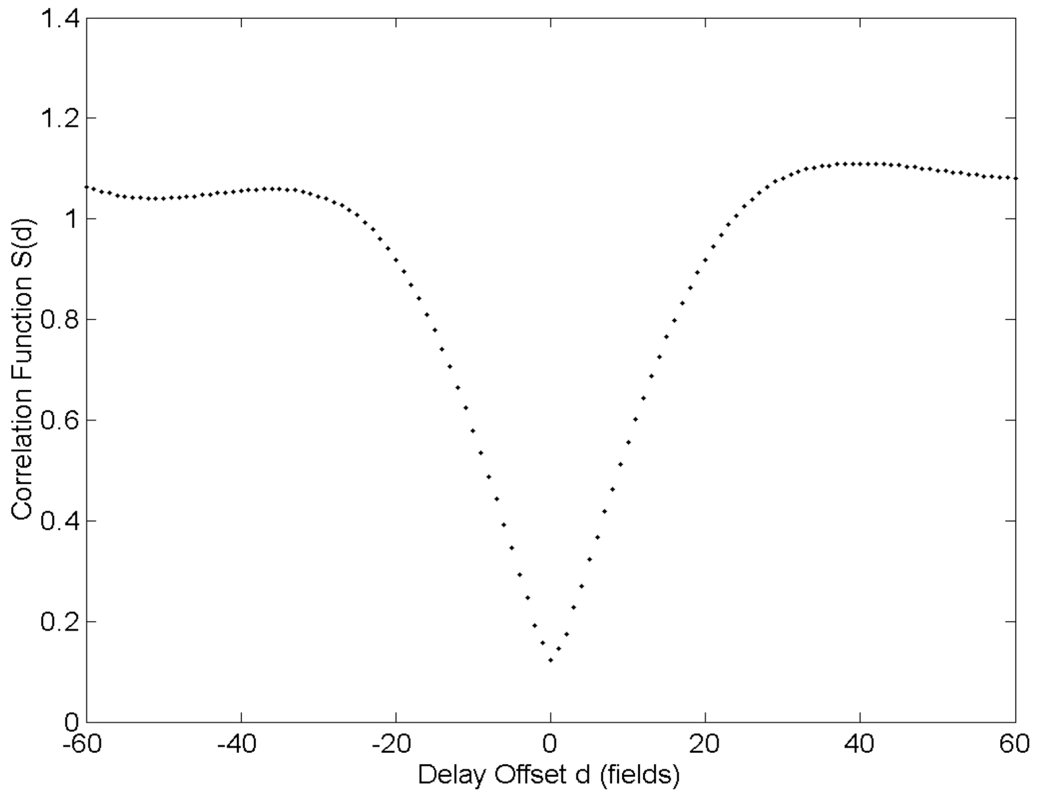

Figure 15. Example plot of correlation function S(d).

3.4.1.3 Description of Entire Algorithm

The following describes how to apply the four features of section 3.4.1.1 and the correlation algorithm of section 3.4.1.2 to achieve the final estimate of the alignment offset S_{min}.

1. Compute the std_{time} of the original and processed TI2 feature streams (section 3.4.1.1). If both computations exceed the TI_THRESHOLD, apply the correlation algorithm from section 3.4.1.2 to the TI2 feature streams. If the results are reliable, accept that alignment offset. Otherwise, proceed to step 2.

2. Compute the std_{time} of the original and processed TI4 feature streams (section 3.4.1.1). If both computations exceed the TI_THRESHOLD, apply the correlation algorithm from section 3.4.1.2 to the TI4 feature streams. If the results are reliable, accept that alignment offset. Otherwise, proceed to step 3.

3. Compute the std_{time} of the original and processed Ymean feature streams (section 3.4.1.1). If both computations exceed the Y_THRESHOLD, apply the correlation algorithm from section 3.4.1.2 to the Ymean feature streams. If the results are reliable, accept that alignment offset. Otherwise, proceed to step 4.

4. Compute the std_{time} of the original and processed TI10 feature streams (section 3.4.1.1). If both computations exceed the TI_THRESHOLD, apply the correlation algorithm from section 3.4.1.2

to the TI10 feature streams. If the results are reliable, accept that alignment offset. Otherwise, proceed to step 5.

5. If the results from step 4 are unreliable, then report that the video sequences cannot be aligned.

3.4.1.4 Threshold Optimization

The behavior of the constant temporal delay algorithm of section 3.4.1.3 is influenced by the setting of three empirical thresholds: TI_THRESHOLD, Y_THRESHOLD, and CORRELATION_THRESHOLD. TI_THRESHOLD sets the amount of variability, or information, which must be present in the TI feature streams before alignment is attempted. When the standard deviation over time of the TI2, TI4, and TI10 feature streams falls below TI_THRESHOLD (recommended value of 0.05), then we have found empirically that the TI feature streams do not contain enough information for alignment to be reliably calculated. This could be due to lack of scene motion (i.e., a still scene), or perhaps a constant amount of scene motion (e.g., the amount of motion present for any given field in the sequence is non-zero but constant). Y_THRESHOLD sets the amount of variability, or information, which must be present in the Ymean feature stream before alignment is attempted. When the standard deviation over time of the Ymean feature stream falls below Y_THRESHOLD (recommended value of 0.5), then the Ymean feature stream does not contain enough information for alignment to be reliably calculated. This could be due to lack of scene motion or perhaps a constant brightness level (e.g., the average brightness level for any given field in the sequence is constant).

If the TI2 and TI4 features fail because of constant scene motion or low frame rate, the Ymean feature can succeed if the scene's brightness level changes. If TI2, TI4, and Ymean all fail, then a final attempt is made to perform alignment using the TI10 feature, which further amplifies the detected scene motion and covers over the periods of very low frame rate. Raising the TI_THRESHOLD will cause the Ymean feature to be used more often. In turn, raising the Y_THRESHOLD will cause the TI10 feature to be used more often. If the TI and Y thresholds are raised too high, there is an increased likelihood that the algorithm will reach step 5 and report that the video sequences cannot be aligned.

The feature sequence correlation function given in section 3.4.1.2 uses CORRELATION_THRESHOLD with a recommended value of 0.8. If the original and processed feature sequences were identical, then their variances would cancel at correct alignment (i.e., S_{min} would be 0.0). On the other hand, if the original and processed waveforms were statistically independent, then their variances would add (i.e., S_{min} would be equal to the square root of 2.0, since the original and processed waveforms are each normalized to have a variance of 1.0). A CORRELATION_THRESHOLD of 0.8 imposes the constraint that at least 36% of the normalized processed variance must be canceled by the original (i.e., $1.0 - [0.8]^2$) in order to declare a good alignment. Raising the correlation threshold will increase the likelihood of falsely concluding that an alignment step has succeeded, when in fact it has failed. Lowering the correlation threshold will favor later steps in the constant temporal delay algorithm and increase the likelihood of reaching step 5 (i.e., the video sequences cannot be aligned). Table 3 summarizes the recommended values for the three thresholds used by the alignment algorithm.

Table 3. Recommended Values for Thresholds Used by Alignment Algorithm

Threshold	**Recommended Value**
TI_THRESHOLD	0.05
Y_THRESHOLD	0.5
CORRELATION_THRESHOLD	0.8

3.4.1.5 Example Results

This constant temporal delay algorithm was tested on selected video scenes and HRCs from the T1A1 video test data set ([1], [4]). The constant alignment algorithm's results all fell within the range of reasonable alignments that would be chosen visually and usually fell near the middle of that range. The example plots that follow are taken from two scenes (5row1, vtc1nw) and four HRCs (#2 - VHS, #11 - QCIF 128 kb/s, #18 - CIF 168 kb/s, #24 - CIF 1536 kb/s). The scene 5row1 depicts five people sitting in a row, conversing at a wooden table while the scene vtc1nw is a close-up shot of a woman's head and shoulders in front of a gray background [1].

Figure 16 shows a plot of the original and processed feature TI2 for the scene 5row1 and HRC 2. This clip was successfully aligned on the first step of the algorithm. The correlation function shown in Figure 15 was computed using the original and processed TI2 feature streams shown in Figure 16.

Figure 17 and Figure 18 depict the TI2 and the Ymean features, respectively, for the scene 5row1 and HRC 11. This clip has an average frame rate of about 10 frames per second. The feature TI2 depicted in Figure 17 was unable to reliably align the clip. The TI4 feature (not depicted) was also unable to align this particular clip. However, the clip was successfully aligned on step 3 of the algorithm using the Ymean feature depicted in Figure 18.

Figure 19 and Figure 20 depict the TI2 and TI4 features, respectively, for the scene vtc1nw and HRC 24. This clip has a frame rate of 15 frames per second. The TI2 feature depicted in Figure 19 is unable to reliably align this clip. However, the clip was successfully aligned on step 2 of the algorithm using the TI4 feature depicted in Figure 20.

The final example depicts the alignment of scene vtc1nw and HRC 18. The feature TI2 is plotted in Figure 21. Note that the frame repeat rate of this clip decreases during periods of high motion, such as toward the end of the clip. The feature TI2 probably fails to reliably align the clip due to the areas of low frame rate during the high motion periods. The feature TI4 (see Figure 22) has a similar problem. Because the average luminance level of the clip is fairly constant over time, the Ymean feature (see Figure 23) does not contain reliable alignment information either. However, the clip was successfully aligned on step 4 of the algorithm using the TI10 feature, which smoothes over the periods of very low frame rate (see Figure 24).

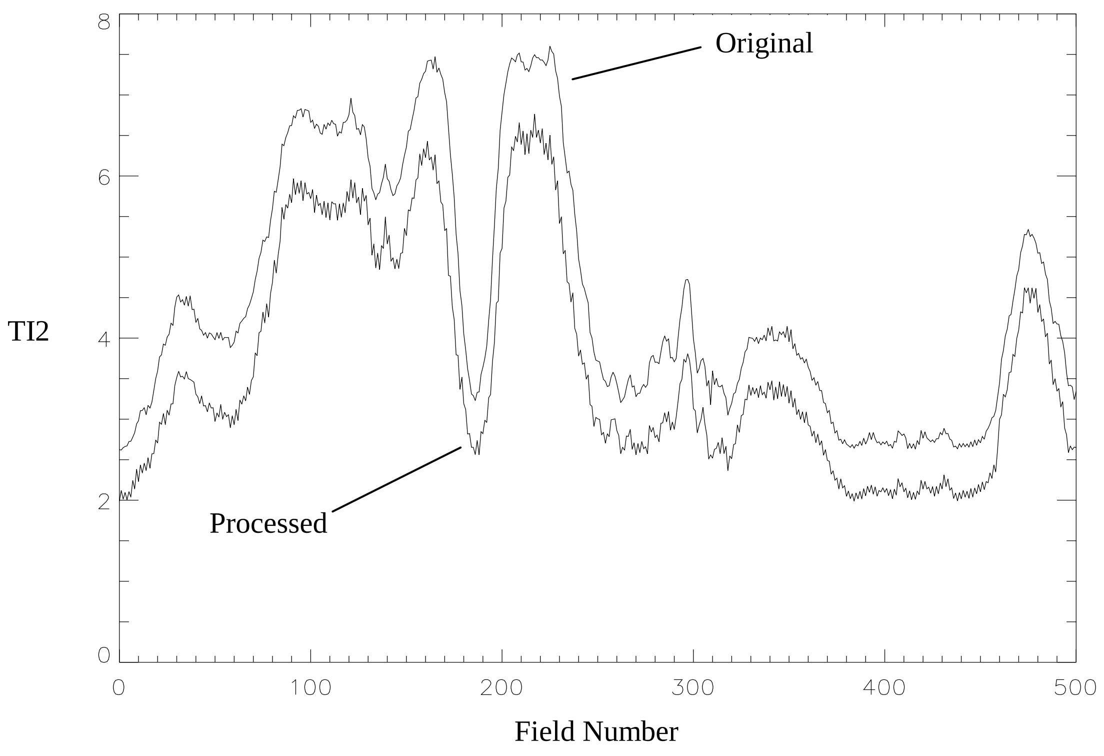

Figure 16. Aligned original and processed feature TI2 for the scene 5row1 and HRC 2.

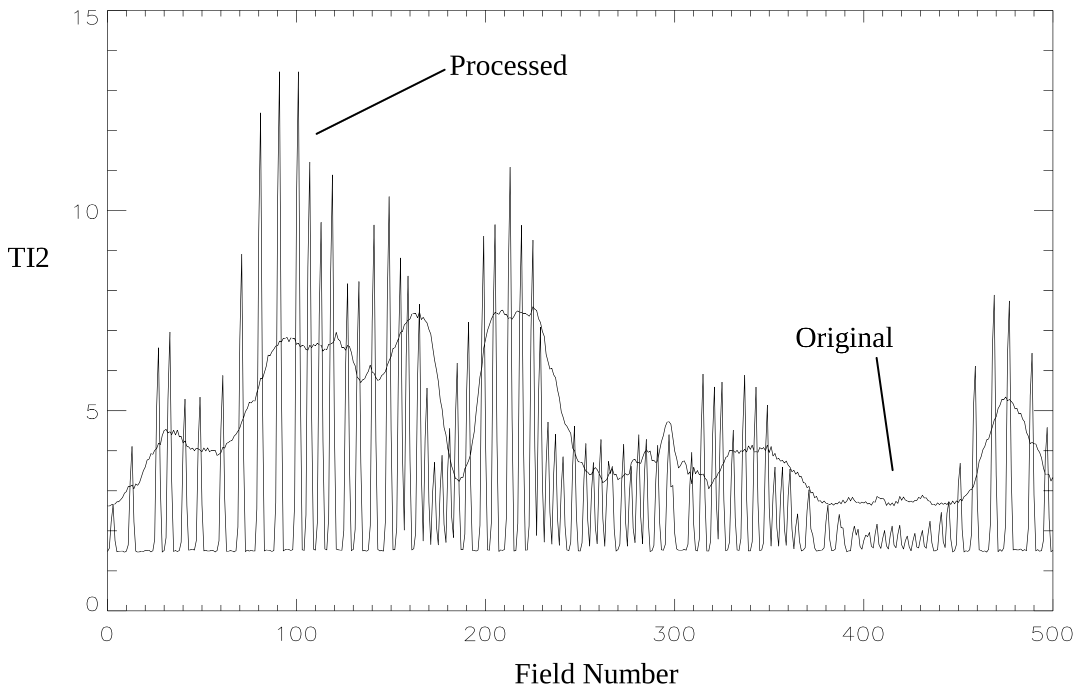

Figure 17. Original and processed feature TI2 for the scene 5row1 and HRC 11, failed alignment.

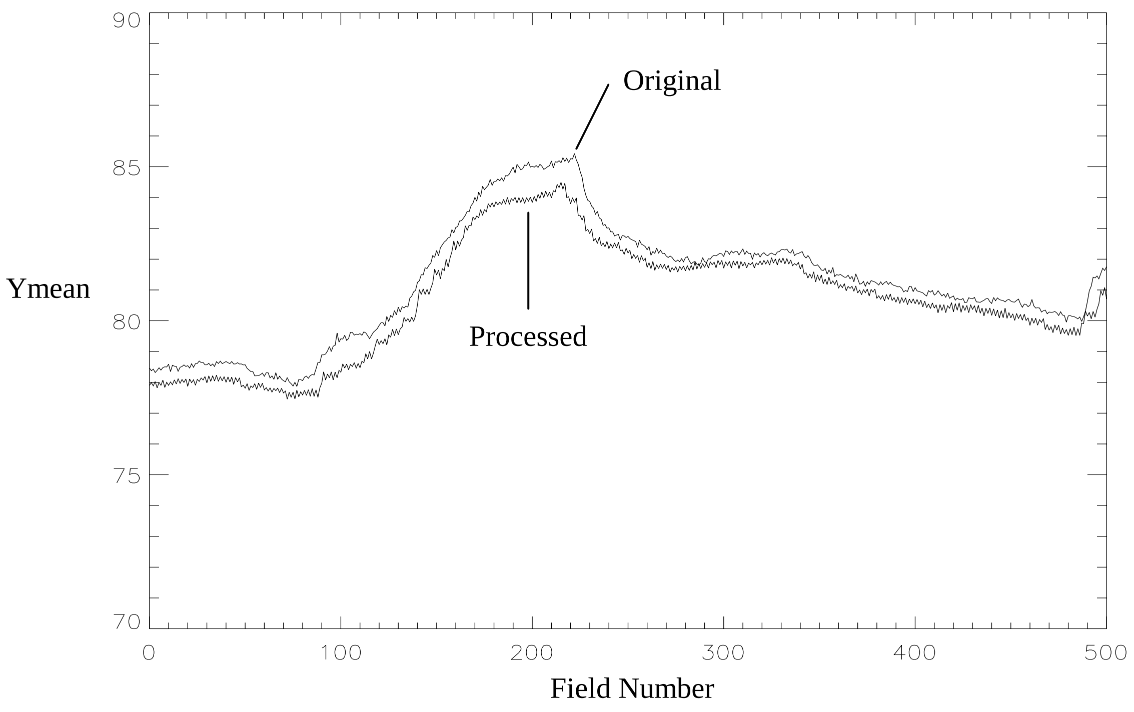

Figure 18. Aligned original and processed feature Ymean for the scene 5row1 and HRC 11.

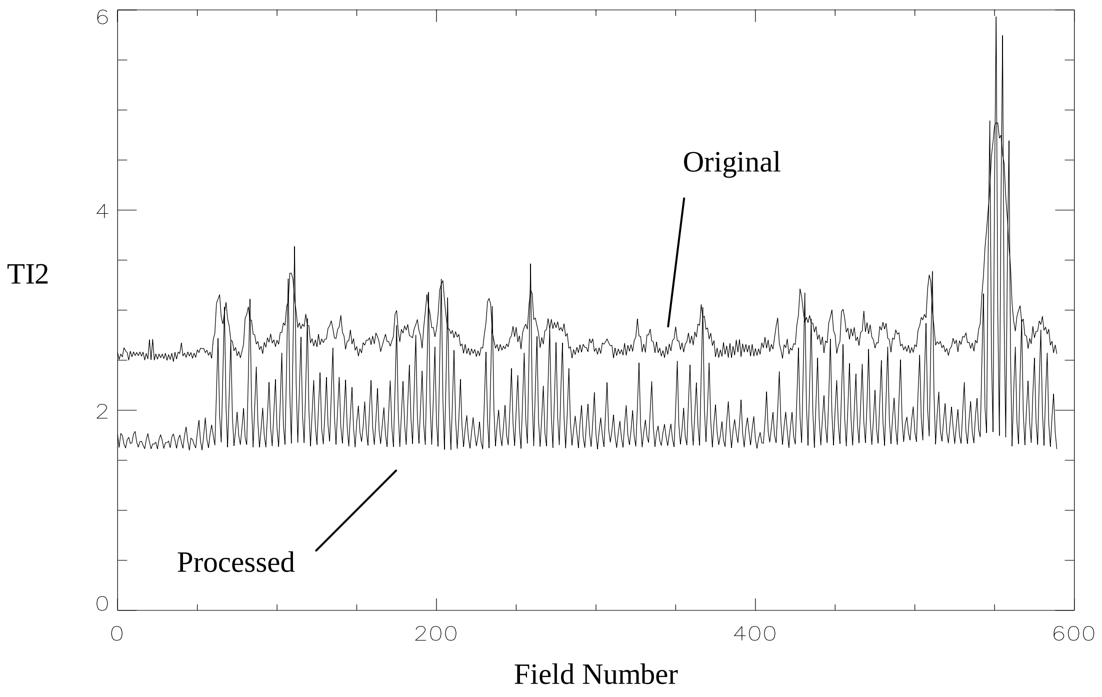

Figure 19. Original and processed feature TI2 for the scene vtc1nw and HRC 24, failed alignment.

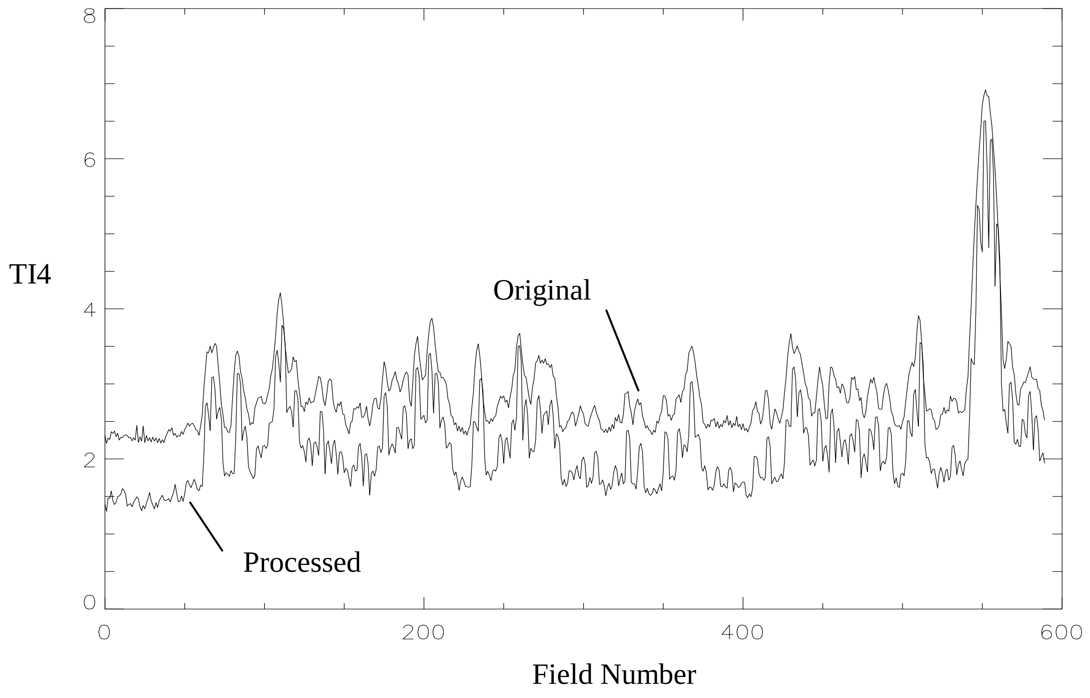

Figure 20. Aligned original and processed feature TI4 for the scene vtc1nw and HRC 24.

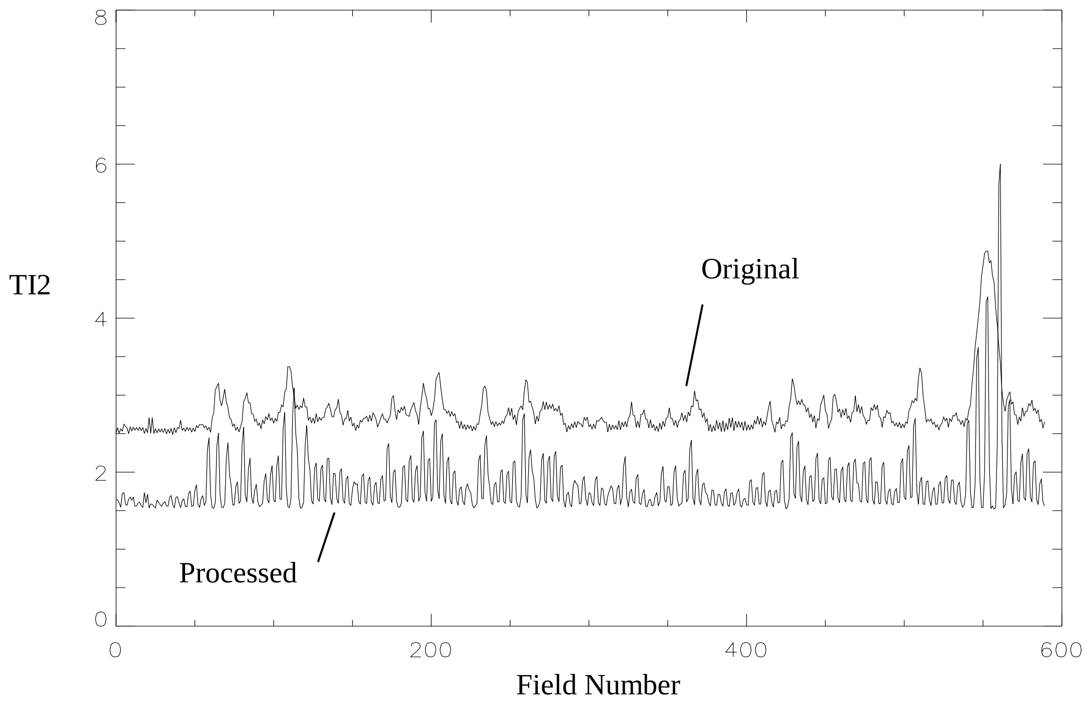

Figure 21. Original and processed feature TI2 for the scene vtc1nw and HRC 18, failed alignment.

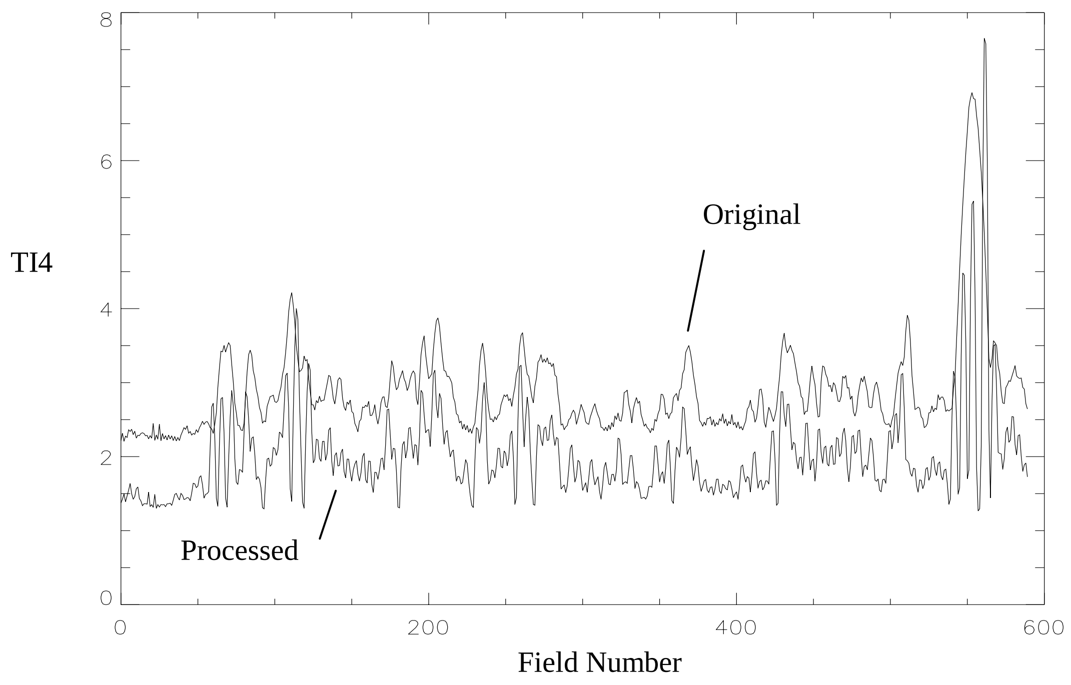

Figure 22. Original and processed feature TI4 for the scene vtc1nw and HRC 18, failed alignment.

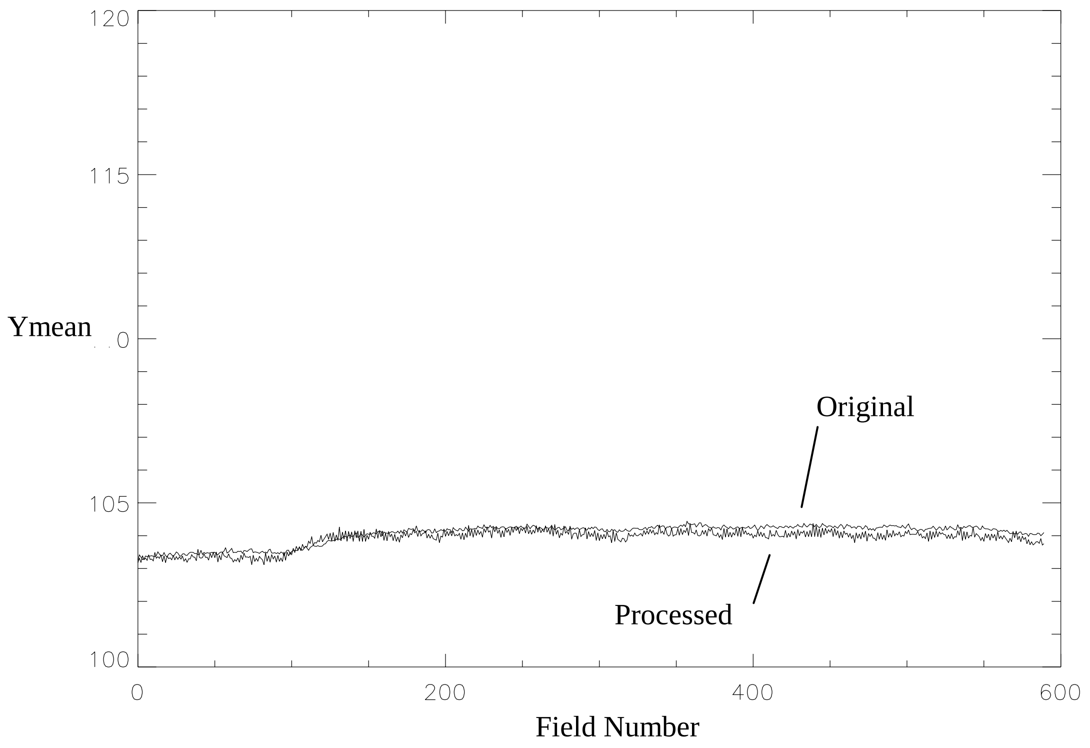

Figure 23. Original and processed feature Ymean for the scene vtc1nw and HRC 18, failed alignment.

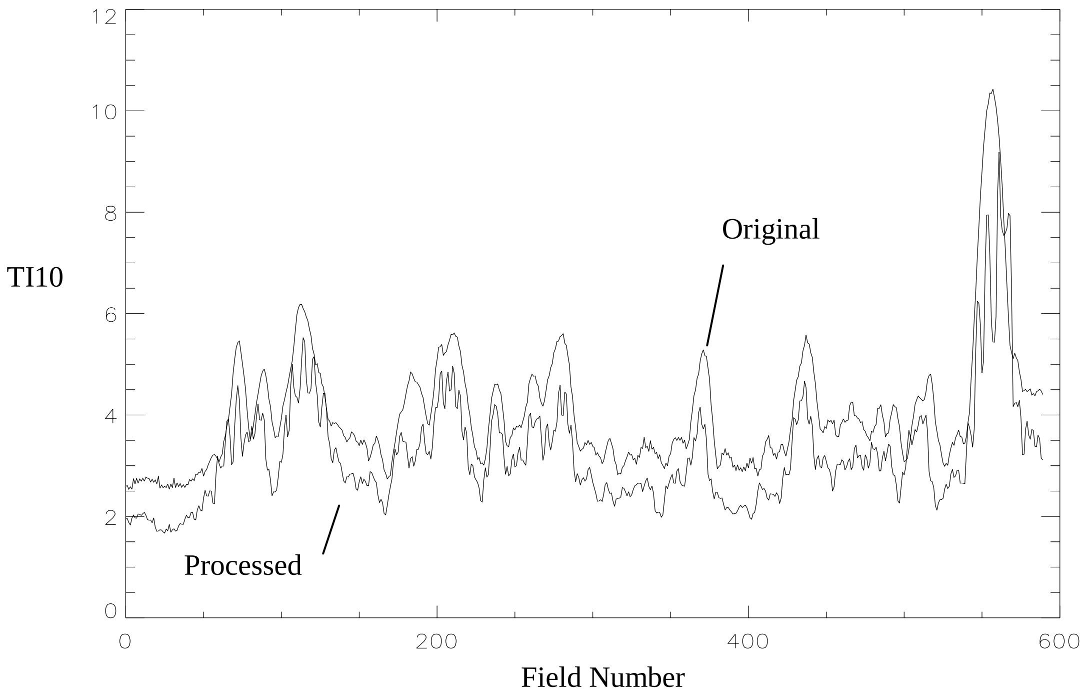

Figure 24. Aligned original and processed feature TI10 for the scene vtc1nw and HRC 18.

3.4.1.6 Observations and Conclusions

The in-service video delay measurement algorithm presented uses a set of very low bandwidth features extracted from processed original and processed video sequences. This algorithm is suitable for measuring video delay in a fully automated, in-service, real time monitoring system or for aligning video in an out-of-service environment, prior to performing video quality measurements. The video delay measurement algorithm utilizes two types of features, those that track changes in scene motion (TI2, TI4, TI10), and those that track changes in scene intensity (Ymean). The video delay measurement algorithm provides good estimates of video delay for a wide range of video scenes and systems by correlating, or time aligning, a sequence of these sampled feature values.

When field or frame repeats are present in the processed video sequence, the definition of video delay is somewhat ambiguous. Does a viewer perceive the delay of the first processed field in the field repeat sequence, the last processed field in the field repeat sequence, or something in between? Further subjective testing will be required to determine this relationship. The algorithm steps from TI2 (which is extremely accurate) to TI4, then to Ymean, and finally to TI10, where each feature is increasingly coarser than the one before. Because the four features operate at different levels of temporal granularity, the video delays measured by these features may differ slightly.

The length of the time window used to estimate delay limits the response of the measurement system to changes in video delay. Shorter time windows can cause measurement errors when the video contains small but perceptible amounts of repetitive motion limited to a small portion of the image; a particularly difficult scene is four seconds of a person wherein the only change in scene content is due to lip movement and camera noise. Shorter time windows can be used when the video contains variations in image intensity and / or scene motion. As the length of the time window increases, the measurement system will respond more slowly to variations in video delay.

As an interesting note, the alignments preferred by the TI features and the Ymean feature may differ slightly when variable video delay is present. The TI features tend to match low motion portions of the original and processed features, so that relatively still portions of the scene will appear best aligned. On the other hand, Ymean tends to match high motion portions of the original and processed video, since these portions of the video sequence tend to contain the greatest luminance level variance.

An alternative strategy would have been to compute TI with an increasing width in frames, ten frames, twenty frames, thirty frames, etc., rather than switching over to the luminance level feature Ymean. Although that technique would probably work reasonably well, it would require the calculation of many more features since the largest TI width would have to be greater than the largest number of times a frame could be repeated. In the T1A1 data set, this time extent is greater than two seconds. Also, whenever the TI width is larger than the smallest TI width necessary to align a clip, then useful temporal alignment information may be lost, and this may decrease the accuracy of the constant alignment. Replacing Ymean with a set of increasing width TI features would also increase computation time and algorithmic complexity. Hence, the Ymean feature is a more economical approach.

The fourth feature, TI10, is used when TI2, TI4, and Ymean fail: clips with small changes in average luminance level (possibly due to extremely small amounts of motion) and low processed frame rates. In these cases, use of a wider TI filter (e.g., TI10) amplifies and smoothes the limited scene motion to the point where alignment may become feasible. In the T1A1 data set, several video teleconferencing scenes fell into this category. All four features will fail when applied to truly still video. For out-of-service systems, however, note that the temporal registration of still video is irrelevant, because all original images are identical.

3.4.2 Frame-Based Algorithm for Estimating Variable Temporal Delays between Original and Processed Video Sequences

This section describes a frame-based temporal registration algorithm. To reduce the influence of distortions on temporal registration, images are spatially sub-sampled and normalized to have unit variance. This algorithm temporally registers each processed image separately, locating the most similar original image. Some of these individual temporal registration measurements may be incorrect but those errors will tend to be randomly distributed. When delay measurements from a series of images are combined by means of a voting scheme, the overall estimate for the average delay of a video sequence becomes quite accurate. This temporal registration algorithm does not use still and nearly motionless portions of the scene, since the original images are nearly identical to each other.

3.4.2.1 Constants Used by the Algorithm

BELOW_WARN:	Threshold used when examining correlations for deciding if a secondary correlation maximum is sufficiently large so as to indicate ambiguous temporal registration. A BELOW_WARN of 0.9 is recommended.
BLOCK_SIZE:	The sub-sampling factor. Specified in frame lines vertically and pixels horizontally. A BLOCK_SIZE of 16 is recommended.
DELTA:	Secondary maximums in the correlation curve that are within DELTA of the maximum (best) correlation are ignored. A DELTA of 4 is recommended.
HFW:	Half of the filter width for the filter used to smooth the histogram of frame-by-frame temporal registration values. A HFW of 3 is recommended.

STILL_THRESHOLD: A threshold that is used to detect still video scenes (frame-based temporal registration cannot be used on still video scenes). A STILL_THRESHOLD of 0.002 is recommended.

3.4.2.2 Inputs to the Algorithm

A sequence of N original video luminance images: $\mathbf{Y_O}(t)$, $0 \leq t < N$. [10]

A sequence of N processed video luminance images: $\mathbf{Y_P}(t)$, $0 \leq t < N$.

Gain and offset correction factors for the processed luminance images.

Spatial registration information: horizontal shift and vertical shift. For interlace video, the vertical shift for each field determines whether the processed video requires reframing.

Valid region of the processed video sequence (i.e., PVR).

Uncertainty (U): a number indicating the accuracy of the initial temporal registration. The initial temporal registration assumption is that the true temporal registration for $\mathbf{Y_P}(t)$ is within plus or minus (U – HFW) of $\mathbf{Y_O}(t)$, for all $0 \leq t < N$.

3.4.2.3 Frames versus Fields

The frame-based temporal registration algorithm works for both progressive and interlace video. If the video sequence is progressive, the algorithm aligns frames. If the video sequence is interlaced, the algorithm aligns fields. When aligning interlaced video sequences, either frame or reframed alignments are considered but not both. When frame alignments are considered, field one of the processed video is compared to field one of the original video, and field two of the processed video is compared to field two of the original video. When reframed alignments are considered, field one of the processed video is compared to field two of the original video, and field two of the processed video is compared to field one of the original video. The spatial registration values that are input to the algorithm determine whether frame or reframe alignments are considered. The presence of reframing is detected by examining the vertical spatial registration for each field. If the field one vertical shift equals the field two vertical shift, then the processed video is not reframed; only frame alignments are considered. If the field two vertical shift is one greater than the field one vertical shift, only reframe alignments are considered. All other combinations of vertical shifts indicate problems that should be fixed prior to temporal registration.

3.4.2.4 Description of the Algorithm

<u>1. Calibrate the video sequences.</u>

Correct the processed video sequence, $\mathbf{Y_P}(t)$, using the spatial registration and gain-offset information given as inputs to the algorithm.

<u>2. Select the sub-region of video to be used.</u>

The sub-region of interest to be used by the algorithm must be a multiple of the BLOCK_SIZE and must fit within the PVR. The largest sub-region that meets these two requirements and is closest to the center of the image should be selected. All further processing will be limited to video within this selected sub-region of interest.

[10] When interlace video requires reframing, the lengths of the original and processed video sequences must be reduced by one to accommodate the reframing. This will reduce the length of the file by one video frame from N as specified in Figure 2.

3. Spatially sub-sample the original and processed images.

Spatially sub-sample the region of interest of $\mathbf{Y_O}(t)$ and $\mathbf{Y_P}(t)$ by a factor of BLOCK_SIZE by computing the mean of each block. For progressive video frames, the sub-sampling will be BLOCK_SIZE horizontally and vertically, while for interlace video fields, the sub-sampling will be BLOCK_SIZE horizontally and BLOCK_SIZE/2 vertically. For example, sub-sampling a progressive video sequence by a BLOCK_SIZE of 16 will take the mean of each 16 pixel by 16 frame line block, while sub-sampling an interlace video sequence by a BLOCK_SIZE of 16 will take the mean of each 16 pixel by 8 field line block. This sub-sampling reduces the impact of impairments on the temporal registration process.

4. Normalize the sub-sampled images.

Normalize each sub-sampled image by the standard deviation of that image. Skip this normalization for any image where the standard deviation is less than one (e.g., images containing a flat field of color).[11] This normalization will minimize the influence of fluctuations in individual image contrast and energy from influencing the temporal registration results. After this step, the original video and processed video sequences will be denoted as $\mathbf{S_O}(t)$ and $\mathbf{S_P}(t)$, respectively, to denote that the images have been sub-sampled and normalized.

5. Compare processed images to original images.

Compare each processed image, $\mathbf{S_P}(t)$, with the original images $\mathbf{S_O}(t+d)$, where the valid values of d are: $(-U \le d \le +U)$ and the valid values of t are: $(U \le t < N - U)$. For processed image t and original image $t+d$, these comparisons will be denoted as $\mathbf{C}_{td}$ and are computed as the standard deviation over space of the image formed by subtracting processed image t from original image $t+d$: $\mathbf{C}_{td}=std_{\text{space}}(\mathbf{S_O}(t+d)-\mathbf{S_P}(t))$. These comparisons, $\mathbf{C}_{td}$, correlate the t^{th} processed image with each original image that is within the registration uncertainty. Lower values of $\mathbf{C}_{td}$ indicate that the processed image looks more like the original image since more of the image variance is cancelled. The range for t, $U \le t < N – U$, covers all processed images for which original images are available for the entire range of temporal registration uncertainty.

6. Perform an overall check for still video.

To determine if there is sufficient motion in the video sequence, average $\mathbf{C}_{td}$ over time index t for each d:

$$A_d = \frac{1}{N - 2 * U} \sum_{t=U}^{N-U-1} \mathbf{C}_{td} \ .$$

This summation includes the range of processed video images t for which the full uncertainty of original images is available. A_d contains one value for each temporal registration delay d being considered. If (maximum(A_d) - minimum(A_d) < STILL_THRESHOLD), then the scene contains insufficient motion for frame-based temporal registration. The entire scene is still or nearly still. The correlation results from the different video delays are then so similar that any differentiation will be due to random chance rather than reliable measurements. If a still video sequence is detected, the user is given a warning to that effect and the algorithm exits at this point.

[11] Normalization is skipped when the standard deviation is less than one to prevent amplification of noise and to prevent the possibility of dividing by zero for images that contain a flat or uniform intensity level.

7. Temporally register each processed image.

For each processed image t ($U \le t < N - U$), find the d within the temporal uncertainty ($-U \le d \le +U$) that minimizes $\mathbf{C}_{td}$. In other words, for each processed image t, find $d_{\min}(t)$ such that $\mathbf{C}_{t\, d\min(t)} \le \mathbf{C}_{td}$, for all d. The best temporal registration of processed image t is given by $d_{\min}(t)$. Most of the time, the temporal registration indicated for individual images will be correct or very close to correct. The temporal registration will be incorrect for some images due to various reasons (image distortion, errors, noise, insufficient motion, etc.).

8. Perform a stillness check on each processed image.

If for a given processed image t and all values of d ($-U \le d \le U$), maximum($\mathbf{C}_{td}$) - minimum($\mathbf{C}_{td}$) < STILL_THRESHOLD, then $d_{\min}(t)$ is undefined for this processed image t. Specifically, there is insufficient motion around image t for frame-based temporal registration to work properly.

9. Form a histogram of all defined temporal registrations.

Compute a histogram using all the defined values of $d_{\min}(t)$ with 2*U+1 bins where each bin represents a different video delay (i.e., from –U to +U). Values of $d_{\min}(t)$ that are undefined (e.g., still images) are left out of the histogram calculation. This histogram, denoted by H_d, is the histogram of temporal delays for all the processed images that had sufficient motion to perform valid temporal registration. Each bin in the histogram contains the number of processed images with that video delay d, where d can take values from –U to +U.

10. Form a smoothed histogram.

Histogram H_d is smoothed by convolving it with a low pass filter of length 2*HFW+1 and defined at index k as:

$$F_k = \frac{0.5 + 0.5 * \cos\left[\pi * (k - \mathrm{HFW})/(1 + \mathrm{HFW})\right]}{\sum_{i=0}^{2*\mathrm{HFW}} \left\{0.5 + 0.5 * \cos\left[\pi * (i - \mathrm{HFW})/(1 + \mathrm{HFW})\right]\right\}}, \quad 0 \le k \le 2 * \mathrm{HFW}\,.$$

When considering the smoothed histogram SH_d that results from this step, the HFW bins on each end of SH_d are treated as undefined. This restricts the video delays that can be estimated to plus or minus (UNCERTAINTY-HFW). Smoothing of the histogram increases the robustness of the video delay estimates.

11. Examine the histogram information.

From the original histogram, H_d, and the smoothed histogram, SH_d, the following three values are determined:

max_H_value:	The maximum value of H_d.
max_SH_offset:	The offset d that maximizes SH_d.
max_SH_value:	The maximum value of SH_d (e.g., at d = max_SH_offset).

Next, the following two checks are performed:

- Was U large enough? Recall that the first and last HFW bins of H_d are missing from SH_d. Examine the values of H_d in these bins. If (H_d > max_H_value * BELOW_WARN), then the temporal registration uncertainty is too small. The algorithm must be re-run with a larger U. The values of d to check are ($-U \le d < -U + HFW$) and ($U - HFW < d \le U$).

- Does SH_d have one well-defined delay? Examine SH_d, except within DELTA of max_SH_offset. If ($SH_d >$ max_SH_value * BELOW_WARN) for any video delay d where ($-U \le d <$ max_SH_offset – DELTA) or (max_SH_offset + DELTA $< d \le U$), then temporal registration is ambiguous.

If the above two checks pass, then the video delay given by max_SH_offset is chosen as the best average temporal registration for the scene.

3.4.2.5 Observations and Conclusions

The frame-based video delay measurement algorithm uses sub-sampled original and processed video sequences. This algorithm is suitable for aligning video in a fully automated out-of-service environment, prior to performing video quality measurements. The frame-based video delay measurement algorithm estimates the temporal registration for each image, forms histograms of those individual estimates, and then uses the most commonly indicated delay as the overall video delay, or temporal registration, for the selected sequence of video frames.

The delay indicated at the final stage of the algorithm (step 11 of section 3.4.2.4) may be different from the delay a viewer might choose, if aligning the scenes by eye. Viewers tend to focus on motion, aligning the high motion parts of the scene, where the frame-based algorithm chooses the most often observed delay over all of the frames that were examined. These overall delay histograms can be examined to determine the extent and statistics of any variable video delay present in the HRC.

3.4.3 Sequence-Based vs. Frame-Based Temporal Registration

This frame-based temporal registration algorithm operates quite differently from the sequence-based temporal registration algorithm in section 3.4.1. The sequence-based algorithm can be implemented using a very low bandwidth ancillary data channel (see Figure 1) while the frame-based algorithm requires a larger bandwidth ancillary data channel. The sequence-based algorithm is slightly better when examining HRCs that do not contain frame-repeats, provided the scene contains variable changes in motion or luminance. The frame-based algorithm is better when examining videoconferencing quality HRCs with frame repeats and variable video delays. Thus, the sequence-based algorithm might be slightly more suited to VQMs for TV systems (see section 6.1), while the frame-based algorithm might be more suited to VQMs for videoconferencing systems (see section 6.2). If the type of HRC being tested is unknown (see section 6.3), the frame-based algorithm is recommended.

3.4.4 Applying Temporal Registration Correction

All of the quality features will require that the temporal delay calculated herein be removed. For positive delays, remove frames from the beginning of the processed file and the end of the original file. For negative delays, remove frames from the end of the processed file and the beginning of the original file. When reframing interlaced video sequences, the processed sequence is reframed. Thus one field should be removed from the beginning and end of the processed video sequence in addition to the above. Simultaneously, one frame must be removed from either the beginning of the original video file (i.e., -1 field delay overall) or the end of the original video file (i.e., +1 field delay overall).

Correcting for temporal registration will, in effect, shorten the length of available images in the video sequence. For simplicity, all further calculations will be based on the number of video frames available after all calibration corrections have been applied.

4. QUALITY FEATURES

4.1 Introduction

A quality *feature* is defined as a quantity of information associated with, or extracted from, a spatial-temporal sub-region of a video stream (either original or processed). The feature streams that are produced are a function of space and time. By comparing features extracted from the calibrated processed video with features extracted from the original video, a set of quality *parameters* (section 5) can be computed that are indicative of perceptual changes in video quality. This section describes a set of quality features that characterize perceptual changes in the spatial, temporal, and chrominance properties of video streams. Normally, a perceptual filter is applied to the video stream to enhance some property of perceived video quality, such as edge information. After this perceptual filtering, features are extracted from spatial-temporal (S-T) sub-regions using a mathematical function (e.g., standard deviation). Finally, a perceptibility threshold is applied to the extracted features.

For the following discussion, an original feature stream will be denoted as $f_o(s, t)$ and the corresponding processed feature stream will be denoted as $f_p(s, t)$, where s and t are indices that denote the spatial and temporal positions, respectively, of the S-T region within the calibrated original and processed video streams. The features will be assigned lettered subscripts as they are discussed in the following sections, where the subscripted letters are chosen to be indicative of what the feature is measuring. All features operate on frames within a calibrated video sequence (see section 3); any interlace issues are addressed during calibration. All features operate independently of image size (i.e., S-T region size does not change when the image size changes).[12]

In summary, feature calculations perform the following steps. Some features may not require those steps marked [Optional].

1. [Optional] Average multiple video frames (section 4.1.1).
2. [Optional] Apply a perceptual filter.
3. Divide the video stream into S-T regions.
4. Extract features, or summary statistics, from each S-T region (e.g., mean, standard deviation).
5. [Optional] Apply a perceptibility threshold.

Some features may utilize two or more different perceptual filters.

4.1.1 Averaging Multiple Video Frames

Averaging multiple video frames together into a single frame before feature extraction (pre-averaging) may be desirable in some special cases. Pre-averaging, when required, is performed as the first step after calibration. Pre-averaging subtly alters the impairments detected by a feature. Pre-averaging may result in a loss of performance (i.e., objective to subjective correlation) or may simply alter the emphasis of a feature. Pre-averaging is one method for reducing a feature's computational complexity. In particular, pre-averaging is used when designing high-speed video quality models (see section 6.4) that utilize perceptual filters with large spatial extents (e.g., see section 4.2.1). Since convolving these large spatial masks with the sampled images requires many computations, users may be willing to tolerate some loss in performance (i.e., objective to subjective correlation) in exchange for faster computation of VQM.

[12] There is an implicit assumption that the viewing distance as a function of picture height remains fixed (e.g., closer viewing distances are used for smaller images). See section 6 for further comments regarding the assumed viewing distance.

Averaging T video frames into a single video frame before applying the spatial filter reduces the computations by a factor of T.

When averaging multiple video frames together, the number of frames to be averaged is determined by the temporal extent of the feature's S-T region size (see section 4.1.2). Thereafter, the temporal extent of the S-T region becomes one averaged frame. After the frames within the video stream have been pre-averaged, the temporal axis of Y no longer corresponds to individual frames. Rather, the temporal axis contains a number of samples equal to the number of valid frames in the calibrated video sequence divided by T, the number of frames averaged. Pre-averaging is performed on a per-pixel basis. Thus, each pre-averaged frame will be of the same size spatially as the individual frames (i.e., the number of pixels and PVR remain unchanged). For example, when averaging T frames to form averaged frame Y'(t):

$$\mathrm{Y}'(i, j, t) = \frac{1}{\mathrm{T}} \sum_{n=0}^{\mathrm{T}-1} \mathrm{Y}(i, j, t * \mathrm{T} + n) \; .$$

4.1.2 S-T Regions

In general, features are extracted from localized S-T regions after the original and processed video streams have been perceptually filtered. The S-T regions are positioned to divide the video streams into abutting S-T regions. Since the processed video has been calibrated, for each processed video S-T region there exists an original S-T region spanning the identical spatial and temporal position within the video stream. Features are extracted from each S-T region by calculating summary statistics or some other mathematical function over the S-T region of interest.

Each S-T region describes a block of pixels. S-T region sizes are described by (1) the number of pixels horizontally, (2) the number of frame lines vertically, and (3) the number of video frames for 30 fps video.[13] Figure 25 illustrates a S-T region of 8 horizontal pixels x 8 vertical lines x 6 video frames. When applied to 30 fps video, this S-T region spans one fifth of a second and contains 384 pixels. One fifth of a second is a desirable temporal extent, due to the ease of frame rate conversions (i.e., one fifth of a second results in an integer number of video frames for video systems operating at 10 fps, 15 fps, 25 fps, and 30 fps).

The spatial region of interest (SROI, see TERMS AND DEFINITIONS) encompassed by all S-T regions is identical for the original and calibrated processed video sequences. The SROI must lie entirely within the PVR, possibly with a buffer of pixels as required by any convolutional perceptual filter. The horizontal width of the SROI must be evenly divisible by the S-T region's horizontal extent. Likewise, the vertical height of the SROI must be evenly divisible by the S-T region's vertical extent. A user might further constrain the SROI to encompass a region of particular interest, such as the center of the video frame.

Temporally, the original and calibrated processed video sequences are divided into an identical number of S-T regions, beginning with the first frame of temporally aligned video. If the number of valid frames available cannot be evenly divided by the S-T region's temporal extent, frames at the end of the clip are dropped from consideration.

For some features such as those presented in section 4.2, the 8x8_6F block achieves close to maximum correlation with subjective ratings. It should be noted, however, that the correlation decreases *slowly* as one moves away from the optimum S-T region size. Horizontal and vertical widths up to 32 or even

[13] 30 fps and 29.97 fps are used interchangeably in this document. For VQM purposes, the slight difference in frame rate is inconsequential.

larger and temporal widths up to 30 frames can be used with satisfactory results, giving the objective measurement system designer considerable flexibility in adapting the features to the available storage or transmission bandwidth.

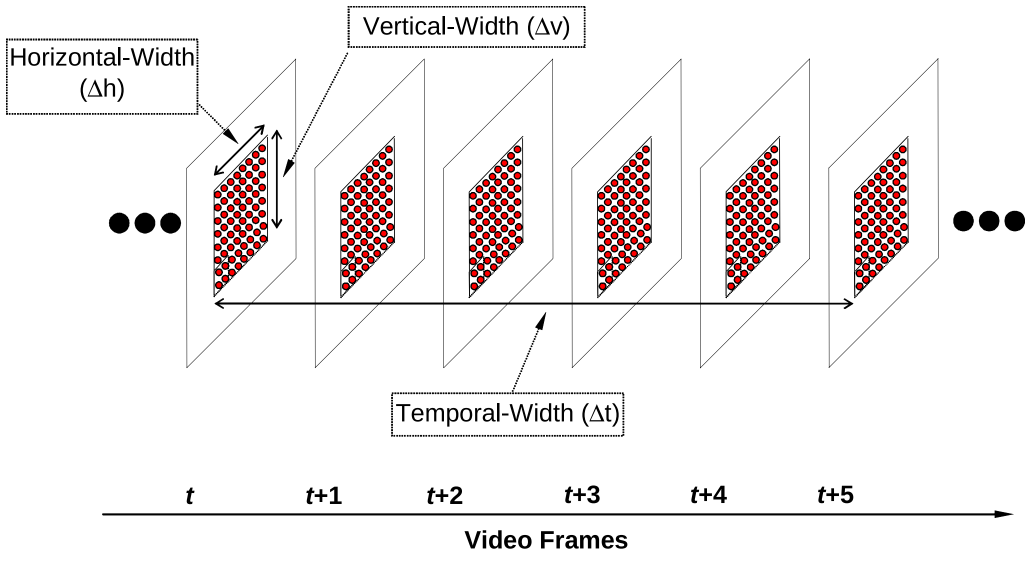

Figure 25. Example spatial-temporal (S-T) region size for extracting features.

After the video stream has been divided into S-T regions, the temporal axis of the feature (*t*) no longer corresponds to individual frames. Rather, the temporal axis contains a number of samples equal to the number of valid frames in the calibrated video sequence divided by the temporal extent of the S-T region.

When computing two or more features simultaneously, further considerations become important. Ideally, all features should be calculated for the same SROI.

4.2 Features Based on Spatial Gradients

Features derived from spatial gradients can be used to characterize perceptual distortions of edges. For example, a general loss of edge information results from blurring while an excess of horizontal and vertical edge information can result from block distortion or tiling. The Y components of the original and processed video streams are filtered using horizontal and vertical edge enhancement filters. Next, these filtered video streams are divided into spatial-temporal (S-T) regions from which features, or summary statistics, are extracted that quantify the spatial activity as a function of angular orientation. Then, these features are clipped at the lower end to emulate perceptibility thresholds. The edge enhancement filters, the S-T region size, and the perceptibility thresholds were selected based on Rec. 601 video that has been subjectively evaluated at a viewing distance of six picture heights. Figure 26 presents an overview of the algorithm used to extract features based on spatial gradients.

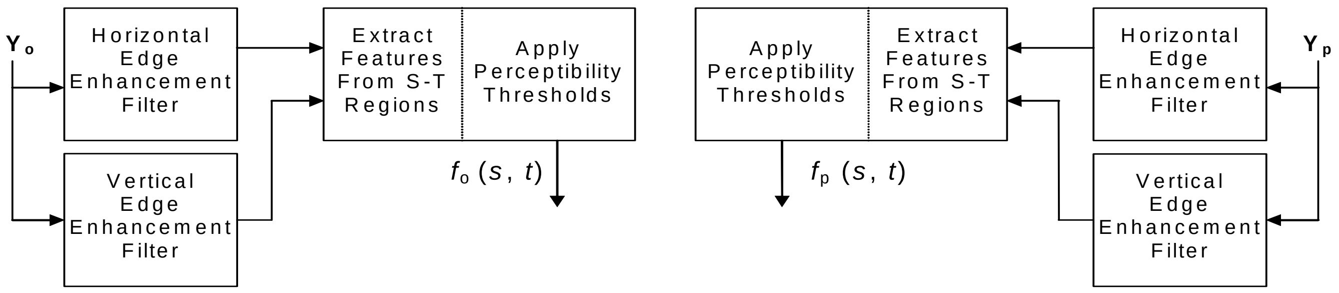
Y o
Horizontal Edge Enhancement Filter
Vertical Edge Enhancement Filter
Extract Features From S-T Regions
Apply Perceptibility Thresholds
f o (s, t)
Apply Perceptibility Thresholds
Extract Features From S-T Regions
f p (s, t)
Horizontal Edge Enhancement Filter
Vertical Edge Enhancement Filter
Y p

Figure 26. Overview of algorithm used to extract spatial gradient features.

4.2.1 Edge Enhancement Filters

The original and processed Y (luminance) video *frames* are first processed with horizontal and vertical edge enhancement filters that enhance edges while reducing noise. The two filters shown in Figure 27 are applied separately, one to enhance horizontal pixel differences while smoothing vertically (left filter), and the other to enhance vertical pixel differences while smoothing horizontally (right filter).

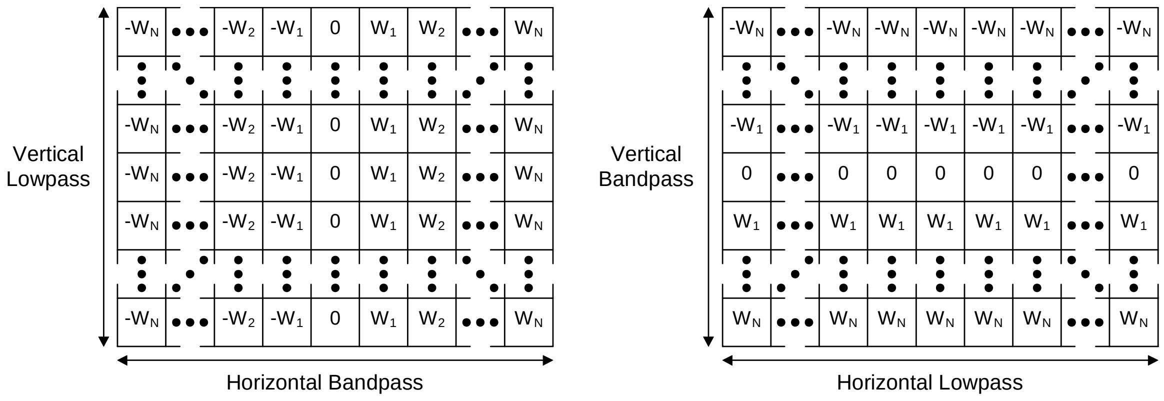

Figure 27. Edge enhancement filters.

The two filters are transposes of each other, have size 13 x 13, and have filter weights given by

$$w_x = k * \left(\frac{x}{c}\right) * \exp\left\{-\left(\frac{1}{2}\right)\left(\frac{x}{c}\right)^2\right\},$$

where *x* is the pixel displacement from the center of the filter (0, 1, 2, …, N), *c* is a constant that sets the width of the bandpass filter, and *k* is a normalization constant selected such that each filter would produce the same gain as a true Sobel filter [19]. The optimal amount of horizontal bandpass filtering for a viewing distance of six times picture height was found to be given by the $c = 2$ filter, which has a peak response at about 4.5 cycles/degree. The bandpass filter weights that were used are given by:

[-.0052625, -.0173446, -.0427401, -.0768961, -.0957739, -.0696751, 0, .0696751, .0957739, .0768961, .0427401, .0173446, .0052625].

Note that the filters in Figure 27 have a flat low-pass response. A flat low-pass response produced the best quality estimate and has the added advantage of being computationally efficient (e.g., for the left filter in Figure 27, one merely has to sum the pixels in a column and multiply once by the weight).

4.2.2 Description of Features f_{SI13} and f_{HV13}

This section describes the extraction of two spatial activity features from S-T regions of the edge-enhanced original and processed video streams from section 4.2.1. These features will be used to detect spatial impairments such as blurring and blocking. The filter shown in Figure 27 (left) enhances spatial gradients in the horizontal (H) direction while the transpose of this filter (right) enhances spatial gradients in the vertical (V) direction. The response at each pixel from the H and V filters can be plotted on a two dimensional diagram such as the one shown in Figure 28 with the H filter response forming the abscissa value and the V filter response forming the ordinate value. For a given image pixel located at row i, column j, and time t, the H and V filter responses will be denoted as $H(i, j, t)$ and $V(i, j, t)$, respectively. These responses can be converted into polar coordinates (R, θ) using the relationships

$$R(i,j,t) = \sqrt{H(i,j,t)^2 + V(i,j,t)^2}\text{ , and}$$

$$\theta(i,j,t) = \tan^{-1}\left[\frac{V(i,j,t)}{H(i,j,t)}\right].$$

The first feature is a measure of overall spatial information (SI) and hence is denoted as f_{SI13} since images were pre-processed using the 13 x 13 filter masks shown in Figure 27. This feature is computed simply as the standard deviation (*std*) over the S-T region of the $R(i, j, t)$ samples, and then clipped at the perceptibility threshold of P (i.e., if the result of the *std* calculation falls below P, f_{SI13} is set equal to P), namely

$$f_{SI13} = \left\{std\left[R(i,j,t)\right]\right\}\Big|_P :\ i,j,t \in \left\{\mathrm{S-T\ Region}\right\}.$$

This feature is sensitive to changes in the overall amount of spatial activity within a given S-T region. For instance, localized blurring produces a reduction in the amount of spatial activity, whereas noise produces an increase. The recommended threshold P for this feature is 12.

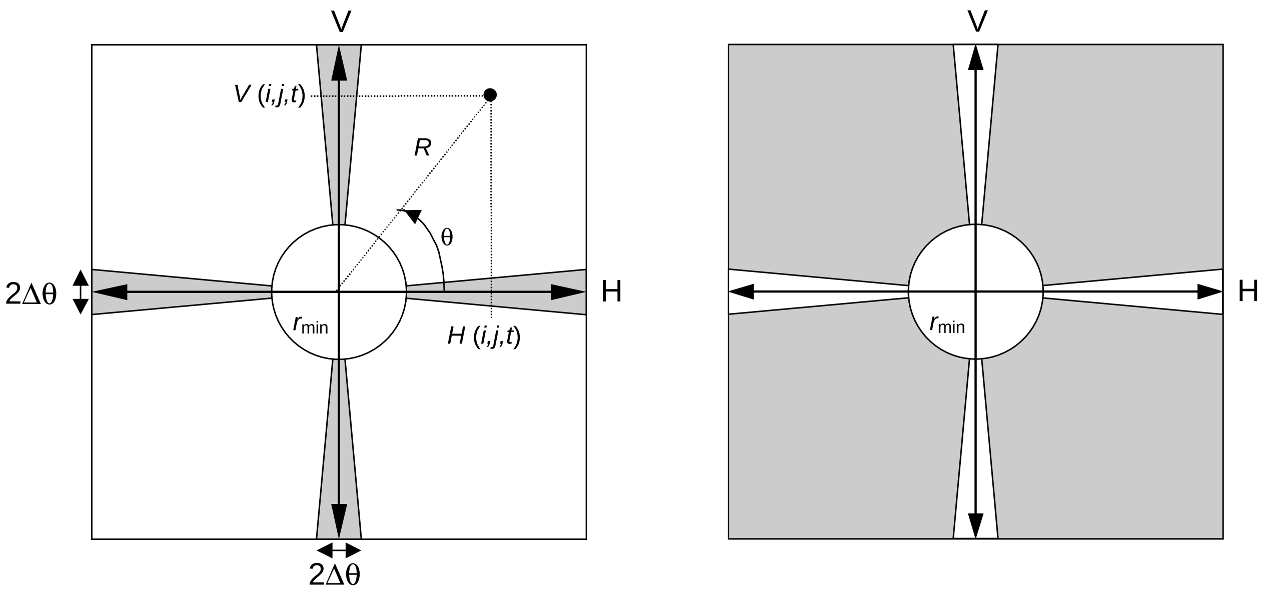

Figure 28. Division of horizontal (H) and vertical (V) spatial activity into *HV* (left) and $\overline{HV}$ (right) distributions.

The second feature, f_{HV13}, is sensitive to changes in the angular distribution, or orientation, of spatial activity. Complementary images are computed with the shaded spatial gradient distributions shown in Figure 28. The image with horizontal and vertical gradients, denoted as *HV*, contains the $R(i, j, t)$ pixels that are horizontal or vertical edges (pixels that are diagonal edges are zeroed). The image with the diagonal gradients, denoted as $\overline{HV}$, contains the $R(i, j, t)$ pixels that are diagonal edges (pixels that are horizontal or vertical edges are zeroed). Gradient magnitudes $R(i, j, t)$ less than $r_{\min}$ are zeroed in both images to assure accurate θ computations. Pixels in *HV* and $\overline{HV}$ can be represented mathematically as

$$HV(i,j,t)=\begin{cases} R(i,j,t) \quad \text{if} \;\; R(i,j,t)\geq r_{\min} \;\; \text{and} \;\; m\dfrac{\pi}{2}-\Delta\theta < \theta(i,j,t) < m\dfrac{\pi}{2}+\Delta\theta \quad (m=0,1,2,3) \\ \\ 0 \quad \text{otherwise} \end{cases},$$

and

$$\overline{HV}(i,j,t)=\begin{cases} R(i,j,t) \quad \text{if} \;\; R(i,j,t)\geq r_{\min} \;\; \text{and} \;\; m\dfrac{\pi}{2}+\Delta\theta \leq \theta(i,j,t) \leq (m+1)\dfrac{\pi}{2}-\Delta\theta \quad (m=0,1,2,3) \\ \\ 0 \quad \text{otherwise} \end{cases}$$

where

$$i,j,t \in \{\text{S}-\text{T Region}\}.$$

For the computation of *HV* and $\overline{HV}$ above, the recommended value for $r_{\min}$ is 20 and the recommended value for Δθ is 0.225 radians. Feature f_{HV13} for one S-T region is then given by the ratio of the mean of

HV to the mean of $\overline{HV}$, where these resultant means are clipped at their perceptibility thresholds *P*, namely

$$f_{HV13} = \left\{mean\left[HV(i,j,t)\right]\right\}\big|_P \Big/ \left\{mean\left[\overline{HV}(i,j,t)\right]\right\}\big|_P .$$

The recommended perceptibility threshold *P* for the mean of *HV* and $\overline{HV}$ is 3. The f_{HV13} feature is sensitive to changes in the angular distribution of spatial activity within a given S-T region. For example, if horizontal and vertical edges suffer more blurring than diagonal edges, f_{HV13} of the processed video will be less than f_{HV13} of the original video. On the other hand, if erroneous horizontal or vertical edges are introduced, say in the form of blocking or tiling distortions, then f_{HV13} of the processed video will be greater than f_{HV13} of the original video. The f_{HV13} feature thus provides a simple means to include variations in the sensitivity of the human visual system with respect to angular orientation.[14]

4.3 Features Based on Chrominance Information

This section presents a single feature that can be used to measure distortions in the chrominance signals (C_B, C_R). For a given image pixel located at row *i*, column *j*, and time *t*, let $C_B(i, j, t)$ and $C_R(i, j, t)$ represent the Rec. 601 C_B and C_R values.[15] The components of a two-dimensional chrominance feature vector, $f_{\text{COHER\_COLOR}}$, are computed as the mean (*mean*) over the S-T region of the $C_B(i, j, t)$ and $C_R(i, j, t)$ samples, respectively, giving more perceptual weight to the C_R component:

$$\underline{f}_{COHER\_COLOR} = \left(mean\left[C_B(i,j,t)\right], W_R * mean\left[C_R(i,j,t)\right]\right):\ i,j,t \in \left\{\text{S} - \text{T Region}\right\}, \text{ and } W_R = 1.5 .$$

The above equation performs coherent integration (hence the name $\underline{f}_{\text{COHER\_COLOR}}$) since the phase relationship between C_B and C_R is preserved. If one is familiar with a vectorscope, the value of the chrominance feature when examining color bar signals is readily apparent. For general-purpose scenes, one can visualize the chrominance feature vector's usefulness for measuring distortions in chrominance for blocks of video that span a range of spatial and temporal extent. However, if S-T region size is too large, then many colors could be included in the calculation, and the usefulness of $\underline{f}_{\text{COHER\_COLOR}}$ is reduced. An S-T region size of 8 horizontal pixels x 8 vertical lines x (1 to 3) video frames produces a robust chrominance feature vector (actually 4 horizontal C_B and C_R pixels, since these signals are sub-sampled by two in the horizontal direction for Rec. 601 sampling).

4.4 Features Based on Contrast Information

Features that measure localized contrast information are sensitive to quality degradations such as blurring (e.g., contrast loss) and added noise (e.g., contrast gain). One localized contrast feature, f_{CONT}, is easily computed for each S-T region from the Y luminance image as

$$f_{CONT} = \left\{std\left[Y(i,j,t)\right]\right\}\big|_P :\ i,j,t \in \left\{\text{S} - \text{T Region}\right\}.$$

The recommended perceptibility threshold *P* for the f_{CONT} feature is between four and six.

[14] This discussion of f_{HV13}, though true in general, is somewhat simplified. For instance, when encountering some shapes the f_{HV13} filter behaves in a manner that may be counter-intuitive (e.g., a corner formed by the joining of a vertical and horizontal line will result in diagonal energy).

[15] Gain and offset corrections are not applied to the C_B and C_R image planes. See section 3.3.4.

4.5 Features Based on Absolute Temporal Information (ATI)

Features that measure distortions in the flow of motion are sensitive to quality degradations such as dropped or repeated frames (motion loss) and added noise (motion gain). An absolute temporal information feature, f_{ATI}, is computed for each S-T region by first generating a motion video stream that is the absolute value of the difference between consecutive video frames at time t and t-1, and then computing the standard deviation over the S-T region. Mathematically, this process will be represented as

$$f_{ATI} = \left\{ std \left| Y(i, j, t) - Y(i, j, t-1) \right| \right\}\Big|_P : i, j, t \in \{\mathrm{S-T\ Region}\}.$$

The recommended perceptibility threshold P for the f_{ATI} feature is between one and three.

The use of a previous frame introduces considerations beyond those required by the other features. When pre-averaging video frames (see section 4.1.1), the difference is taken between the current pre-averaged frame and previous pre-averaged frame. When calculating f_{ATI} jointly with another feature (e.g., $f_{CONTRAST\_ATI}$ from section 4.6) or for use in a model (see section 6), the requirement of an extra frame complicates the task of placement of S-T regions (see section 4.1.2).

4.6 Features Based on the Cross Product of Contrast and Absolute Temporal Information

The perceptibility of spatial impairments can be influenced by the amount of motion that is present. Likewise, the perceptibility of temporal impairments can be influenced by the amount of spatial detail that is present. A feature derived from the cross product of contrast information and absolute temporal information can be used to partially account for these interactions. This feature, denoted as $f_{CONTRAST\_ATI}$, is computed as the product of the features in section 4.4 and 4.5.[16] The recommended perceptibility threshold P = 3 is applied separately to each feature (f_{CONT} and f_{ATI}) before computing their cross product. Impairments will be more visible in S-T regions that have a low cross product than in S-T regions that have a high cross product. This is particularly true of impairments like noise and error blocks [2].

The requirement of an extra frame for f_{ATI} complicates $f_{CONTRAST\_ATI}$ slightly, since the S-T regions used by both f_{CONT} and f_{ATI} must be placed identically. Either one frame at the beginning of the video sequence must be left unused for f_{ATI}, or the S-T regions located at the beginning of the video sequence must contain one fewer frame (e.g., given a temporal extent of 6F, the first f_{ATI} S-T region would use 5F instead of 6F). The parameters and models specified herein presume the second solution will be used.

5. QUALITY PARAMETERS

5.1 Introduction

Quality parameters that measure distortions in video quality due to gains and losses in the feature values are first calculated for each S-T region by comparing the original feature values, $f_o(s, t)$, with the corresponding processed feature values, $f_p(s, t)$ (section 5.2). Several functional relationships are used to emulate the visual masking of impairments for each S-T region. Next, error-pooling functions across space and time emulate how humans deduce subjective quality ratings. Error pooling across space will

[16] A standard cross product of the f_{CONT} and f_{ATI} features (i.e., $f_{CONT} * f_{ATI}$) is used for the processed $f_p(s, t)$ and original $f_o(s, t)$ features in the *ratio_loss* and *ratio_gain* comparison functions described in section 5.2.1. However, for the *log_loss* and *log_gain* comparison functions, the processed and original features are computed as $\log_{10}[f_{CONT}] * \log_{10}[f_{ATI}]$, and the comparison functions use subtraction rather than the logarithm of the ratio (i.e., $f_p(s, t) - f_o(s, t)$ is used rather than $\log_{10}[f_p(s, t) / f_o(s, t)]$).

be referred to as spatial collapsing (section 5.3), and error pooling across time will be referred to as temporal collapsing (section 5.4). Sequential application of the spatial and temporal collapsing functions to the stream of S-T quality parameters produces quality parameters for the entire video clip, which is nominally 5 to 10 seconds in duration. The final time-collapsed parameter values may be scaled and clipped (section 5.5) to account for nonlinear relationships between the parameter value and perceived quality and to further reduce the parameter's sensitivity.

In summary, parameter calculations perform the following steps. Some features may not require the [Optional] step.

1. Compare original feature values with processed feature values.
2. Perform spatial collapsing.
3. Perform temporal collapsing.
4. [Optional] Perform nonlinear scaling and/or clipping.

All parameters are designed to be either all positive or all negative. A parameter value of zero indicates no impairment.

5.2 Comparison Functions

The perceptual impairment at each S-T region is calculated using functions that model visual masking of the spatial and temporal impairments. This section presents the masking functions that are used by the various parameters to produce quality parameters as a function of space and time.

5.2.1 Error Ratio and Logarithmic Ratio

Loss and gain are normally examined separately, since they produce fundamentally different effects on quality perception (e.g., loss of spatial activity due to blurring and gain of spatial activity due to noise or blocking). Of the many comparison functions that have been evaluated, two forms have consistently produced the best correlation to subjective ratings. Each of these forms can be used with either gain or loss calculations for a total of four basic S-T comparison functions. The four primary forms are:

$$ratio\_loss(s,t) = np\left\{\frac{f_p(s,t) - f_o(s,t)}{f_o(s,t)}\right\},$$

$$ratio\_gain(s,t) = pp\left\{\frac{f_p(s,t) - f_o(s,t)}{f_o(s,t)}\right\},$$

$$\log\_loss(s,t) = np\left\{\log_{10}\left[\frac{f_p(s,t)}{f_o(s,t)}\right]\right\}, \text{ and}$$

$$\log\_gain(s,t) = pp\left\{\log_{10}\left[\frac{f_p(s,t)}{f_o(s,t)}\right]\right\},$$

where *pp* is the positive part operator (i.e., negative values are replaced with zero), and *np* is the negative part operator (i.e., positive values are replaced with zero).

These visual masking functions imply that impairment perception is inversely proportional to the amount of localized spatial or temporal activity that is present. In other words, spatial impairments become less visible as the spatial activity increases (i.e., spatial masking), and temporal impairments become less visible as the temporal activity increases (i.e., temporal masking). While the logarithmic and ratio comparison functions behave very similarly, the logarithmic function tends to be slightly more advantageous for gains while the ratio function tends to be slightly more advantageous for losses. The logarithm function has a larger dynamic range, and this is useful when the processed feature values greatly exceed the original feature values.

5.2.2 Euclidean Distance

Another useful S-T comparison function is simple Euclidean distance, represented by the length of the difference vector between the original feature vector $f_o(s, t)$ and the corresponding processed feature vector, $f_p(s, t)$,

$$euclid(s,t) = \left\| \underline{f}_p(s,t) - \underline{f}_o(s,t) \right\|.$$

Figure 29 gives an illustration of Euclidean distance for a two-dimensional feature vector extracted from a S-T region (e.g., the $f_{\mathrm{COHER\_COLOR}}$ feature vector of section 4.3), where s and t are indices that denote the spatial and temporal positions, respectively, of the S-T region within the calibrated original and processed video streams. The dashed line in Figure 29 shows the Euclidean distance. The Euclidean distance measure can be generalized for feature vectors that have an arbitrary number of dimensions.

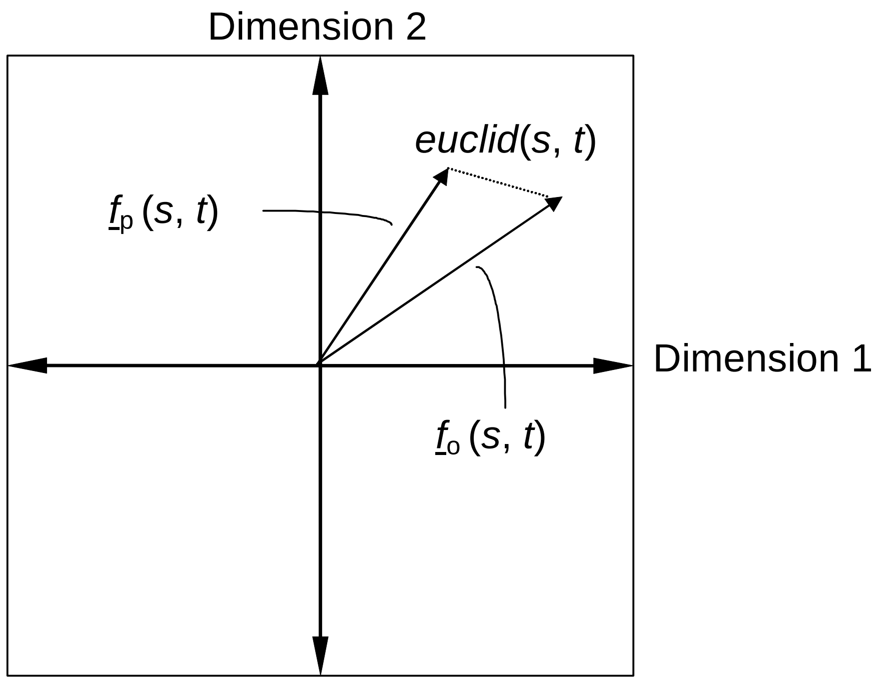

Figure 29. Illustration of the Euclidean distance *euclid*(*s*, *t*) for a two-dimensional feature vector.

5.3 Spatial Collapsing Functions

The parameters from the S-T regions (from section 5.2) form three-dimensional matrixes spanning one temporal axis and two spatial dimensions (i.e., horizontal and vertical placement of the S-T region). Next, impairments from the S-T regions with the same time index t are pooled using a spatial collapsing function. Spatial collapsing yields a time history of parameter values. This time history of parameter values, denoted generically as $p(t)$, must then be temporally collapsed using a temporal collapsing function given in section 5.4. Table 4 presents a summary of the most commonly used spatial collapsing functions.

Extensive investigation has revealed that the optimal spatial collapsing functions normally involve some form of worst case processing, like the average of the worst 5% of the distortions observed over the spatial index *s* ([32]-[34]). This is because localized impairments tend to draw the focus of the viewer, making the worst part of the picture the predominant factor in the subjective quality decision. For example, the spatial collapsing function "*above95%*" is computed at each temporal index *t* for the *log_gain*(*s,t*) function in section 5.2.1 as the average of the most positive 5% of the values over the spatial index *s*.[17] This amounts to sorting the gain distortions from low to high at each temporal index *t* and averaging those distortions that are above the 95% threshold (since more positive values imply greater distortion). Similarly, loss distortions such as those produced by the *ratio_loss*(*s,t*) function in section 5.2.1 would be sorted at each temporal index *t*, but the average of those distortions that are "*below5%*" is used (since losses are negative).

5.4 Temporal Collapsing Functions

The parameter time history results *p*(*t*) output from the spatial collapsing function (from section 5.3) are next pooled using a temporal collapsing function to produce an objective parameter *p* for the video clip, which is nominally 4 to 10 seconds in length. Viewers seem to use several temporal collapsing functions when subjectively rating video clips that are approximately 10 seconds in length. The *mean* over time is indicative of the average quality that is observed during the time period. The *90%* and *10%* levels over time are indicative of the worst transient quality that is observed for gains and losses, respectively (e.g., digital transmission errors may cause a 1 to 2 second disturbance in the processed video). After temporal collapsing, a given parameter *p* is either all negative or all positive, but not both. Table 5 presents a summary of the most commonly used temporal collapsing functions.

[17] Notice that the time index, *t*, does not indicate individual frames (see section 4.1.2) here. Instead, each value of *t* corresponds to those S-T regions having the same time extent.

Table 4. Spatial Collapsing Functions and Their Definitions

Spatial Collapsing Function	Definition
below5%	For each temporal index *t*, sort the parameter values from low to high. Compute the average of all the parameter values that are less than or equal to the 5% threshold level. For loss parameters, this spatial collapsing function produces a parameter that is indicative of the worst quality over space.
above95%	For each temporal index *t*, sort the parameter values from low to high. Compute the average of all the parameter values that are greater than or equal to the 95% threshold level. For gain parameters, this spatial collapsing function produces a parameter that is indicative of the worst quality over space.
mean	For each temporal index *t*, compute the average of all the parameter values. This spatial collapsing function produces a parameter that is indicative of the average quality over space.
std	For each temporal index *t*, compute the standard deviation of all the parameter values. This spatial collapsing function produces a parameter that is indicative of the quality variations over space.
below5%tail	For each temporal index *t*, sort the parameter values from low to high. Compute the average of all the parameter values that are less than or equal to the 5% threshold level, and then subtract the 5% level from this average. For loss parameters, this spatial collapsing function allows one to measure the spread of the worst quality levels over space. It is useful for measuring the perceptual quality effects of spatially localized distortions.
above99%tail	For each temporal index *t*, sort the parameter values from low to high. Compute the average of all the parameter values that are greater than or equal to the 99% threshold level, and then subtract the 99% level from this average. For gain parameters, this spatial collapsing function allows one to measure the spread of the worst quality levels over space. It is useful for measuring the perceptual quality effects of spatially localized distortions.

Table 5. Temporal Collapsing Functions and Their Definitions

Temporal Collapsing Function	**Definition**
10%	Sort the time history of the parameter values from low to high and select the 10% threshold level. For loss parameters, this temporal collapsing function produces a parameter that is indicative of the worst quality over time. For gain parameters, it produces a parameter that is indicative of the best quality over time.
25%	Sort the time history of the parameter values from low to high and select the 25% threshold level.
50%	Sort the time history of the parameter values from low to high and select the 50% threshold level.
90%	Sort the time history of the parameter values from low to high and select the 90% threshold level. For loss parameters, this temporal collapsing function produces a parameter that is indicative of the best quality over time. For gain parameters, it produces a parameter that is indicative of the worst quality over time.
mean	Compute the mean of the time history of the parameter values. This produces a parameter that is indicative of the average quality over time.
std	Compute the standard deviation of the time history of the parameter values. This temporal collapsing function produces a parameter that is indicative of the quality variations over time.
above90%tail	Sort the time history of the parameter values from low to high and compute the average of all the parameter values that are greater than or equal to the 90% threshold level, and then subtract the 90% level from this average. For gain parameters, this temporal collapsing function allows one to measure the spread of the worst quality levels over time. It is useful for measuring the perceptual quality effects of temporally localized distortions.

5.5 Nonlinear Scaling and Clipping

The all-positive or all-negative temporally collapsed parameter *p* from section 5.4 may be scaled to account for nonlinear relationships between the parameter value and perceived quality. It is preferable to remove any nonlinear relationships before building the video quality models (section 6), since a linear least-squares algorithm will be used to determine the optimal parameter weights. The two nonlinear scaling functions that might be applied are the square root function, denoted by *sqrt*, and the square function, denoted by *square*. If the *sqrt* function is applied to an all-negative parameter, the parameter is first made all positive (i.e., absolute value taken).

Finally, a clipping function denoted as *clip_T*, where *T* is the clipping threshold, might be applied to reduce the sensitivity of the parameter to small impairments. The clipping function replaces any parameter value between the clipping level and zero with the clipping level, and then the clipping level is subtracted from all resulting parameter values. This is represented mathematically as

$$clip\_T(p) = \begin{cases} \max(p,T) - T & if\ \ p\ is\ all\ \ positive \\ \min(p,T) - T & if\ \ p\ is\ all\ negative \end{cases} .$$

5.6 Parameter Naming Convention

This section summarizes the technical naming convention used for video quality parameters. This convention assigns to each parameter a lengthy name consisting of identifying words (sub-names) separated by underscores. The technical parameter name summarizes the exact process used to calculate the parameter. Each sub-name identifies one function or step in the process of calculating the parameter. Sub-names are listed in the order in which they occur, from left to right. Table 6 summarizes the sub-names used to create the technical parameter name, listed in the order that they occur. Section 5.6.1 provides examples of technical parameter names and their associated sub-names from Table 6.

Table 6. Technical Naming Convention Used for Video Quality Parameters

Sub-name	Definition	Examples
Average Multiple Video Frames	Present only when the first step of the parameter is to average multiple video frames (see section 4.1.1). Sub-name consists of (1) *avg*, an abbreviation for average, (2) the number of frames to be averaged, and (3) the letter *F*, indicating that frames have been averaged. When absent, the “Block Frames” sub-name must be present.	*avg18F* *avg6F* *avg2F*
Color	The color space image planes used by the parameter.	*Y* for luminance image plane *color* for (C_B, C_R) image planes
Feature Specific	The “Feature Specific” sub-name describes the calculations that make this parameter unique. All other sub-names that follow are generic processes that can be used by many different types of parameters. The “Feature Specific” sub-name is usually the name of the feature that is extracted from the “Color” plane at this point in the flow, hence the location of this sub-name. However, information not otherwise covered by the naming convention can also be	*si13* for the f_{SI13} feature in section 4.2.2 *hv13_angleX.XXX_rminYY* for the f_{HV13} feature in section 4.2.2, where *X.XXX* is $\Delta\theta$ and *YY* is the $r_{\min}$

	included here. For example, the HV parameter applies the "Block Statistic" sub-name separately to the HV and $\overline{HV}$ image planes. The subsequent ratio of HV to $\overline{HV}$ is specified by the "Feature Specific" sub-name (i.e., rather than occupying a separate sub-name after the "Block Statistic").	*coher_color* for the f_{COHER_COLOR} feature in section 4.3 *cont* for the f_{CONT} feature in section 4.4 *ati* for the f_{ATI} feature in section 4.5 *contrast_ati* for the $f_{CONTRAST\_ATI}$ feature in section 4.6
Block Shift	Present when S-T blocks slide (e.g., overlap in time). When absent, blocks are assumed to abut in time.	*sliding*
Full Image	Present when the S-T block size contains the entire valid region of the image. When absent, the "Block Size" sub-name must be present.	*image*
Block Size	Present when the image is divided into S-T blocks (see 4.1.2). For consistency, block size is always indicated relative to the luminance (Y) plane's frame lines and frame pixels. Thus, for 4:2:2 sampled video, color blocks will actually contain half the specified number of pixels horizontally. When absent, the "Full Image" sub-name must be present.	*8x8* for blocks that include 8 frame lines vertically by 8 frame pixels horizontally *128x128* for blocks that include 128 frame lines vertically by 128 frame pixels horizontally
Block Frames	Indicates the temporal extent of the S-T blocks (see 4.1.2), referenced to 30 frames per second (fps) video. For example, *6F* is used to represent one fifth of a second, regardless of the frame rate of the video being measured (e.g., 5 frames from a 25 fps system, 3 frames from a 15 fps system, 2 frames from a 10 fps system). When absent, the "Average Multiple Video Frames" sub-name must be present.	*1F* for a temporal extent of one frame *2F* for a temporal extent of two frames *6F* for a temporal extent of one fifth of a second *18F* for a temporal extent of three fifths of a second
Block Statistic	The statistical function used to extract the feature from each S-T region, producing one number for each S-T block of pixels. Present unless "Block Size" = *1x1* (i.e., 1 pixel). Before the Block Statistic has been applied, intermediate results contain time histories of images with one number per pixel (i.e., filtered images); afterward, intermediate results contain one number per each S-T region (i.e., feature images). Parameters that have two image planes (e.g., *hv13* and *coher_color*) will apply the Block Statistic separately to both image planes, producing two feature images.	*mean* is the average of the pixel values *std* is the standard deviation of the pixel values *rms* is the root mean square of the pixel values

Perceptibility Threshold	The values produced by the "Block Statistic" may be clipped at a perceptibility threshold *P*. Values between zero and this threshold are replaced with the threshold.	*3* for a minimum feature value of 3.0 *12* for a minimum feature value of 12.0
Comparison Function	The function used to compare features extracted from the original and processed feature streams (see section 5.2). Before the Comparison Function, the intermediate results contain time histories of original and processed feature images; afterward the intermediate results contain a time history of parameter images.	*log_gain* (see section 5.2.1) *ratio_loss* (see section 5.2.1) *euclid* (see section 5.2.2).
Spatial Collapsing Function	See section 5.3. The function is applied to each parameter image (e.g., all S-T regions having the same temporal index) and produces a time history of parameter values. Before spatial collapsing, intermediate results consist of parameter images containing one value for each S-T block; afterward, intermediate results are a time history of numbers (i.e., parameter time history). Must be present for all parameters except "Full Image" parameters.	See Table 4
Temporal Collapsing Function	See section 5.4. The function is applied to the parameter time history and produces one parameter value for the entire video sequence. After temporal collapsing, the parameter contains either all negative values or all positive values, but not both. Zero is associated with no impairment, and parameter values further from zero have higher impairments. Must be present for all parameters.	See Table 5
Nonlinear Function	See section 5.5. Examination of the parameter's values may indicate that the parameter should be scaled in a nonlinear fashion to linearly track the subjective data. The Nonlinear Function performs this final scaling. If the *sqrt* function is applied to an all-negative parameter, the parameter is first made all positive (i.e., absolute value taken).	*sqrt* for the square root of the temporally collapsed parameter value *square* for the square of the temporally collapsed parameter value
Clipping Function	See section 5.5. Final examination of the parameter values may indicate a need to further reduce the sensitivity of the parameter to small impairments (e.g., parameter values near zero). Replace any value between the clipping level *T* and zero with the clipping level, and then subtract the clipping level from all resulting parameter values.	*clip_0.45* If parameter values are positive, replace all values less than 0.45 with 0.45 and then subtract 0.45 from all the parameter values. If parameter values are negative, replace all values greater than –0.45 with –0.45 and then add 0.45 to all the parameter values.

5.6.1 Example Parameter Names

This section includes five example technical names, and a step-by-step description of the sub-naming procedure given in Table 6.

avg18F_Y_si13_8x8_std_6_ratio_loss_below5%_mean

avg18F_Y represents a pixel-by-pixel averaging of eighteen consecutive original luminance (Y) frames, and eighteen consecutive processed luminance frames. *si13* represents filtering of those averaged images with the 13x13 spatial masks in section 4.2.1 in preparation for extraction of the f_{SI13} feature in section 4.2.2. *8x8* represents dividing each filtered image into blocks that are 8 frame lines high by 8 pixels wide. *std* represents taking the standard deviation of each block. *6* represents application of a perceptibility threshold, replacing all standard deviation values below 6.0 with a value of 6.0. *ratio_loss* represents comparing the original and processed features from each block using the *ratio_loss* function. *below5%* represents spatially collapsing the parameter values at each time index using the *below5%* function. *mean* represents temporally collapsing the parameter time history using the *mean* function.

color_coher_color_8x8_1F_mean_euclid_std_10%_clip_0.8

color represents using the C_B and C_R image planes. *coher_color* represents preservation of the phase relationship between the C_B and C_R images (by treating them separately) in preparation for extraction of the f_{COHER_COLOR} feature in section 4.3. *8x8_1F* represents dividing each frame into blocks that are 8 frame lines high by 4 C_B and C_R pixels wide (due to 4:2:2 subsampling of the C_B and C_R image planes) by 1 frame in time. *mean* represents taking the mean value of each block. *euclid* represents computing the Euclidean distance between original vectors (C_B, C_R) and processed vectors (C_B, C_R) for each S-T block. *std* represents the *std* spatial collapsing function. *10%* represents the *10%* temporal collapsing function. *clip_0.8* represents clipping the final parameter value at a minimum of 0.8 (i.e., replacing all values below 0.8 with 0.8, and then subtracting 0.8).

Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_ratio_loss_below5%_mean_square_clip_0.05

Y means that the luminance image plane is used. *hv13* represents filtering of the Y images with the 13x13 spatial masks in section 4.2.1 in preparation for extraction of the f_{HV13} feature in section 4.2.2 (i.e., the *HV* and $\overline{HV}$ images are created and treated separately until after the Perceptibility Threshold). *angle0.225* and *rmin20* represents a $\Delta\theta$ of 0.225 radians and an $r_{\min}$ of 20 for calculation of the f_{HV13} feature. *8x8_6F* represents dividing the video stream into S-T regions containing eight frame lines vertically by eight pixels horizontally by six frames temporally. *mean* represents taking the mean value of each S-T block for *HV* and $\overline{HV}$. *3* represents the application of a perceptibility threshold to these means, replacing all values less than 3.0 with 3.0. Next, the f_{HV13} feature in section 4.2.2 is calculated as the ratio of clipped means of *HV* to the clipped means of $\overline{HV}$, as specified in *hv13_angle0.225_rmin20*, the Feature Specific sub-name. *ratio_loss* represents using the *ratio_loss* comparison function for each original and corresponding processed f_{HV13} feature extracted from a S-T block. *below5%* specifies the spatial collapsing function. *mean* specifies the temporal collapsing function. *square* specifies the nonlinear function for each time-collapsed parameter value. *clip_0.05* represents the clipping function, where all values below 0.05 are replaced with 0.05, and then 0.05 is subtracted from the result (recall that the all-negative parameter will become an all-positive parameter due to the nonlinear function, *square*).

Y_contrast_ati_4x4_6F_std_3_ratio_gain_mean_10%

Y means the luminance plane is used. *contrast_ati* represents computing two separate filtered versions of the image in preparation for extraction of the $f_{CONTRAST\_ATI}$ feature in section 4.6. The first filter, *contrast*, will consider the luminance planes directly (section 4.4). The second filter, *ati*, will consider images generated by taking differences between successive luminance planes (section 4.5). The *contrast* and *ati* images are treated separately until after the Thresholding. *4x4_6F* means that the two video streams are divided into S-T regions containing four frame lines vertically by four pixels horizontally by six frames temporally. The first S-T block of *ati* images will actually contain only 5 images rather than 6 since an *ati* image cannot be generated for the first frame in the sequence (i.e., there is no earlier image in time available). This exception is specified as part of the Feature Specific sub-name. *std* represents taking the standard deviation of each block. Then, as specified in the Feature Specific sub-name in section 4.6, apply a perceptibility threshold of 3 to both the *contrast* and *ati* features (replace all values less than 3 with 3.0). Next, multiply the *contrast* block-value with the *ati* block-value for each S-T block (see footnote in section 4.6 for special instructions on how to perform this multiplication) and continue calculations with this combined feature image. *ratio_gain* is the comparison function used to compare each original and processed feature from the S-T blocks. *mean* is the spatial collapsing function. *10%* is the temporal collapsing function.

Y_cont_16x16_2F_std_4_log_gain_above99%tail_above90%tail_sqrt

This parameter uses the *Y* luminance plane. *cont* (abbreviation for contrast) represents preparing the image for extraction of the f_{CONT} feature in section 4.4. *16x16_2F* represents dividing the video stream into S-T regions containing sixteen frame lines vertically by sixteen pixels horizontally by two frames temporally. *std* represents taking the standard deviation of each S-T block. *4* represents application of a perceptibility threshold, where all values less than 4 are replaced with 4.0. *log_gain* is the comparison function used to compare each original and processed feature from the S-T blocks. *above99%tail* is the spatial collapsing function. *above90%tail* is the temporal collapsing function. *sqrt* is the nonlinear function that is applied to the temporally collapsed parameter.

6. VQM MODELS

A VQM model is a particular algorithm used to compute VQM that has been specifically optimized to achieve maximum objective to subjective correlation based upon certain optimization criteria, including the range of quality over which the model applies and the speed of computation. This section describes five VQM models:

1. Television (section 6.1) – optimized for television (e.g., MPEG-2).
2. Videoconferencing (section 6.2) – optimized for videoconferencing (e.g., H.263, MPEG-4).
3. General (section 6.3) – optimized using a wide range of video quality and bit rates.
4. Developer (section 6.4) – optimized using a wide range of video quality and bit rates with the added constraint of fast computation.
5. Peak-signal-to-noise-ratio (PSNR) – optimized using a wide range of video quality and bit rates.

VQM models 1 to 5 consist of a linear combination of video quality parameters whose naming conventions are described in section 5.6. The selection of video quality parameters was determined by the optimization criteria given above. The fifth model uses the peak-signal-to-noise-ratio (PSNR) measurement that is described in detail in ([3], [8]). The methods and procedures used to determine the optimal parameter weights for each of the models are given in section 8. Each VQM model produces output values that range from zero (no perceived impairment) to approximately one (maximum perceived impairment).

VQM models 1 to 4 were designed based on Rec. 601 video that has been subjectively evaluated at a viewing distance of six picture heights. When analyzing video sequences for different viewing distances, a scaling factor must be applied to the VQM results. As viewing distance increases, impairments become less visible; as viewing distance decreases, impairments become more visible. Care should be taken when comparing VQM results for video sequences that will be viewed at different viewing distances.

6.1 Television VQM (VQM$_T$)

This section provides a full description of the television model VQM (henceforth denoted as VQM$_T$), which has been designed to track subjective quality judgments of video scenes from digital television systems. The television model has objective parameters for measuring the perceptual effects of the usual types of television impairments including blurring, block distortion, noise (in both the luminance and chrominance channels), and error blocks (e.g., what might typically be seen when digital transmission errors are present).[18]

Results are presented in section 8.1 that compare VQM$_T$ with mean opinion scores from seven different subjective tests that spanned many different scenes and digital television systems (e.g., contribution-quality and distribution-quality television applications with bit rates > 1.5 Mb/s).

VQM$_T$ consists of a linear combination of nine parameters. Four parameters are based on features extracted from spatial gradients of the Y luminance component (section 4.2.2), one parameter is based on features extracted from the vector formed by the two chrominance components (C_B, C_R) (section 4.3), and four parameters are based on contrast features extracted from the Y luminance component (section 4.4). VQM$_T$ is computed as

VQM$_T$ =

{-0.1582 * Y_si13_8x8_6F_std_12_ratio_loss_below5%_10%

+0.3039 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_ratio_loss_below5%_10%_square_clip_0.06

+0.3307 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_log_gain_above95%_25%_clip_0.13

+0.0310 * color_coher_color_8x8_1F_mean_euclid_std_10%_clip_0.6

-3.4032 * [Y_si13_8x8_6F_std_8_log_gain_mean_mean |$^{0.10}$]

+0.1503 * Y_cont_16x16_2F_std_4_log_gain_std_std_sqrt

+2.0097 * Y_cont_16x16_2F_std_4_log_gain_below5%_50%_sqrt

-2.1356 * Y_cont_16x16_2F_std_6_ratio_loss_below5%tail_90%_clip_0.012

+0.3874 * Y_cont_16x16_2F_std_4_log_gain_above99%tail_above90%tail_sqrt} |$_{0.0}$

The square on the hv_loss parameter is necessary to linearize the parameter response with respect to the subjective data. Note that since the hv_loss parameter becomes positive after the square, a positive multiplying weight is used. Also note that the hv_loss parameter is clipped at 0.06, the hv_gain parameter is clipped at 0.13, the color parameter is clipped at 0.6, and the Y_cont_loss parameter is clipped at -0.012. The three Y_cont_gain parameters require a square root function to linearize their responses with respect to the subjective data. The si_gain parameter is the only quality *improvement* parameter in the model (since the si_gain parameter is positive, a negative weight results in negative

[18] In the experience of the authors, robust objective parameters for measuring the full perceptual impact of error blocks are extremely difficult to develop and integrate into an overall model. This is because error blocks (such as might result from error bursts) are very localized in space and time. Setting the spatial and temporal collapsing functions to properly detect error blocks makes the parameters too sensitive to other types of impairments, and thus their weight in the overall model must be reduced to prevent over-penalization. To date, the best spatial and temporal collapsing functions the authors have found for error block detection are based on the "tail" function described in sections 5.3 and 5.4.

contributions to VQM which produce quality improvements). The si_gain parameter measures improvements to quality that result from edge sharpening or enhancement. Clipping of the parameter at an *upper* threshold of 0.10 (specified by the notation $|^{0.10}$) before multiplying by the parameter weight prevents excessive improvements to VQM of more than about 1/3 of a quality unit, which is the maximum improvement observed in the television subjective data set (i.e., an HRC will only be rewarded for a little edge enhancement).

The total VQM after the contributions of all the parameters are added up is clipped at a lower threshold of 0.0 (specified by the notation $|_{0.0}$) 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 currently available subjective data.

If $VQM_T > 1.0$, then $VQM_T = (1 + c)*VQM_T / (c + VQM_T)$, where $c = 0.5$.

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

6.2 Videoconferencing VQM (VQM_V)

This section provides a full description of the videoconferencing model VQM (henceforth denoted as VQM_V), which has been designed to track subjective quality judgments of video scenes from videoconferencing systems. The videoconferencing model has objective parameters for measuring the perceptual effects of the usual types of videoconferencing impairments including blurring, block distortion, and jerky/unnatural motion. While one objective parameter for measuring noise has been included in this model (ati_gain), the parameter does not fully capture the perceptual effects of added noise. Objective parameters for measuring error blocks and color distortions have not been included in the videoconferencing model since these types of impairments are normally of much less perceptual impact than blurring, block distortion, and jerky/unnatural motion. If noise, error blocks, or color distortions are present, one should use the general model given in section 6.3. Results are presented in section 8.2 that compare VQM_V with mean opinion scores from four different subjective tests that spanned many different scenes and videoconferencing systems (e.g., videoconferencing applications from 10 kb/s to 1.5 Mb/s).

VQM_V consists of a linear combination of six parameters. Four parameters are based on features extracted from spatial gradients of the Y luminance component (section 4.2.2), and two parameters are based on features extracted from the pre-averaged absolute temporal information of the Y luminance component (section 4.5). VQM_V is given by

VQM_V =

{-0.1656 * Y_si13_8x8_6F_std_12_ratio_loss_below10%_10%

+ [0.8452 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_ratio_loss_below5%_mean_square

-0.9817 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_ratio_loss_below50%tail_mean_square] $|_{0.0}$

+0.2954 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_log_gain_above95%_25%_clip_0.45

+0.6274 * avg6F_Y_ati_32x32_std_1_log_gain_mean_10%

-0.1224 * avg6F_Y_ati_32x32_std_3_ratio_loss_below5%_10%}

The square on the hv_loss parameters is necessary to linearize the parameter responses with respect to the subjective data. Note that a combined hv_loss parameter is used that includes two different spatial collapsing functions. This combined hv_loss parameter is clipped at a lower threshold of 0.0 to prevent negative VQM numbers. Also note that the hv_gain parameter is clipped at 0.45.

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 currently available subjective data.

If $VQM_V > 1.0$, then $VQM_V = (1 + c)*VQM_V / (c + VQM_V)$, where $c = 0.5$.

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

6.3 General VQM (VQM_G)

This section provides a full description of the general model VQM (henceforth denoted as VQM_G), which has been designed to track subjective quality judgments of video scenes that can span a very wide range of quality levels. The general model has objective parameters for measuring the perceptual effects of a wide range of impairments such as blurring, block distortion, jerky/unnatural motion, noise (in both the luminance and chrominance channels), and error blocks (e.g., what might typically be seen when digital transmission errors are present, see footnote 18). Results are presented in section 8.3 that compare VQM_G with mean opinion scores from eleven different subjective tests that spanned many different scenes, video systems, and coding technologies. Seven of these data sets contained mostly video scenes from contribution-quality and distribution-quality television applications (> 1.5 Mb/s) while four of these data sets contained mostly video scenes from videoconferencing applications (from 10 kb/s to 1.5 Mb/s).

VQM_G consists of a linear combination of seven parameters. Four parameters are based on features extracted from spatial gradients of the Y luminance component (section 4.2.2), two parameters are based on features extracted from the vector formed by the two chrominance components (C_B, C_R) (section 4.3), and one parameter is based on contrast and absolute temporal information features, both extracted from the Y luminance component (sections 4.4 and 4.5, respectively). VQM_G is given by

VQM_G =

{-0.2097 * Y_si13_8x8_6F_std_12_ratio_loss_below5%_10%

+0.5969 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_ratio_loss_below5%_mean_square_clip_0.06

+0.2483 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_log_gain_above95%_mean

+0.0192 * color_coher_color_8x8_1F_mean_euclid_std_10%_clip_0.6

-2.3416 * [Y_si13_8x8_6F_std_8_log_gain_mean_mean_clip_0.004 $|^{0.14}$]

+0.0431 * Y_contrast_ati_4x4_6F_std_3_ratio_gain_mean_10%

+0.0076 * color_coher_color_8x8_1F_mean_euclid_above99%tail_std} $|_{0.0}$

The square on the hv_loss parameter is necessary to linearize the parameter response with respect to the subjective data. Note that since the hv_loss parameter becomes positive after the square, a positive multiplying weight is used. Also note that the hv_loss parameter is clipped at 0.06, the color parameter is clipped at 0.6, and the si_gain parameter is clipped at 0.004. The si_gain parameter is the only quality *improvement* parameter in the model (since the si_gain parameter is positive, a negative weight results in negative contributions to VQM which produce quality improvements). The si_gain parameter measures improvements to quality that result from edge sharpening or enhancement. Clipping of the parameter at an *upper* threshold of 0.14 immediately before multiplying by the parameter weight prevents excessive improvements to VQM of more than about 1/3 of a quality unit, which is the maximum improvement observed in the general subjective data set (i.e., an HRC will only be rewarded for a little edge enhancement).

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 currently available subjective data.

If $VQM_G > 1.0$, then $VQM_G = (1 + c)*VQM_G / (c + VQM_G)$, where $c = 0.5$.

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

6.4 Developer VQM (VQM_D)

This section provides a full description of the developer model VQM (henceforth denoted as VQM_D), which has the same design constraints as the general model except that an additional constraint requires this model to have an order of magnitude less computational complexity than the general model in section 6.3. To achieve the order of magnitude increase in computational speed, only the Y luminance channel is used, and 18 images are pre-averaged as given in section 4.1.1. The developer model has objective parameters for measuring the perceptual effects of blurring, block distortion, and jerky/unnatural motion. While one objective parameter for measuring noise has been included in this model (ati_gain), the parameter does not fully capture the perceptual effects of added noise. Objective parameters for measuring error blocks and color distortions have not been included in the developer model since they would adversely impact the computational complexity of the model. If noise, error blocks, or color distortions are present, one should use the general model given in section 6.3. Results are presented in section 8.4 that compare VQM_D with mean opinion scores from the same eleven subjective tests that were used for the general model.

VQM_D consists of a linear combination of five parameters. Three parameters are based on features extracted from spatial gradients (section 4.2.2), and two parameters are based on the absolute temporal information feature (section 4.5). VQM_D is given by

VQM_D =

{-0.6289 * avg18F_Y_si13_8x8_std_6_ratio_loss_below5%_mean_clip_0.03

+0.2305 * avg18F_Y_hv13_angle0.225_rmin20_8x8_mean_3_ratio_loss_below5%_10%_square_clip_0.06

+0.1551 * avg18F_Y_hv13_angle0.225_rmin20_8x8_mean_3_log_gain_above95%_mean

+1.0587 * avg18F_Y_ati_8x8_std_1_log_gain_mean_10%

-0.1444 * avg18F_Y_ati_8x8_std_3_ratio_loss_below5%_10%}

The square on the hv_loss parameter is necessary to linearize the parameter response with respect to the subjective data. Note that since the hv_loss parameter becomes positive after the square, a positive multiplying weight is used. Also note that the si_loss parameter is clipped at 0.03 and the hv_loss parameter is clipped at 0.06.

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 currently available subjective data.

If $VQM_D > 1.0$, then $VQM_D = (1 + c)*VQM_D / (c + VQM_D)$, where $c = 0.5$.

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

6.5 Peak-Signal-to-Noise-Ratio VQM (VQM_P)

The fifth model uses the peak-signal-to-noise-ratio (PSNR) measurement that is described in detail in references ([3], [8]). The PSNR-based VQM model (henceforth denoted as VQM_P) uses a form of the logistics function that is recommended in reference [8]. However, the fitting weights are slightly different here from what is given in reference [8] because the logistics fit presented here used all the subjective data described in section 7 while the logistics fit given in reference [8] used only a subset of data sets four and six (see section 7.4). Results are presented in section 8.5 that compare VQM_P with mean opinion scores from the same eleven subjective tests that were used for the general model (VQM_G). VQM_P is given by

$$\mathrm{VQM}_{\mathrm{P}} = \frac{1}{1 + e^{0.1701 * (\mathrm{PSNR} - 25.6675)}}, \quad 10 \le \mathrm{PSNR} \le 55$$

The bounds on PSNR in the above equation come from the range of PSNR that was encountered in the subjective data. The logistics function asymptotically limits VQM_P to always fall between 0 (high quality) and 1 (low quality).

The only deviation from the PSNR description in references ([3], [8]) is a clipping threshold inserted to avoid dividing by zero when operating on binary identical images. This clipping threshold is 130 dB for the video clip's PSNR and 65 dB for an individual image's PSNR.

7. DESCRIPTION OF SUBJECTIVE DATA SETS

The eleven subjective experiments were conducted from 1992 to 1999. All of the data sets were collected in accordance with the most recent version of ITU-R Recommendation BT.500 [12] or ITU-T Recommendation P.910 [16] that was available when the experiment was performed. All of the data sets used scenes from 9 to 10 seconds in duration. Nine of the data sets (i.e., data sets one to nine) used double stimulus testing where viewers saw both the original and processed sequences. Two of the data sets (i.e., data sets ten and eleven) used single stimulus testing where viewers saw only the processed sequence. Seven of the data sets were primarily television experiments (i.e., data sets one to seven) while four of the data sets were primarily videoconferencing experiments (i.e., data sets eight to eleven). For brevity, only a summary of each subjective experiment is given here. The reader is directed to the accompanying references for more complete descriptions of the experiments.

7.1 Data Set One [32]

A panel of 32 viewers rated a total of 42 video clips that were generated by pairing sub-groups of six scenes each (total number of scenes in the test was 12) with seven different MPEG-2 systems. The 12 test scenes included sports material and classical Rec. 601 test scenes. The nine MPEG-2 systems operated at bit rates from 2 Mb/s to 8 Mb/s. Viewers were shown the original and processed sequences in randomized A/B ordering and asked to rate the quality of B using A as a reference. The experiment utilized a seven-point comparison scale (B much worse than A, B worse than A, B slightly worse than A, B the same as A, B slightly better than A, B better than A, B much better than A).

7.2 Data Set Two [11]

A panel of 32 viewers rated the difference in quality between original scenes with controlled amounts of added noise and the resultant MPEG-2 compression-processed output. The data set contains a total of 105 video clips that were generated by pairing seven test scenes at three different noise levels with five MPEG-2 video systems. The seven test scenes were chosen to span a range of spatial detail, motion, brightness, and contrast. The five MPEG-2 video systems operated at bit rates from 1.8 Mb/s to 13.9 Mb/s. The subjective test procedure was the same as data set one.

7.3 Data Set Three [32]

A panel of 32 viewers rated a total of 112 video clips that were generated by pairing sub-groups of eight scenes each (total number of scenes in the test was 16) with 14 different video systems. The 16 test scenes spanned a wide range of spatial detail, motion, brightness, and contrast, and included scene material from movies, sports, nature, and classical Rec. 601 test scenes. The 14 video systems included MPEG-2 systems operated at bit rates from 2 Mb/s to 36 Mb/s with controlled error rates, multi-generation MPEG-2, multi-generation ½-inch professional record/play cycles, VHS, and video teleconferencing systems operating at bit rates from 768 kb/s to 1.5 Mb/s. The subjective test procedure was the same as data set one.

7.4 Data Sets Four to Seven [18]

Pairing ten scenes with nine video systems generated four data sets, each of 90 video clips. These data sets will be referred to as (4) 525-line high quality, (5) 625-line high quality, (6) 525-line low quality, and (7) 625-line low quality. For each data set, a total of 60 to 80 viewers from four different laboratories (i.e., 15 to 20 viewers per laboratory) rated subjective quality using the double stimulus continuous quality scale (DSCS). The twenty different test scenes (ten for 525-line, ten for 625-line) included sports material, classical Rec. 601 test scenes, moving graphics, and stills. The video systems included MPEG-2 systems operating at bit rates from 2 Mb/s to 50 Mb/s, video teleconferencing systems operating at 768 kb/s and 1.5 Mb/s, some systems with digital transmission errors, multi-generation MPEG-2, and multi-generation ½-inch professional record/play cycles, where composite and/or component signal formats were used.

Since the range of subjective quality in the high quality data sets spanned a very small portion of the total subjective scale, the high quality data sets were combined with the corresponding low quality data sets (i.e., data set six was combined with data set seven to produce one 525-line data set while data set eight was combined with data set nine to produce one 625-line data set). This produced a range of quality in each combined data set that was similar to the range of quality present in the other data sets. Thus, results for these combined data sets are more directly comparable to results from the other data sets.

7.5 Data Set Eight ([4], [5])

Viewer panels comprising a total of 30 viewers from three different laboratories rated 600 video clips that were generated by pairing 25 test scenes with 24 video systems. The 25 test scenes included scenes from 5 categories: (1) one person, mainly head and shoulders, (2) one person with graphics and/or more detail, (3) more than one person, (4) graphics with pointing, and (5) high object and/or camera motion. The 24 video systems included proprietary and standardized video teleconferencing systems operating at bit rates from 56 kb/s to 1.5 Mb/s with controlled error rates, one 45 Mb/s codec, and VHS record/play cycle. Viewers were shown the original version first, then the degraded version, and asked to rate the difference in perceived quality using the 5-point impairment scale (imperceptible, perceptible but not annoying, slightly annoying, annoying, very annoying).

7.6 Data Set Nine ([28], [30])

A panel of 48 viewers rated a total of 132 video clips that were generated by random and deterministic pairing of 36 test scenes with 27 video systems. The 36 test scenes contained widely varying amounts of spatial and temporal information. The 27 video systems included digital video compression systems operating at bit-rates from 56 kb/s to 45 Mb/s with controlled error rates, NTSC encode/decode cycles, VHS and S-VHS record/play cycles, and VHF transmission. The subjective test procedure was the same as data set eight.

7.7 Data Set Ten [21]

This data set was a subjective test evaluation of proponent MPEG-4 systems that utilized a panel of 15 expert viewers. We selected a subset of 164 video clips from the main data set. The subset was selected to span the full range of quality and included eight common intermediate format (CIF) resolution test scenes and 41 video systems from the basic compression tests. The eight video scenes included scenes from 2 categories: (1) low spatial detail and low amount of movement, and (2) medium spatial detail and low amount of movement or vice versa. The 41 video systems operated at bit rates from 10 kb/s to 112 kb/s. Viewers were shown only the degraded version and asked to rate the quality on an 11-point numerical scale, with 0 being the worst quality and 10 being the best.

7.8 Data Set Eleven [20]

A panel of 18 viewers rated 48 video clips in a desktop video teleconferencing application. Pairing six scenes with eight different video systems generated the 48 video clips. The six test scenes were selected from ANSI T1.801.01 and were the scenes *5row1*, *filter*, *smity2*, *vtc1nw*, *washdc*, and one scene that included portions of both *vtc2zm* and *vtc2mp*. The eight video systems included seven desktop video teleconferencing systems operating at bit rates from 128 kb/s to 1.5 Mb/s and one NTSC encode/decode cycle. Viewers were shown only the degraded version and asked to rate the quality on the absolute category rating scale (excellent, good, fair, poor, bad).

8. ROOT MEAN SQUARE ERROR (RMSE) ANALYSIS

The following provides an overview of the algorithm used to estimate the root mean square error (RMSE) and objective to subjective correlation of each VQM model. The first step is to determine the parameter weights for each VQM model, and the gain/offset correction factors for each individual subjective data set that will be used to create one large combined subjective data set. The multiple data sets fitting (MDSF) procedure given in Appendix B provides an optimal procedure for the simultaneous computation of VQM model weights and subjective data set scaling factors. In addition to the objective parameter values and subjective mean opinion scores, the following inputs to the MDSF algorithm are required: (1) a per data set cost function $\mathbf{C}_\mathrm{i}$ that specifies how to weight fitting errors associated with each clip in the data set, (2) an error ratio r that specifies how to distribute the total fitting error between subjective score error and objective parameter error, and (3) a reference subjective data set j to which all the other subjective data sets will be scaled. The subjective scores in each data set must be normalized to lie between zero (i.e., the no impairment end of the subjective scale) and one (i.e., the maximum impairment end of the subjective scale) before being used by the MDSF algorithm. Pre-normalization of the objective parameter data may also be desirable if the parameters have widely varying scales or ranges.

The per data set cost function that was used by all stages of the MDSF algorithm was

$\mathbf{C}_i$ = sqrt (view$_i$) for each clip in data set i,

where view$_i$ is the total number of viewers in data set i. Thus, clips that come from data sets with more viewers will have more weight in the fitting procedures used by the MDSF algorithm. This makes sense since the confidence intervals of subjective ratings are proportional to the reciprocal of the square root of the number of viewers. For example, clips in data sets with 60-80 viewers (e.g., data sets four to seven in section 7) will have twice the weight of clips in a data set with 15-20 viewers (e.g., data set eleven in section 7).

An error ratio $r = 1.0$ was used by all stages of the MDSF algorithm. This error ratio assumes that the subjective and objective data have approximately equal amounts of error and thus the MDSF algorithm will distribute the total error equally between the two.

A three-step procedure was used to design the three primary VQM models (i.e., television, videoconferencing, and general purpose). This three-step procedure allows one to obtain optimal fitting on the television and videoconferencing models (whose data do not overlap), while accepting some performance degradation on the general-purpose model. First, the MDSF algorithm was performed on the television data (data sets one to seven in section 7), where data set three was used as the reference. Data set three was chosen as the reference for the television data since it has the widest range of quality. Next, the MDSF algorithm was performed on the videoconferencing data (data sets eight to eleven in section 7), where data set eight was used as the reference (this videoconferencing data set has the widest range of quality). The television and videoconferencing data sets output from steps 1 and 2 were then input into the MDSF algorithm again to produce the general model. For this final step, the videoconferencing data was used as the reference since it had the widest range of quality. These final subjective and objective data sets produced by the MDSF algorithm were then linearly scaled such that the average of the lower twenty and upper twenty objective samples produced zero and one, respectively. This procedure for defining the VQM scale is more robust than using just a single sample (i.e., best quality observed and worst quality observed).

The developer model was computed using the MDSF algorithm called with the final scaled subjective data (now one large combined subjective data set) and the developer's objective parameters, using itself as a reference. Lastly, the PSNR model was obtained by performing a least squares fit using the two free parameter logistics relationship shown in section 6.5 between PSNR and the final scaled subjective data.

The RMSE and correlation results between each of the five VQM models (television, videoconferencing, general, developer, and PSNR) and the subjective data are given in sections 8.1 to 8.5, respectively.

8.1 Television Model Error

The television model used the first seven data sets described in section 7. Figure 30 shows the scatter plot of clip subjective quality versus clip VQM_T, where each data set is plotted in a different color (1 = black, 2 = red, 3 = green, 4 = blue, 5 = yellow, 6 = magenta, 7 = cyan). The overall Pearson linear correlation coefficient between clip subjective quality and clip VQM_T for the data points in Figure 30 is 0.918 and the RMSE is 0.061.

Figure 31 shows the effect of averaging over scenes to produce a single subjective score (i.e., HRC subjective quality) and objective score (i.e., HRC VQM_T) for each video system. HRC subjective quality is indicative of how the system responds (on average) to a set of video scenes. The overall Pearson linear correlation coefficient between HRC subjective quality and HRC VQM_T for the data points in Figure 31 is 0.981 and the RMSE is 0.025. For making video system (i.e., HRC) comparisons, the estimate of HRC subjective quality provided by HRC VQM_T is more accurate than the estimate of clip subjective quality provided by clip VQM_T. This can be seen by comparing the amount of scatter in Figure 31 with the amount of scatter in Figure 30 or by comparing the two RMSEs.

The per data set Pearson linear correlation coefficients between clip subjective quality and clip VQM_T are given in the first column of Table 7. The second column of Table 7 gives the individual data set correlation coefficients between HRC subjective quality and HRC VQM_T.

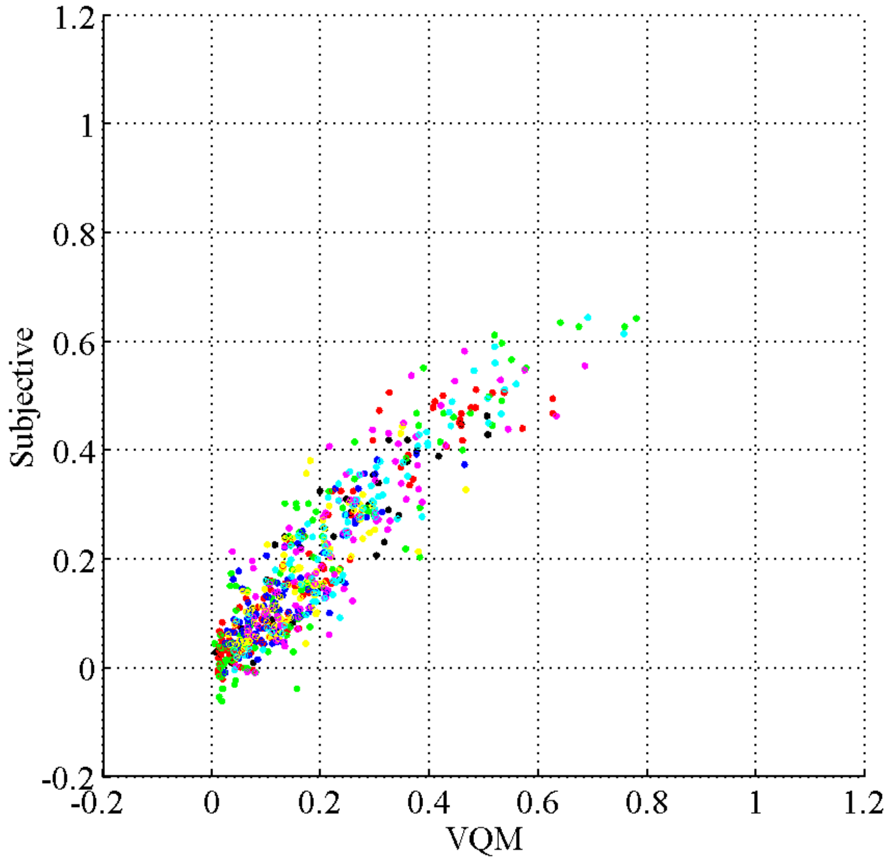

Figure 30. Clip subjective quality vs. clip VQM_T.

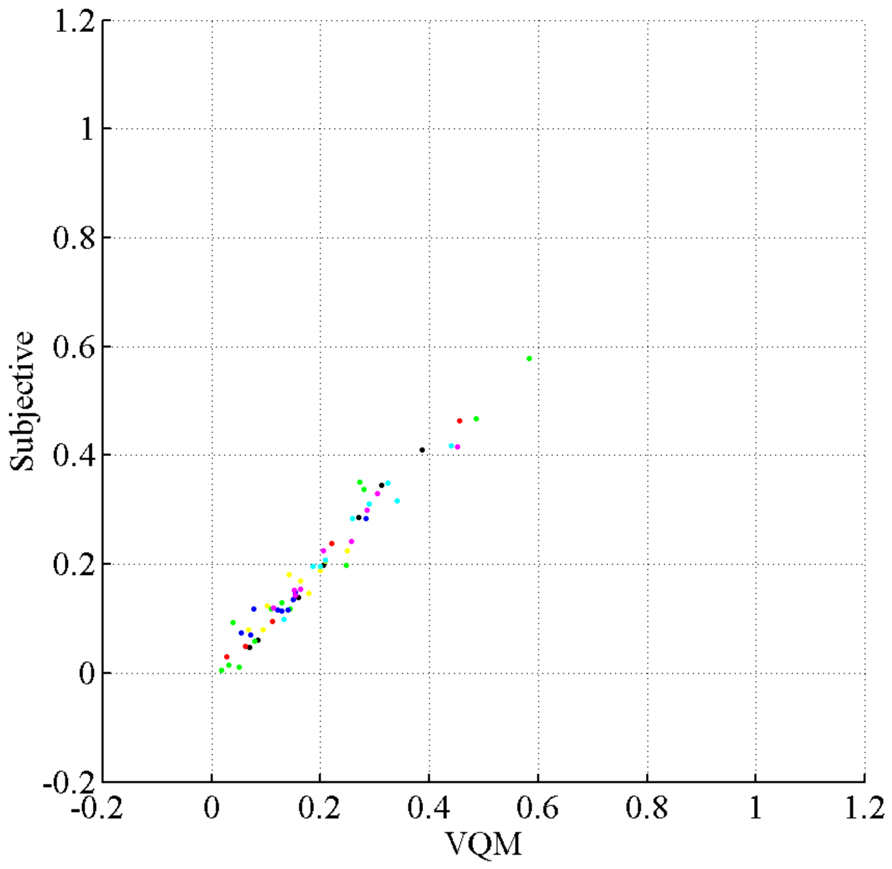

Figure 31. HRC subjective quality vs. HRC VQM$_T$.

Table 7. Pearson Linear Correlation Coefficients Between Subjective Data and VQM$_T$

Data Set Number (See Section 7)	**Pearson Linear Correlation Coefficients**	
	Clip VQM$_T$ (Figure 30)	**HRC VQM$_T$ (Figure 31)**
One (black)	0.916	0.998
Two (red)	0.950	0.998
Three (green)	0.927	0.978
Four (blue)	0.848	0.958
Five (yellow)	0.798	0.928
Six (magenta)	0.869	0.984
Seven (cyan)	0.928	0.973

8.2 Videoconferencing Model Error

The videoconferencing model used data sets eight to eleven described in section 7. Figure 32 shows the scatter plot of clip subjective quality versus clip VQM_V, where each data set is plotted in a different color (8 = gray, 9 = dark red, 10 = copper, 11 = aquamarine). The overall Pearson linear correlation coefficient between clip subjective quality and clip VQM_V for the data points in Figure 32 is 0.929 and the RMSE is 0.095.

Figure 33 shows the effect of averaging over scenes to produce a single subjective score (i.e., HRC subjective quality) and objective score (i.e., HRC VQM_T) for each video system. HRC subjective quality is indicative of how the system responds (on average) to a set of video scenes. The overall Pearson linear correlation coefficient between HRC subjective quality and HRC VQM_V for the data points in Figure 33 is 0.969 and the RMSE is 0.053. For making video system (i.e., HRC) comparisons, the estimate of HRC subjective quality provided by HRC VQM_V is more accurate than the estimate of clip subjective quality provided by clip VQM_V. This can be seen by comparing the amount of scatter in Figure 33 with the amount of scatter in Figure 32, or by comparing the two RMSEs.

The per data set Pearson linear correlation coefficients between clip subjective quality and clip VQM_V are given in the first column of Table 8. The second column of Table 8 gives the individual data set correlation coefficients between HRC subjective quality and HRC VQM_V.

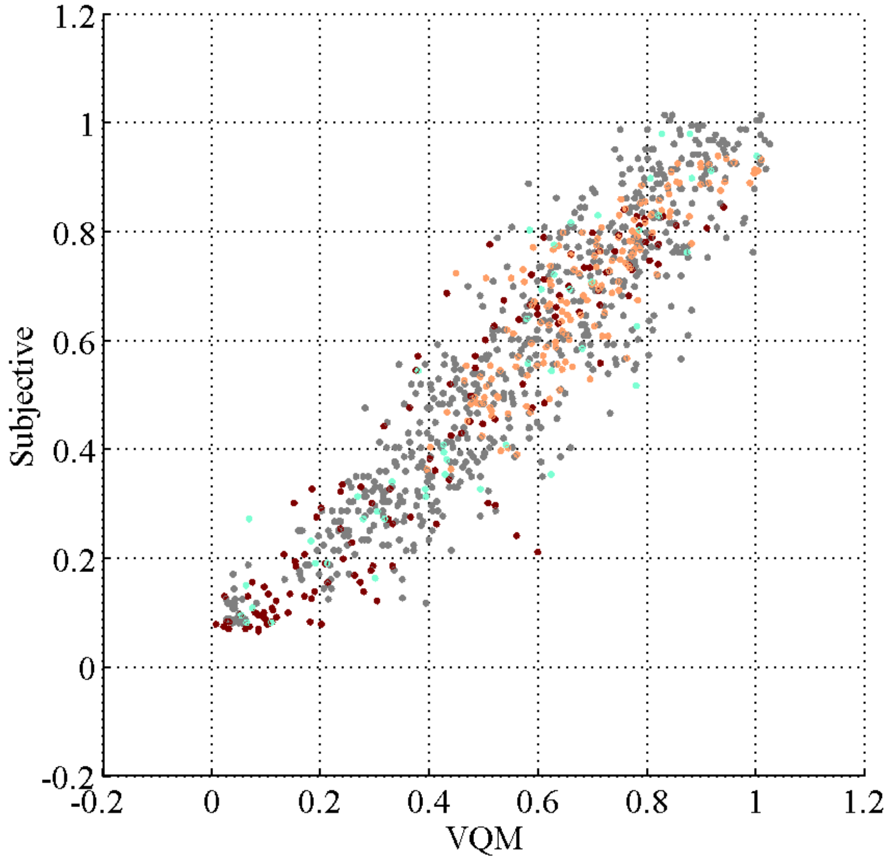

Figure 32. Clip subjective quality vs. clip VQM_V.

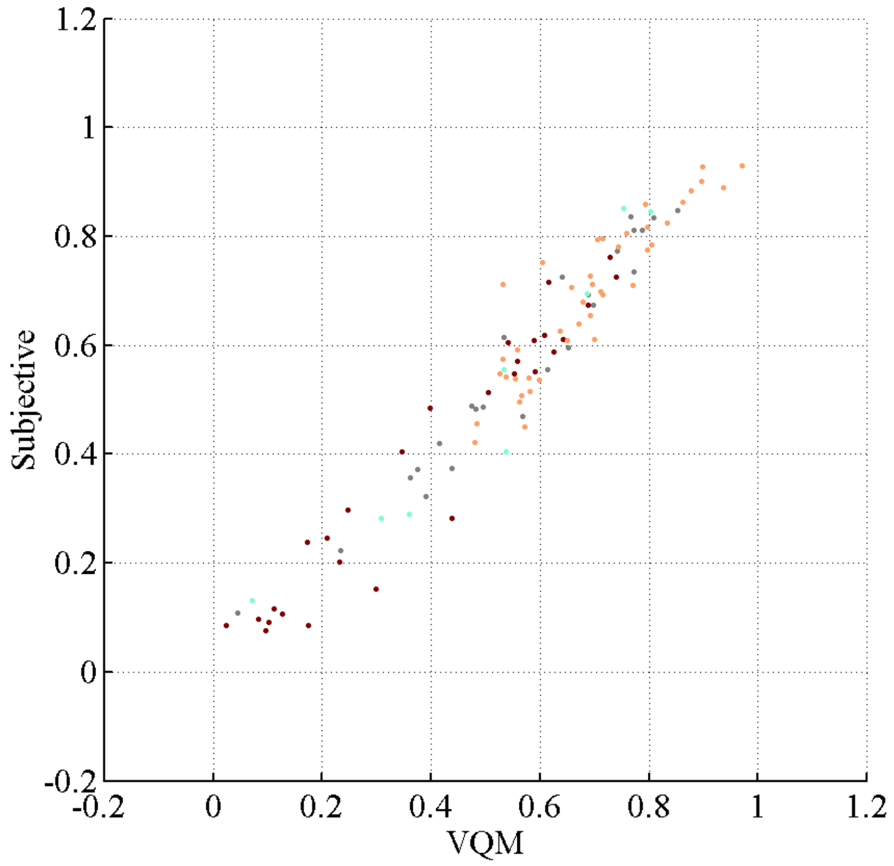

Figure 33. HRC subjective quality vs. HRC VQM$_V$.

Table 8. Pearson Linear Correlation Coefficients Between Subjective Data and VQM$_V$

Data Set Number (See Section 7)	**Pearson Linear Correlation Coefficients**	
	Clip VQM$_V$ (Figure 32)	**HRC VQM$_V$ (Figure 33)**
Eight (gray)	0.922	0.974
Nine (dark red)	0.933	0.968
Ten (copper)	0.869	0.910
Eleven (aquamarine)	0.923	0.964

8.3 General Model Error

The general model used all eleven data sets described in section 7. Figure 34 shows the scatter plot of clip subjective quality versus clip VQM_G, where each data set is plotted in a different color (1 = black, 2 = red, 3 = green, 4 = blue, 5 = yellow, 6 = magenta, 7 = cyan, 8 = gray, 9 = dark red, 10 = copper, 11 = aquamarine). The overall Pearson linear correlation coefficient between clip subjective quality and clip VQM_G for the data points in Figure 34 is 0.948 and the RMSE is 0.091.

Figure 35 shows the effect of averaging over scenes to produce a single subjective score (i.e., HRC subjective quality) and objective score (i.e., HRC VQM_G) for each video system. HRC subjective quality is indicative of how the system responds (on average) to a set of video scenes. The overall Pearson linear correlation coefficient between HRC subjective quality and HRC VQM_G for the data points in Figure 35 is 0.980 and the RMSE is 0.054. For making video system (i.e., HRC) comparisons, the estimate of HRC subjective quality provided by HRC VQM_G is more accurate than the estimate of clip subjective quality provided by clip VQM_G. This can be seen by comparing the amount of scatter in Figure 35 with the amount of scatter in Figure 34, or by comparing the two RMSEs.

The per data set Pearson linear correlation coefficients between clip subjective quality and clip VQM_G are given in the first column of Table 9. The second column of Table 9 gives the individual data set correlation coefficients between HRC subjective quality and HRC VQM_G.

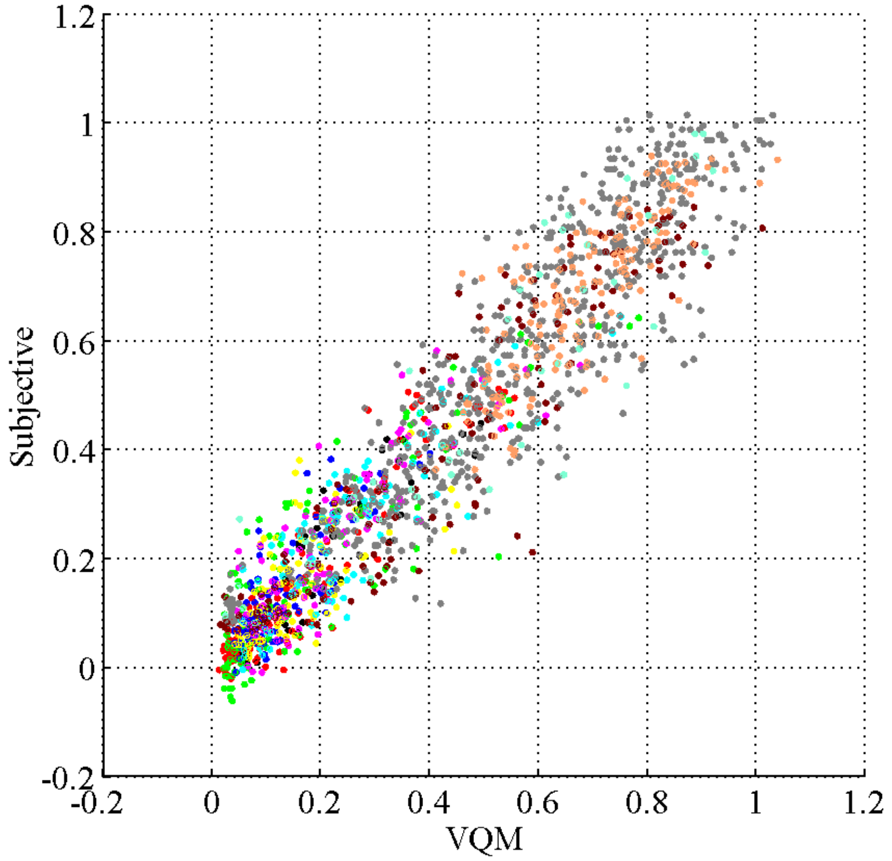

Figure 34. Clip subjective quality vs. clip VQM_G.

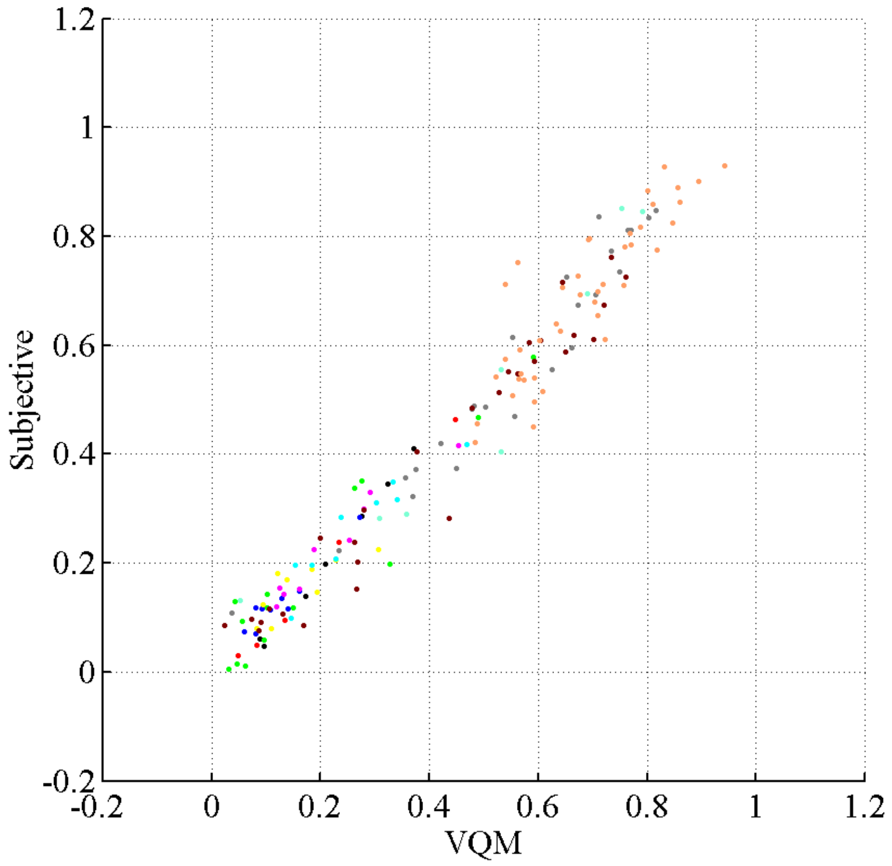

Figure 35. HRC subjective quality vs. HRC VQM_G.

Table 9. Pearson Linear Correlation Coefficients Between Subjective Data and VQM_G

Data Set Number (See Section 7)	**Pearson Linear Correlation Coefficients**	
	Clip VQM_G (Figure 34)	**HRC VQM_G (Figure 35)**
One (black)	0.904	0.998
Two (red)	0.944	0.997
Three (green)	0.887	0.944
Four (blue)	0.817	0.955
Five (yellow)	0.676	0.808

Six (magenta)	0.872	0.974
Seven (cyan)	0.877	0.941
Eight (gray)	0.910	0.970
Nine (dark red)	0.933	0.977
Ten (copper)	0.830	0.880
Eleven (aquamarine)	0.915	0.961

8.4 Developer Model Error

The developer model used all eleven data sets described in section 7. Figure 36 shows the scatter plot of clip subjective quality versus clip VQM_D, where each data set is plotted in a different color (1 = black, 2 = red, 3 = green, 4 = blue, 5 = yellow, 6 = magenta, 7 = cyan, 8 = gray, 9 = dark red, 10 = copper, 11 = aquamarine). The overall Pearson linear correlation coefficient between clip subjective quality and clip VQM_D for the data points in Figure 36 is 0.940 and the RMSE is 0.097.

Figure 37 shows the effect of averaging over scenes to produce a single subjective score (i.e., HRC subjective quality) and objective score (i.e., HRC VQM_D) for each video system. HRC subjective quality is indicative of how the system responds (on average) to a set of video scenes. The overall Pearson linear correlation coefficient between HRC subjective quality and HRC VQM_D for the data points in Figure 37 is 0.973 and the RMSE is 0.062. For making video system (i.e., HRC) comparisons, the estimate of HRC subjective quality provided by HRC VQM_D is more accurate than the estimate of clip subjective quality provided by clip VQM_D. This can be seen by comparing the amount of scatter in Figure 37 with the amount of scatter in Figure 36, or by comparing the two RMSEs.

The per data set Pearson linear correlation coefficients between clip subjective quality and clip VQM_D are given in the first column of Table 10. The second column of Table 10 gives the individual data set correlation coefficients between HRC subjective quality and HRC VQM_D.

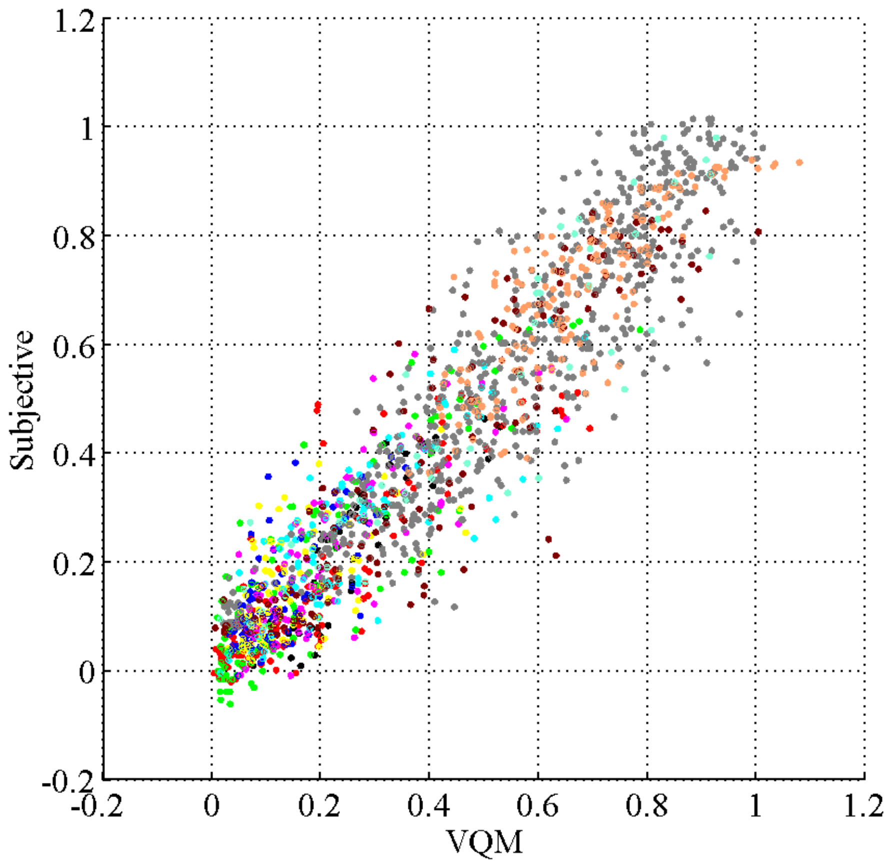

Figure 36. Clip subjective quality vs. clip VQM_D.

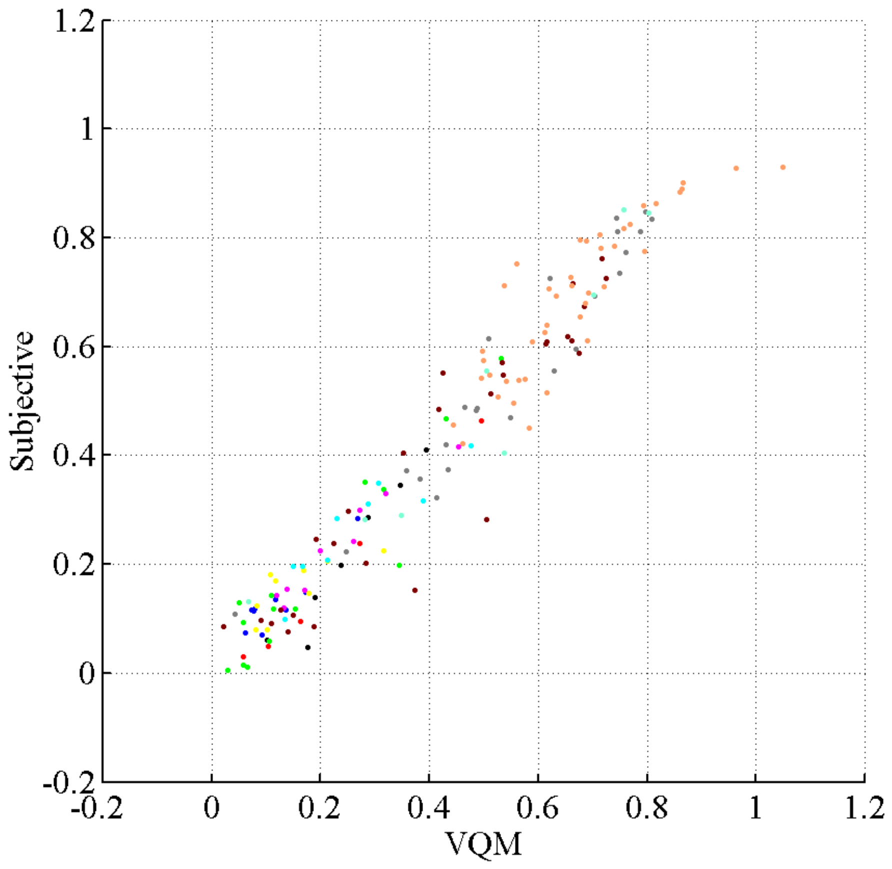

Figure 37. HRC subjective quality vs. HRC VQM$_D$.

Table 10. Pearson Linear Correlation Coefficients Between Subjective Data and VQM$_D$

Data Set Number (See Section 7)	**Pearson Linear Correlation Coefficients**	
	Clip VQM$_D$ (Figure 36)	**HRC VQM$_D$ (Figure 37)**
One (black)	0.887	0.967
Two (red)	0.881	0.996
Three (green)	0.881	0.942
Four (blue)	0.693	0.914
Five (yellow)	0.624	0.770

Six (magenta)	0.824	0.977
Seven (cyan)	0.820	0.912
Eight (gray)	0.908	0.964
Nine (dark red)	0.909	0.945
Ten (copper)	0.832	0.879
Eleven (aquamarine)	0.923	0.963

8.5 PSNR Model Error

The PSNR model used all eleven data sets described in section 7. Figure 38 shows the scatter plot of clip subjective quality versus clip VQM_P, where each data set is plotted in a different color (1 = black, 2 = red, 3 = green, 4 = blue, 5 = yellow, 6 = magenta, 7 = cyan, 8 = gray, 9 = dark red, 10 = copper, 11 = aquamarine). The overall Pearson linear correlation coefficient between clip subjective quality and clip VQM_P for the data points in Figure 38 is 0.781 and the RMSE is 0.178.

Figure 39 shows the effect of averaging over scenes to produce a single subjective score (i.e., HRC subjective quality) and objective score (i.e., HRC VQM_P) for each video system. HRC subjective quality is indicative of how the system responds (on average) to a set of video scenes. The overall Pearson linear correlation coefficient between HRC subjective quality and HRC VQM_P for the data points in Figure 39 is 0.895 and the RMSE is 0.139. Unlike the previous 4 models (sections 8.1 to 8.4), the HRC VQM_P plot shown in Figure 39 exhibits some noticeable systematic errors that could be removed with further least squares fitting. This observation suggests that it might be more advantageous in some applications to use HRC quality data to deduce the logistic fit given in section 6.5, rather than clip quality data. If this is done, the RMSE error for clip VQM_P increases somewhat (about 7%), but the RMSE error for HRC VQM_P decreases by a greater amount (about 15%). In any case, for making video system (i.e., HRC) comparisons, the estimate of HRC subjective quality provided by HRC VQM_P is more accurate than the estimate of clip subjective quality provided by clip VQM_P. This can be seen by comparing the amount of scatter in Figure 39 with the amount of scatter in Figure 38 or by comparing the two RMSEs.

The per data set Pearson linear correlation coefficients between clip subjective quality and clip VQM_P are given in the first column of Table 11. The second column of Table 11 gives the individual data set correlation coefficients between HRC subjective quality and HRC VQM_P.

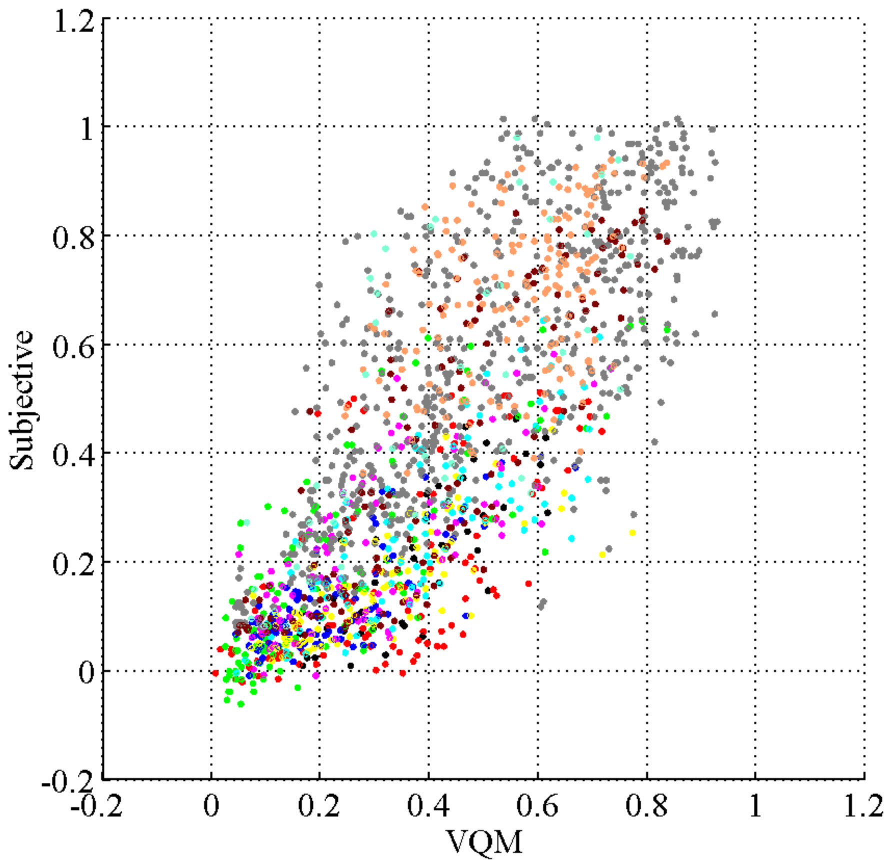

Figure 38. Clip subjective quality vs. clip VQM_P.

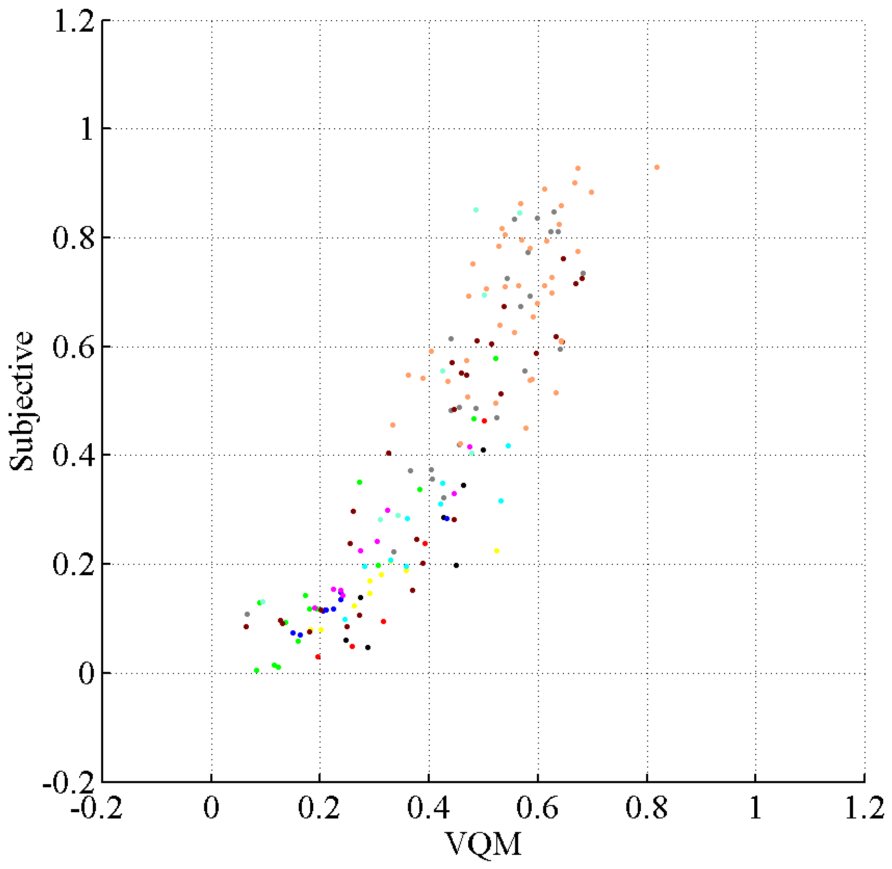

Figure 39. HRC subjective quality vs. HRC VQM$_P$.

Table 11. Pearson Linear Correlation Coefficients Between Subjective Data and VQM$_P$

Data Set Number (See Section 7)	**Pearson Linear Correlation Coefficients**	
	Clip VQM$_P$ (Figure 38)	**HRC VQM$_P$ (Figure 39)**
One (black)	0.778	0.912
Two (red)	0.670	0.965
Three (green)	0.800	0.948
Four (blue)	0.790	0.994
Five (yellow)	0.760	0.886

Six (magenta)	0.722	0.963
Seven (cyan)	0.774	0.899
Eight (gray)	0.732	0.870
Nine (dark red)	0.848	0.914
Ten (copper)	0.446	0.611
Eleven (aquamarine)	0.719	0.860

9. ROOT CAUSE ANALYSIS (RCA)

A final VQM score that tracks the perceived quality of the video stream is a very useful quantity. However, sometimes more detailed information is desired by system designers and operators so they can determine how to fix or improve upon the video system being tested. The goal of root cause analysis (RCA) is to provide more detailed information than a single quality score can provide. Two types of RCA will be considered in this section. The first type of RCA, called calibration RCA (section 9.1), is derived from results generated by the calibration routines in section 3. The second type of RCA, called impairment RCA (section 9.2), provides the user with specifics of the nature of the perceived impairments. For example, was the impairment due to block distortion, blurring, jerky/unnatural motion (i.e., dropped frames), added noise, or some other type of distortion? Impairment RCA is derived from the objective quality parameters described in section 5. A new kind of RCA subjective test, described in Appendix C, is used together with these objective quality parameters to build predictors of perceived impairment types.

9.1 Calibration Root Cause Analysis (RCA)

Calibration Root Cause Analysis (RCA) is obtained by examining the calibration data (both final and intermediate) for potential problems. Section 9.1.1 of this document describes the calibration RCA that can be determined by examining the final calibration results. Section 9.1.2 describes the calibration RCA that can be determined by examining the intermediate calibration results (intermediate results differ from final results in that they are much more detailed in nature and are not normally provided to the user unless requested). Section 9.1.3 describes the severity of the problems detected by the calibration RCA.

9.1.1 Calibration RCA from Final Results

9.1.1.1 Whole Image Original Valid Region (OVR)

If the OVR includes the entire image, and the image is larger than 480 lines by 640 pixels,

Print, “Warning for clip”, clip_name

“Original valid region set to the entire image. Pixels near the edge of the image may not contain valid picture information. Automatic calculation of original valid region recommended.”

9.1.1.2 Small Processed Valid Region (PVR)

If the automatically calculated PVR has thrown away more than 15% of the processed horizontal pixels or 15% of the processed vertical lines,

> Print, “Small automatically calculated processed valid region”, PVR

9.1.1.3 Large Horizontal Shift

If the processed horizontal shift is less than –5 pixels or greater than +5 pixels,

> Print, “Large processed video horizontal shift”, horizontal_shift

9.1.1.4 Non-Zero Vertical Shift

If interlace video is being measured, and the vertical shifts of NTSC field one and field two do not indicate either framed or reframed video (see section 3.1.2),

> Print, “Unacceptable difference between field one and field two vertical shifts. Likely causes: erroneous manual entry, or if video appears to move up and down at the frame rate, then the NTSC field time ordering is inverted. The HRC or video system under test should be corrected before conducting any further analysis.”

If interlace video is being measured, and the vertical shifts of NTSC field one and field two indicate reframing,

> Print, “Interlaced images reframed: Field one becomes field two of the current frame, and field two becomes field one of the next frame. NTSC field time ordering is preserved.”

If progressive video is being measured, and the vertical shift is anything but zero,

> Print, “Non-zero processed vertical shift of ”, vertical shift, “frame lines.”

If interlace video is being measured, and the vertical shifts of NTSC field one and field two are equal, and the processed vertical shift is anything but zero on field one,

> Print, “Non-zero processed field one and field two vertical shift of ”, f1_vertical shift, “ field lines.”

If interlace video is being measured, and reframing is detected, and NTSC field one or field two does not have a vertical shift of zero,

> Print, “Non-zero processed vertical shift. Field one vertical shift of ”, f1_vertical shift, “field lines, and field two vertical shift of”, f2_vertical_shift, “field lines.”

9.1.1.5 Large Gain Error

If the luminance (Y) channel gain is less than 0.90 or greater than 1.1,

> Print, “Large Y gain error”, Y_gain

If the blue chrominance (C_B) channel gain is less than 0.90 or greater than 1.1,

> Print, “Large C_B gain error”, C_B_gain

If the red chrominance (C_R) channel gain is less than 0.90 or greater than 1.1,

> Print, “Large C_R gain error”, C_R_gain

9.1.1.6 Large Offset Error

If the Y channel offset is less than –10 or greater than +10,

Print, “Large Y offset error”, Y_offset

If the C_B channel offset is less than –10 or greater than +10,

Print, “Large C_B offset error”, C_B_offset

If the C_R channel offset is less than –10 or greater than +10,

Print, “Large C_R offset error”, C_R_offset

If any large gain error or offset error statement is printed,

Print, “Gain is the amplitude scaling that the HRC performs on the video scenes. Gain is related to contrast. A gain of 1.0 is ideal (i.e., no amplitude scaling). A gain of 1.25 means that each pixel's amplitude has been multiplied by 1.25. Offset is the amplitude shift that the HRC performs on the video scenes. Offset is related to brightness. An offset of 0 is ideal (i.e., no amplitude shift). An offset of 20 means that 20 has been added to all of the pixels.”

Print, “Formula: processed = gain * original + offset.”

9.1.1.7 Small Temporal Valid Region (TVR)

If the temporal registration routine reduces the TVR of any scene by more than 15%, then for each scene

Print, “The temporal valid region of processed scene ”, scene_name, “ was reduced by ”, tvr_reduction, “ seconds.” [19]

9.1.1.8 Different Temporal Registrations

If different scenes have different temporal registrations,

Print, “Scenes have different temporal registrations. Maximum temporal registration difference = ”, max_temporal_registration_difference, “ seconds.” [20]

Print, "Possible causes: variable video delay that is scene dependent, variations in frame grab start times (e.g., when the scenes were frame grabbed independently of each other), temporal registration inaccuracies due to highly distorted video (e.g., many dropped frames), video with very small amounts of motion, or video with repetitive motion.”

9.1.2 Calibration RCA from Intermediate Results

9.1.2.1 Different Spatial Registrations When Computed from Scenes (Section 3.1.5)

The following calibration RCA is performed for a given HRC on the list of spatial registrations for each scene of that HRC. This same logical flow could also be applied to an individual scene by comparing the results from individual frames from that scene.

[19] Note that the TVR reduction is specified in seconds to allow consistent reporting for 15 frames per second (fps) and 30 fps video systems.

[20] Note that the maximum temporal shift difference between two scenes is specified in seconds to allow consistent reporting for 15 fps and 30 fps video systems.

If at least 75% of the vertical and horizontal shifts exactly match the final spatial registration, then spatial registration is excellent.

Else, if at least 75% of vertical registrations exactly match the final vertical spatial registration, and at least 75% of horizontal shifts match the final horizontal shift within plus or minus one pixel,

> Print, "Good spatial registration consistency, with slight horizontal shift ambiguity."

Else, if at least 75% of horizontal shifts match the final horizontal shift within plus or minus one pixel, and at least 75% of vertical registrations match the final vertical registration or one adjacent vertical registration (e.g., either +1 line or –1 line but not both),[21]

> Print, "Fair spatial registration consistency, with slight ambiguities."

Else, if at least 50% of horizontal shifts match the final horizontal shift within plus or minus one pixel, and at least 50% of vertical registrations match the final vertical registration or one adjacent vertical registration (e.g., either +1 line or –1 line but not both),

> Print, "Poor spatial registration consistency, with significant ambiguities. Spatial registration algorithm may be encountering temporal registration problems. Spatial registration probably accurate but may be erroneous enough to cause measurement errors."

Else,

> Print, "Unacceptably inconsistent spatial registration; actual spatial registration extremely ambiguous. Spatial registration algorithm may be encountering temporal registration problems. Processed video may have spatial scaling issues. Recommend color bars be used for spatial registration or another alternate method for spatial registration. Quality prediction errors may result from current spatial registration."

9.1.2.2 Different Gains, When Computed from Scenes (Section 3.3.3)

If different scenes have Y gains that differ by more than 0.1,

> Print, "Scenes have different Y gains ranging from", minimum_Y_gain, "to", maximum_Y_gain.

If different scenes have C_B gains that differ by more than 0.1,

> Print, "Scenes have different C_B gains ranging from", minimum_C_B_gain, "to", maximum_C_B_gain.

If different scenes have C_R gains that differ by more than 0.1,

> Print, "Scenes have different C_R gains ranging from", minimum_C_R_gain, "to", maximum_C_R_gain.

9.1.2.3 Different Offsets, When Computed from Scenes (Section 3.3.3)

If different scenes have Y offsets that differ by more than 10,

> Print, "Scenes have different Y offsets, ranging from", minimum_Y_offset, "to", maximum_Y_offset.

If different scenes have C_B offsets that differ by more than 10,

[21] For progressive video, this means either +1 frame line or –1 frame line, but not both. For interlaced video, this means either +1 field line or –1 field line, but not both, with the added constraint that the vertical spatial registrations of NTSC field one and field two must be in the same direction to be counted.

Print, “Scenes have different C_B offsets, ranging from”, minimum_C_B_offset, “to”, maximum_C_B_offset.

If different scenes have C_R offsets that differ by more than 10,

Print, “Scenes have different C_R offsets, ranging from”, minimum_C_R_offset, “to”, maximum_C_R_offset.

If any gain or offset by scenes warning was printed,

Print, “Most likely, HRC gains & offsets are changing due to highly distorted scene content. This may reduce the accuracy of the overall estimate of HRC gain & offset. Alternatively, the scenes may be from different HRCs. Please check to make sure that all scenes come from the same HRC.”

9.1.2.4 Original Valid Region When Computed Automatically (Section 3.2.2.1)

If the original valid region appears to be empty,

Print, “WARNING: original scene”, scene_name, “does not contain any scene content. Valid region set to the entire image.”

If the original valid region contains less than 10% of the image horizontally or vertically, then print:

Print, “WARNING: original scene”, scene_name, “picture content less than 10% of the image. Valid region set to the entire image.”

9.1.2.5 Processed Valid Region When Computed from Scenes (Section 3.2.2.2)

If the processed valid region appears to be empty,

Print, “WARNING: processed scene ”, scene_name, “ does not contain any scene content. Valid region set to the original valid region.”

If the processed valid region contains less than 10% of the image horizontally or vertically,

Print, “WARNING: processed scene ”, scene_name, “ picture content less than 10% of the image. Valid region set to the original valid region.”

9.1.3 Severity of Calibration Problems

All calibration RCA analyses as a whole are classified as either “warning” or “error,” where the error classification is more severe than the warning classification. If any of the following conditions are true, then the error classification is used. When errors are encountered, the user should determine the cause of the error and correct it before proceeding with VQM measurements.

- Y, Cb, or Cr gain less than 0.6 or greater than 1.4.
- Y, Cb, or Cr offset less than –40 or greater than 40.
- Unacceptably inconsistent spatial registration (see section 9.1.2.1).
- Unacceptable differences between NTSC field one and field two vertical shifts (see section 9.1.1.4).
- Horizontal shift less than –20 or greater than 20.
- Vertical shift less than –24 frame lines or greater than 24 frame lines.

9.2 Impairment Root Cause Analysis (RCA)

Impairment RCA covers automated methods for estimating the types of artifacts and impairments that are contributing to the VQM score. These artifact types are defined in [2] and include such items as blurring, block distortion (i.e., tiling or error blocks), jerky/unnatural motion (i.e., dropped frames), and added noise. Impairment RCA can provide the system designer and operator with more information than an overall quality score. Thus, impairment RCA may provide useful indicators as to *why* the video system is producing the given quality level.

We would like impairment RCA to be available for any quality level. For instance, if block distortion is the primary contributor to the perceived impairments, we would like to know this, regardless of whether the VQM score is 0.1 (good quality) or 0.9 (poor quality). For this reason, a special impairment RCA subjective experiment was designed, the details of which are described in Appendix C. In this experiment, viewers were asked to determine if certain artifacts were present in the video without regard to the perceived quality level. This subjective data was then used to develop impairment RCA estimators for the VQM models given in section 6. The impairment RCA estimators utilized the same basic objective parameters as those found in the various VQM models, except that different spatial-temporal collapsing functions (see sections 5.3 and 5.4) were allowed. This was necessary due to the nature of the impairment RCA subjective data, which is more binary-like (i.e., artifact is perceived or not perceived) than the VQM scores themselves. Thus, spatial-temporal collapsing functions were chosen for artifact detection rather than quality level estimation. The inclusion of impairment RCA estimators adds very little to the overall computational requirements since most computations occur prior to spatial-temporal collapsing.

We developed impairment RCA estimators for only a selected number of artifact types for the four VQM models given in sections 6.1 to 6.4. The impairment RCA estimators presented here, as well as the procedures used for conducting impairment RCA subjective tests and developing impairment RCA estimators, are intended to be preliminary and are provided as a proof of concept that can be built upon and refined.[22]

All impairment RCA estimators return a number between 0% and 100%, where 100% means that the artifact was perceived as being the primary artifact by all the viewers, 50% means that the artifact was perceived as being a secondary artifact, and 0% means the artifact was not perceived.

9.2.1 Impairment RCA for Television Model

This section presents impairment RCA estimators for the television model that is described in section 6.1.

$$
\begin{aligned}
\text{BLURRING}_t = \\
&\{100 * [-0.030 \\
&\quad -1.374 * \text{Y\_contrast\_16x16\_2F\_std\_6\_ratio\_loss\_below2\%\_mean} \\
&\quad -15.166 * \text{Y\_contrast\_16x16\_2F\_std\_4\_log\_gain\_std\_above90\%\_square}]\}\Big|_{0}^{100}
\end{aligned}
$$

[22] Due to the preliminary nature of the impairment RCA models, we have not included objective to subjective plots or correlation coefficients.

The notation $\Big|_{0}^{100}$ means to clip the overall estimate at 0% and 100% (i.e., if the estimate goes below 0% clip at 0%, and if the estimate goes above 100% clip at 100%).

JERKY_MOTION$_t$ =

{100 * [-0.018

+3.670 * Y_si13_8x8_6F_std_12_ratio_loss_below10%_std

+2.434 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_log_gain_mean_75%_clip_0.075]} $\Big|_{0}^{100}$

GLOBAL_NOISE$_t$ =

{100 * [20.392 * Y_contrast_16x16_2F_std_4_log_gain_50%_10%]} $\Big|_{0}^{100}$

BLOCK_DISTORTION$_t$ =

{100 * [-0.080

+1.074 * Y_si13_8x8_6F_std_8_log_gain_above99.5%_std_sqrt

+1.144 * Y_hv13_angle0.225_rmin20_8x8_6F_mean_3_log_gain_above_95%_25%_sqrt_clip_0.45

+1.405 * Y_cont_16x16_2F_std_6_ratio_loss_below_10%_25%_clip_0.2

+0.033 * color_coher_color_8x8_1F_mean_euclid_above99%tail_std]} $\Big|_{0}^{100}$

ERROR_BLOCKS$_t$ =

{100 * [-0.028

+0.321 * Y_cont_16x16_2F_std_4_log_gain_above99%tail_above90%tail_sqrt

+0.065 * color_coher_color_8x8_1F_mean_euclid_above99%tail_std]} $\Big|_{0}^{100}$

9.2.2 Impairment RCA for Videoconferencing Model

This section presents impairment RCA estimators for the videoconferencing model that is described in section 6.2.

BLURRING_v =

$$\begin{aligned}&\{100 * [-1.453 * \text{Y\_si13\_8x8\_6F\_std\_12\_ratio\_loss\_below5\%\_mean}\\&\qquad \left.-24.578 * \text{Y\_si13\_8x8\_6F\_std\_8\_log\_gain\_mean\_std}]\}\right|_{0}^{100}\end{aligned}$$

JERKY_MOTION_v =

$$\begin{aligned}&\{100 * [-0.018\\&\qquad +3.670 * \text{Y\_si13\_8x8\_6F\_std\_12\_ratio\_loss\_below10\%\_std}\\&\qquad \left.+2.434 * \text{Y\_hv13\_angle0.225\_rmin20\_8x8\_6F\_mean\_3\_log\_gain\_mean\_75\%\_clip\_0.075}]\}\right|_{0}^{100}\end{aligned}$$

BLOCK_DISTORTIONv =

$$\begin{aligned}&\{100 * [-0.077\\&\qquad +1.252 * \text{Y\_si13\_8x8\_6F\_std\_8\_log\_gain\_above99.5\%\_std\_sqrt}\\&\qquad +0.662 * \text{Y\_hv13\_angle0.225\_rmin20\_8x8\_6F\_mean\_3\_log\_gain\_above\_90\%\_25\%\_sqrt\_clip\_0.32}\\&\qquad \left.+1.860 * \text{avg6F\_Y\_ati\_32x32\_std\_3\_ratio\_loss\_mean\_mean}]\}\right|_{0}^{100}\end{aligned}$$

9.2.3 Impairment RCA for General Model

This section presents impairment RCA estimators for the general model that is described in section 6.3.

BLURRING_g =

$$\begin{aligned}&\{100 * [-1.349 * \text{Y\_si13\_8x8\_6F\_std\_12\_ratio\_loss\_below5\%\_mean}\\&\qquad \left.-0.033 * \text{Y\_contrast\_ati\_4x4\_6F\_std\_3\_ratio\_gain\_above95\%\_mean}]\}\right|_{0}^{100}\end{aligned}$$

JERKY_MOTION_g =

$$\left.\{100 * [1.334 * \text{Y\_contrast\_ati\_4x4\_6F\_std\_3\_log\_loss\_below5\%\_std}]\}\right|_{0}^{100}$$

GLOBAL_NOISE_g =

$$\left.\{100 * [0.438 * \text{Y\_contrast\_ati\_4x4\_6F\_std\_3\_ratio\_gain\_mean\_10\%}]\}\right|_{0}^{100}$$

$$\text{BLOCK\_DISTORTION}_g =$$

$$\begin{aligned}
&\{100 * [-0.093 \\
&\quad +1.134 * \text{Y\_si13\_8x8\_6F\_std\_8\_log\_gain\_above99.5\%\_std\_sqrt} \\
&\quad +0.849 * \text{Y\_hv13\_angle0.225\_rmin20\_8x8\_6F\_mean\_3\_log\_gain\_above\_95\%\_25\%\_sqrt\_clip\_0.45} \\
&\quad -1.552 * \text{Y\_contrast\_ati\_4x4\_6F\_std\_3\_log\_loss\_std\_mean\_clip\_0.09} \\
&\quad +0.034 * \text{color\_coher\_color\_8x8\_1F\_mean\_euclid\_above\_99\%tail\_std}]\}\Big|_{0}^{100}
\end{aligned}$$

9.2.4 Impairment RCA for Developer Model

This section presents impairment RCA estimators for the developer model that is described in section 6.4.

$$\text{BLURRING}_d =$$

$$\begin{aligned}
&\{100 * [-0.843 * \text{avg18F\_Y\_si13\_8x8\_std\_6\_ratio\_loss\_below5\%\_90\%} \\
&\quad -0.435 * \text{avg18F\_Y\_ati\_8x8\_std\_3\_ratio\_loss\_below5\%\_mean}]\}\Big|_{0}^{100}
\end{aligned}$$

$$\text{JERKY\_MOTION}_d =$$

$$\{100 * [0.879 * \text{avg18F\_Y\_si13\_8x8\_std\_6\_log\_gain\_above95\%\_90\%}]\}\Big|_{0}^{100}$$

10. CONCLUSIONS

The introduction of digital video compression, transmission, and storage systems into telecommunications networks has made it necessary to develop new objective quality measurement methods. This is because the performance of digital video systems is variable and depends upon the dynamic characteristics of both the original video (e.g., spatial detail, motion) and the digital transmission system (e.g., bit rate, error rate). We have presented a set of quality measurement techniques that have been tested over a wide range of digital video systems, from videoconferencing systems to broadcast television systems. These techniques include algorithms for calibrating processed digital video (spatial registration, temporal registration, gain and level offset estimation, and valid region estimation), estimating overall levels of perceptual impairments, and performing detailed root cause analysis of these impairments. Extensive tests, both subjective and objective, were conducted to verify the techniques presented here and to determine their measurement accuracies. We thus believe that the methods are robust and should find wide application not only to current digital video systems but also to future ones.

11. REFERENCES

[1] ANSI T1.801.01 – 1995, “American National Standard for Telecommunications – Digital Transport of Video Teleconferencing/Video Telephony Signals – Video Test Scenes for Subjective and Objective Performance Assessment,” American National Standards Institute.

[2] ANSI T1.801.02 – 1995, “American National Standard for Telecommunications – Digital Transport of Video Teleconferencing/Video Telephony Signals – Performance Terms, Definitions, and Examples,” American National Standards Institute.

[3] ANSI T1.801.03 – 1996, “American National Standard for Telecommunications – Digital Transport of One-Way Video Signals – Parameters for Objective Performance Assessment,” American National Standards Institute.

[4] ANSI Accredited Standards Working Group T1A1.5 contribution number T1A1.5/94-118R1, “Subjective test plan (tenth and final draft),” Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005, Oct. 1993.

[5] ANSI Accredited Standards Working Group T1A1.5 contribution number T1A1.5/94-148, “Correlation of objective and subjective measures of video quality,” Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005, Sep. 1994.

[6] ATIS Technical Report T1.TR.72 – 2001, “Methodological Framework for Specifying Accuracy and Cross-Calibration of Video Quality Metrics,” Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005, Oct. 2001.

[7] ATIS Technical Report T1.TR.73 – 2001, “Video Normalization Methods Applicable to Objective Video Quality Metrics Utilizing a Full Reference Technique,” Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005, Oct. 2001.

[8] ATIS Technical Report T1.TR.74 – 2001, “Objective Video Quality Measurement using a Peak Signal-to-Noise Ratio (PSNR) Full Reference Technique,” Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005, Oct. 2001.

[9] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson, “On the impact of policing and rate guarantees in diff-serv networks: A video streaming application perspective,” in Proc. *SIGCOMM 2001* (Association for Computing Machinery, Special Interest Group on Data Communications), Aug. 2001.

[10] G. W. Cermak, S. Wolf, E. P. Tweedy, M. H. Pinson, and A. A. Webster, “Validating objective measures of MPEG video quality,” *SMPTE Journal*, Vol. 107, No. 4, pp. 226-235, Apr. 1998.

[11] C. Fenimore, J. M. Libert, and S. Wolf, “Perceptual effects of noise in digital video compression,” *SMPTE Journal*, Vol. 109, pp. 178-186, Mar. 2000.

[12] ITU-R Recommendation BT.500, “Methodology for subjective assessment of the quality of television pictures,” Recommendations of the ITU, Radiocommunication Sector.

[13] ITU-R Recommendation BT.601, “Encoding parameters of digital television for studios,” Recommendations of the ITU, Radiocommunication Sector.

[14] ITU-T Recommendation H.261, “Video codec for audiovisual services at p x 64 kbit/sec,” Recommendations of the ITU, Telecommunication Standardization Sector.

[15] ITU-T Recommendation J.143, “User requirements for objective perceptual video quality measurements in digital cable television,” Telecommunication Standardization Sector.

[16] ITU-T Recommendation P.910, "Subjective video quality assessment methods for multimedia applications," Recommendations of the ITU, Telecommunication Standardization Sector.

[17] ITU-T Recommendation P.931, "Multimedia communications delay, synchronization, and frame rate measurement," Recommendations of the ITU, Telecommunication Standardization Sector.

[18] ITU-T COM 9-80-E, "Final report from the video quality experts group (VQEG) on the validation of objective models of video quality assessment," approved for release at VQEG meeting number 4, Ottawa, Canada, Mar. 2000.

[19] A. K. Jain, *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice-Hall Inc., 1989, pp. 348-357.

[20] C. Jones and D. J. Atkinson, "Development of opinion-based audiovisual quality models for desktop video-teleconferencing," in *Proc. 6th IEEE International Workshop on Quality of Service*, Napa, California, May 18-20, 1998.

[21] F. Pereira, "MPEG-4 video subjective test procedures and results," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 1, Feb. 1997.

[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 1989.

[23] SMPTE 125M, "Television – Component Video Signal 4:2:2 – Bit-Parallel Digital Interface," Society of Motion Picture and Television Engineers, 595 West Hartsdale Avenue, White Plains, NY 10607.

[24] SMPTE 170M, "SMPTE Standard for Television – Composite Analog Video Signal – NTSC for Studio Applications," Society of Motion Picture and Television Engineers, 595 West Hartsdale Avenue, White Plains, NY 10607.

[25] SMPTE Recommended Practice 187 – 1995, "Center, Aspect Ratio, and Blanking of Video Images," Society of Motion Picture and Television Engineers, 595 West Hartsdale Avenue, White Plains, NY 10607.

[26] United States Patent 5,446,492, "Perception-Based Video Quality Measurement System," Aug. 29, 1995.

[27] United States Patent 5,596,364, "Perception-Based Audio-Visual Synchronization Measurement System," Jan. 21, 1997.

[28] S. Voran and S. Wolf, "The development and evaluation of an objective video quality assessment system that emulates human viewing panels," in *Proc. International Broadcasting Convention (IBC)*, Jul. 1992, pp. 504-508.

[29] S. Voran, "Estimation of system gain and bias using noisy observations with known noise power ratio," NTIA Technical Report 02-395, July 2002.

[30] A. A. Webster, C. T. Jones, M. H. Pinson, S. D. Voran, and S. Wolf, "An objective video quality assessment system based on human perception," *Human Vision, Visual Processing, and Digital Display IV, Proceedings of the SPIE*, Vol. 1913, Feb. 1993, pp. 15-26.

[31] S. Wolf, "Measuring the end-to-end performance of digital video systems," *IEEE Transactions on Broadcasting*, Vol. 43, No. 3, pp. 320-328, Sep. 1997.

[32] S. Wolf and M. Pinson, "In-service performance metrics for MPEG-2 video systems," in *Proc. Made to Measure 98 - Measurement Techniques of the Digital Age Technical Seminar*, technical

conference jointly sponsored by the International Academy of Broadcasting (IAB), the ITU, and the Technical University of Braunschweig (TUB), Montreux, Switzerland, Nov. 12-13, 1998.

[33] S. Wolf and M. Pinson, “Spatial-temporal distortion metrics for in-service quality monitoring of any digital video system,” in *Proc. SPIE International Symposium on Voice, Video, and Data Communications*, Boston, MA, Sep. 1999.

[34] S. Wolf and M. Pinson, “The relationship between performance and spatial-temporal region size for reduced-reference, in-service video quality monitoring systems,” in *Proc. SCI / ISAS 2001 (Systematics, Cybernetics, and Informatics / Information Systems Analysis and Synthesis)*, Jul. 2001, pp. 323-328.

[35] M. Pinson and S. Wolf, “Video Quality Measurement User’s Manual,” NTIA Handbook 02-1, Feb. 2002.

APPENDIX A: SPECIAL SMPTE COLOR BAR

This appendix contains Matlab Version 6 routines for generating and storing the Rec. 601 SMPTE color bar with an embedded spatial registration pattern (see Figure 9). The routines are called from a Matlab command line window as follows:

- [y, cb, cr] = smpte_with_pattern;
- write_yuv ('c:\smpte_with_pattern.yuv', y, cb, cr);

A.1 ROUTINE SMPTE_WITH_PATTERN.M

```
function [varargout] = smpte_with_pattern()
%  [varargout] = SMPTE_WITH_PATTERN()
%
%  Generates an ITU-R Recommendation BT.601 SMPTE color bar with an embedded
%  spatial registration pattern.  Rec. 601 frame lines 206 to 285 (where
%  numbering begins at 1) contain this special ramp pattern that facilitates
%  spatial registration of NTSC fields 1 and 2.
%
%  An alternating line pattern of 10 lines (5 from each field) has also been
%  inserted in the overscan area from line 476 to line 485 to allow an easy
%  method to differentiate NTSC field 1 from NTSC field 2.
%
%  If one output argument is requested, returns [y] image.
%
%  If two output arguments are requested, returns [cb, cr] images.
%
%  If three output arguments are requested, returns [y, cb, cr] images.
%
%  The cb and cr images are pixel replicated to produce 720 pixels x 486 lines,
%  so that all three image planes [y, cb, cr] are 720 pixels x 486 lines.
%

num_rows=486;
num_cols=720;

%  Use 601 levels
black = 16;
white = 235;
c_black = 128;  %  For the Cb and Cr channels, black = 128
c_amp = 112;  %  Normal excursions of the cb and cr signals about c_black

%  Define the start and width for the EIA color bars and reverse blue color
%  bars.  Odd pixel start location so cb & cr sub-sampling by 2 will produce
%  correct 4:2:2 image pixels.
bar_start = 11;
bar_width = 100;  % Seven bars, each of bar_width pixels

%  Define EIA color bars, 75 percent amplitude, 100 percent saturation
eia_beg_line = 2;
eia_end_line = 205;

%  Y bars, from left to right
y_white = 180;
y_yellow = 162;
y_cyan = 131;
y_green = 112;
y_mag = 84;
y_red = 65;
y_blue = 35;

%  Cr bars, from left to right
cr_white = 128;
cr_yellow = 142;
cr_cyan = 44;
cr_green = 58;
cr_mag = 198;
cr_red = 212;
cr_blue = 114;
```

```
%  Cb bars, from left to right
cb_white = 128;
cb_yellow = 44;
cb_cyan = 156;
cb_green = 72;
cb_mag = 184;
cb_red = 100;
cb_blue = 212;

%  Basic ramp pattern is shown below.
%
%  Total ramp pattern consists of the basic ramp pattern rotated horizontally
%  between rows.  For the Y channel, the rotation is one pixel at a time
%  between successive rows.  If the ramp is also put on the Cb and Cr channels,
%  (you must uncomment code to put ramp on Cb and Cr), the rotation is two
%  pixels every other row.
%
%  The upper left quadrant of the pattern is generated by the above rotations.
%  Two mirror flips are then used to generate the upper right quadrant and
%  then the bottom half.  Finally, one Y column immediately to the right of
%  the center is deleted to yield a symmetrical right-left chroma pattern
%  (you must uncomment code to put ramp on Cb and Cr).
%
%  The goal of this pattern is to provide a means of aligning NTSC fields
%  1 and 2 vertically and horizontally while not overly stressing low bit
%  rate codecs.
%
%  Small ramp
%y_ramp = [16 16 72 72 128 128 184 184 240 240 184 184 128 128 72 72];

%  Large ramp
%y_ramp = [16 16 30 30 44 44 58 58 72 72 86 86 100 100 114 114 128 128 ...
%         142 142 156 156 170 170 184 184 198 198 212 212 226 226 240 240 ...
%         226 226 212 212 198 198 184 184 170 170 156 156 142 142 128 128 ...
%         114 114 100 100 86 86 72 72 58 58 44 44 30 30];

%  Combined ramp
y_ramp = [16 16 72 72 128 128 184 184 240 240 184 184 128 128 72 72 16 16 ...
        30 30 44 44 58 58 72 72 86 86 100 100 114 114 128 128 142 142 ...
        156 156 170 170 184 184 198 198 212 212 226 226 240 240 184 184 ...
        128 128 184 184 240 240 226 226 212 212 198 198 184 184 170 170 ...
        156 156 142 142 128 128 114 114 100 100 86 86 72 72 58 58 44 44 30 30];

c_ramp = y_ramp;

% number of times to repeat basic ramp in h direction for left half of pattern
num_ramps = 4;

ramp_beg_line = 206;  % start on NTSC field 1 (early field), even line number

% total number of ramp lines must be even => ramp_end_line is odd
ramp_end_line = 285;

ramp_start = 9;  % pixel start position of ramp, must be odd number

%  Define small EIA color bars
seia_beg_line = 286;
seia_end_line = 325;

%  Reverse blue color bars from left to right:
%  blue, black, magenta, black, cyan, black, white
blue_beg_line = 326;
blue_end_line = 365;

%  PLUGE test signal
pluge_beg_line = 366;
pluge_end_line = 475;

%  Three pulse [Y Cb Cr] level reference (lref): [y1 cb1 cr1], [y2 cb2 cr2],
%  [y3 cb3 cr3], all of duration lref_width.  Odd lref_start location so
%  cb & cr sub-sampling by 2 will produce correct 4:2:2 images.
lref_start = 9;  % pixel location to start 3-pulse level reference
lref_width = 126; % pixel width of each pulse in 3-pulse level reference
y1 = black;
cr1 = 95;  % 75 percent amplitude of 40 IRE "-I" phase modulation
```

```
cb1 = 158;

y2 = white;
cr2 = c_black;
cb2 = c_black;

y3 = black;
cr3 = 149;  % 75 percent amplitude of 40 IRE "+Q" phase modulation
cb3 = 174;

%  Three pulse black level reference (bref): black-bref_amp, black,
%  black+bref_amp, all of duration bref_width (no color, cb = cr = 128).
bref_start = lref_start+504;  % pixel location to start 3-pulse black ref
bref_width = 34;  % pixel width of black pulses
bref_amp = 9; % deviation from black (4 IRE)

%  Alternating line pattern, field 1 versus field 2
alt_beg_line = 476;
alt_end_line = 485;
alt_start = bar_start;  % pixel location to start alternating line pattern
alt_width = 7*bar_width;  % pixel width of alternating line pattern
y_low = black;
y_high = white;
cb_low = c_black - c_amp;
cb_high = c_black + c_amp;
cr_low = c_black - c_amp;
cr_high = c_black + c_amp;

%  Initialize the Y, Cb, and Cr images
y = ones(num_rows,num_cols)*black;
cb = ones(num_rows,num_cols)*c_black;
cr = ones(num_rows,num_cols)*c_black;

%  Generate the EIA color bars
y(eia_beg_line:eia_end_line, bar_start:bar_start+bar_width-1) = y_white;
y(eia_beg_line:eia_end_line, bar_start+bar_width:bar_start+2*bar_width-1) = y_yellow;
y(eia_beg_line:eia_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = y_cyan;
y(eia_beg_line:eia_end_line, bar_start+3*bar_width:bar_start+4*bar_width-1) = y_green;
y(eia_beg_line:eia_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = y_mag;
y(eia_beg_line:eia_end_line, bar_start+5*bar_width:bar_start+6*bar_width-1) = y_red;
y(eia_beg_line:eia_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = y_blue;

cb(eia_beg_line:eia_end_line, bar_start:bar_start+bar_width-1) = cb_white;
cb(eia_beg_line:eia_end_line, bar_start+bar_width:bar_start+2*bar_width-1) = cb_yellow;
cb(eia_beg_line:eia_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = cb_cyan;
cb(eia_beg_line:eia_end_line, bar_start+3*bar_width:bar_start+4*bar_width-1) = cb_green;
cb(eia_beg_line:eia_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = cb_mag;
cb(eia_beg_line:eia_end_line, bar_start+5*bar_width:bar_start+6*bar_width-1) = cb_red;
cb(eia_beg_line:eia_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = cb_blue;

cr(eia_beg_line:eia_end_line, bar_start:bar_start+bar_width-1) = cr_white;
cr(eia_beg_line:eia_end_line, bar_start+bar_width:bar_start+2*bar_width-1) = cr_yellow;
cr(eia_beg_line:eia_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = cr_cyan;
cr(eia_beg_line:eia_end_line, bar_start+3*bar_width:bar_start+4*bar_width-1) = cr_green;
cr(eia_beg_line:eia_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = cr_mag;
cr(eia_beg_line:eia_end_line, bar_start+5*bar_width:bar_start+6*bar_width-1) = cr_red;
cr(eia_beg_line:eia_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = cr_blue;

%  Generate the left half of ramp pattern.
y_ramp = repmat(y_ramp,[1 num_ramps]);
c_ramp = repmat(c_ramp,[1 num_ramps]);
h_len = length(y_ramp);
v_len_half = (ramp_end_line-ramp_beg_line+1)/2;
ramp_half_line = ramp_beg_line+v_len_half-1;

next_v = 1;
%  Generate upper left quadrant
for v_line = ramp_beg_line:ramp_half_line
   y(v_line,ramp_start:ramp_start+h_len-1) = y_ramp;
   %  Uncomment these two lines to put ramp on Cb and Cr
   %cb(v_line,ramp_start:ramp_start+h_len-1) = c_ramp;
   %cr(v_line,ramp_start:ramp_start+h_len-1) = c_ramp;

        %  Get setup for next assignment
   next_v = next_v+1;
   %  Rotate Y one element to the left every line
   y_ramp = [y_ramp(2:h_len) y_ramp(1)];

   %  Rotate C two elements to the left every other line (next_v = 3, 5, 7, ...)
   %  Uncomment the next three lines to put ramp on Cb and Cr
   %if (floor((next_v+1)/2) == (next_v+1)/2)
   %   c_ramp = [c_ramp(3:h_len) c_ramp(1:2)];
   %end
end

%  Generate the mirror image upper right quadrant of ramp pattern.
y(ramp_beg_line:ramp_half_line,ramp_start+h_len:ramp_start+2*h_len-1) = ...
   fliplr(y(ramp_beg_line:ramp_half_line,ramp_start:ramp_start+h_len-1));

%  Uncomment next four lines to put ramp on Cb and Cr
%cb(ramp_beg_line:ramp_half_line,ramp_start+h_len:ramp_start+2*h_len-1) = ...
%   fliplr(cb(ramp_beg_line:ramp_half_line,ramp_start:ramp_start+h_len-1));
%cr(ramp_beg_line:ramp_half_line,ramp_start+h_len:ramp_start+2*h_len-1) = ...
%   fliplr(cr(ramp_beg_line:ramp_half_line,ramp_start:ramp_start+h_len-1));

%  Generate the mirror image bottom half of the ramp pattern
y(ramp_half_line+1:ramp_end_line,ramp_start:ramp_start+2*h_len-1) = ...
   flipud(y(ramp_beg_line:ramp_half_line,ramp_start:ramp_start+2*h_len-1));

%  Uncomment next four lines to put ramp on Cb and Cr
%cb(ramp_half_line+1:ramp_end_line,ramp_start:ramp_start+2*h_len-1) = ...
%   flipud(cb(ramp_beg_line:ramp_half_line,ramp_start:ramp_start+2*h_len-1));
%cr(ramp_half_line+1:ramp_end_line,ramp_start:ramp_start+2*h_len-1) = ...
%   flipud(cr(ramp_beg_line:ramp_half_line,ramp_start:ramp_start+2*h_len-1));

%  Take out one Y column immediately to right of mirror point to produce
%  left-right symmetry in the chroma channels.  Fill with black.
y(ramp_beg_line:ramp_end_line,ramp_start+h_len:ramp_start+2*h_len-2) = ...
   y(ramp_beg_line:ramp_end_line,ramp_start+h_len+1:ramp_start+2*h_len-1);
y(ramp_beg_line:ramp_end_line,ramp_start+2*h_len-1) = ...
   black*ones(ramp_end_line-ramp_beg_line+1,1);

%  Generate the small EIA color bars
y(seia_beg_line:seia_end_line, bar_start:bar_start+bar_width-1) = y_white;
y(seia_beg_line:seia_end_line, bar_start+bar_width:bar_start+2*bar_width-1) = y_yellow;
y(seia_beg_line:seia_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = y_cyan;
y(seia_beg_line:seia_end_line, bar_start+3*bar_width:bar_start+4*bar_width-1) = y_green;
y(seia_beg_line:seia_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = y_mag;
y(seia_beg_line:seia_end_line, bar_start+5*bar_width:bar_start+6*bar_width-1) = y_red;
y(seia_beg_line:seia_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = y_blue;

cb(seia_beg_line:seia_end_line, bar_start:bar_start+bar_width-1) = cb_white;
cb(seia_beg_line:seia_end_line, bar_start+bar_width:bar_start+2*bar_width-1) = cb_yellow;
cb(seia_beg_line:seia_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = cb_cyan;
cb(seia_beg_line:seia_end_line, bar_start+3*bar_width:bar_start+4*bar_width-1) = cb_green;
cb(seia_beg_line:seia_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = cb_mag;
cb(seia_beg_line:seia_end_line, bar_start+5*bar_width:bar_start+6*bar_width-1) = cb_red;
cb(seia_beg_line:seia_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = cb_blue;

cr(seia_beg_line:seia_end_line, bar_start:bar_start+bar_width-1) = cr_white;
cr(seia_beg_line:seia_end_line, bar_start+bar_width:bar_start+2*bar_width-1) = cr_yellow;
cr(seia_beg_line:seia_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = cr_cyan;
cr(seia_beg_line:seia_end_line, bar_start+3*bar_width:bar_start+4*bar_width-1) = cr_green;
cr(seia_beg_line:seia_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = cr_mag;
cr(seia_beg_line:seia_end_line, bar_start+5*bar_width:bar_start+6*bar_width-1) = cr_red;
cr(seia_beg_line:seia_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = cr_blue;

%  Generate the reverse blue color bars
y(blue_beg_line:blue_end_line, bar_start:bar_start+bar_width-1) = y_blue;
y(blue_beg_line:blue_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = y_mag;
y(blue_beg_line:blue_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = y_cyan;
y(blue_beg_line:blue_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = y_white;

cb(blue_beg_line:blue_end_line, bar_start:bar_start+bar_width-1) = cb_blue;
cb(blue_beg_line:blue_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = cb_mag;
cb(blue_beg_line:blue_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = cb_cyan;
cb(blue_beg_line:blue_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = cb_white;

cr(blue_beg_line:blue_end_line, bar_start:bar_start+bar_width-1) = cr_blue;
cr(blue_beg_line:blue_end_line, bar_start+2*bar_width:bar_start+3*bar_width-1) = cr_mag;
cr(blue_beg_line:blue_end_line, bar_start+4*bar_width:bar_start+5*bar_width-1) = cr_cyan;
cr(blue_beg_line:blue_end_line, bar_start+6*bar_width:bar_start+7*bar_width-1) = cr_white;
```

```
%  Generate the PLUGE signal
y(pluge_beg_line:pluge_end_line, lref_start:lref_start+lref_width-1) = y1;
cb(pluge_beg_line:pluge_end_line, lref_start:lref_start+lref_width-1) = cb1;
cr(pluge_beg_line:pluge_end_line, lref_start:lref_start+lref_width-1) = cr1;

y(pluge_beg_line:pluge_end_line, lref_start+lref_width:lref_start+2*lref_width-1) = y2;
cb(pluge_beg_line:pluge_end_line, lref_start+lref_width:lref_start+2*lref_width-1) = cb2;
cr(pluge_beg_line:pluge_end_line, lref_start+lref_width:lref_start+2*lref_width-1) = cr2;

y(pluge_beg_line:pluge_end_line, lref_start+2*lref_width:lref_start+3*lref_width-1) = y3;
cb(pluge_beg_line:pluge_end_line, lref_start+2*lref_width:lref_start+3*lref_width-1) = cb3;
cr(pluge_beg_line:pluge_end_line, lref_start+2*lref_width:lref_start+3*lref_width-1) = cr3;

y(pluge_beg_line:pluge_end_line, bref_start:bref_start+bref_width-1) = black-bref_amp;
y(pluge_beg_line:pluge_end_line, bref_start+bref_width:bref_start+2*bref_width-1) = black;
y(pluge_beg_line:pluge_end_line, bref_start+2*bref_width:bref_start+3*bref_width-1) = ...
    black+bref_amp;

%  Generate the alternative line pattern
for i = alt_beg_line:2:alt_end_line
   y(i, alt_start:alt_start+alt_width-1) = y_high;
   cb(i, alt_start:alt_start+alt_width-1) = cb_high;
   cr(i, alt_start:alt_start+alt_width-1) = cr_high;
   y(i+1, alt_start:alt_start+alt_width-1) = y_low;
   cb(i+1, alt_start:alt_start+alt_width-1) = cb_low;
   cr(i+1, alt_start:alt_start+alt_width-1) = cr_low;
end

%  Smooth the horizontal Y transitions by taking 2 points from each side
%  and fitting an exact half cosine cycle to 6 samples, with the center 4
%  being the transition and the other 2 matching the former and later
%  amplitudes.
%
%  For the Cb and Cr channels, since they will be sub-sampled by two
%  horizontally, smooth these transitions by taking 1 point from each side
%  and fitting an exact half cosine cycle to 4 samples, with the center 2
%  being the transition and the other 2 matching the former and the
%  later amplitudes.
%
%  Don’t smooth the ramp pattern.
%

%  Smooth Y transitions, yt must be an even length vector.
nypts = 6;
yt = 0:(0.5/(nypts-1)):0.5;
ytsize2 = size(yt,2)/2;
for v_line = 1:ramp_beg_line-1
   i_trans = find(diff(y(v_line,:)));
   num_trans = size(i_trans,2);
   for i=1:num_trans
      fit_func = (y(v_line,i_trans(i))-y(v_line,i_trans(i)+1))*cos(2*pi*yt)/2.0 + ...
         (y(v_line,i_trans(i))+y(v_line,i_trans(i)+1))/2.0;
      y(v_line,i_trans(i)-(ytsize2-1):i_trans(i)+ytsize2) = fit_func;
   end
end
for v_line = ramp_end_line+1:num_rows
   i_trans = find(diff(y(v_line,:)));
   num_trans = size(i_trans,2);
   for i=1:num_trans
      fit_func = (y(v_line,i_trans(i))-y(v_line,i_trans(i)+1))*cos(2*pi*yt)/2.0 + ...
         (y(v_line,i_trans(i))+y(v_line,i_trans(i)+1))/2.0;
      y(v_line,i_trans(i)-(ytsize2-1):i_trans(i)+ytsize2) = fit_func;
   end
end

%  Smooth Cb transitions
ncpts = 2+(nypts-2)/2;
ct = 0:(0.5/(ncpts-1)):0.5;
for v_line = 1:ramp_beg_line-1
   i_trans = find(diff(cb(v_line,:)));
   num_trans = size(i_trans,2);
   for i=1:num_trans
      fit_func = (cb(v_line,i_trans(i))-cb(v_line,i_trans(i)+1))*cos(2*pi*ct)/2.0 + ...
         (cb(v_line,i_trans(i))+cb(v_line,i_trans(i)+1))/2.0;
      fit_func2 = zeros(1,nypts);
      fit_func2(1) = fit_func(1);
      for j=2:2:nypts-1
         fit_func2(j) = fit_func(2+(j-2)/2);
         fit_func2(j+1) = fit_func(2+(j-2)/2);
      end
      fit_func2(nypts) = fit_func(ncpts);
      cb(v_line,i_trans(i)-(ytsize2-1):i_trans(i)+ytsize2) = fit_func2;
   end
end
for v_line = ramp_end_line+1:num_rows
   i_trans = find(diff(cb(v_line,:)));
   num_trans = size(i_trans,2);
   for i=1:num_trans
      fit_func = (cb(v_line,i_trans(i))-cb(v_line,i_trans(i)+1))*cos(2*pi*ct)/2.0 + ...
         (cb(v_line,i_trans(i))+cb(v_line,i_trans(i)+1))/2.0;
      fit_func2 = zeros(1,nypts);
      fit_func2(1) = fit_func(1);
      for j=2:2:nypts-1
         fit_func2(j) = fit_func(2+(j-2)/2);
         fit_func2(j+1) = fit_func(2+(j-2)/2);
      end
      fit_func2(nypts) = fit_func(ncpts);
      cb(v_line,i_trans(i)-(ytsize2-1):i_trans(i)+ytsize2) = fit_func2;
   end
end

%  Smooth Cr transitions
for v_line = 1:ramp_beg_line-1
   i_trans = find(diff(cr(v_line,:)));
   num_trans = size(i_trans,2);
   for i=1:num_trans
      fit_func = (cr(v_line,i_trans(i))-cr(v_line,i_trans(i)+1))*cos(2*pi*ct)/2.0 + ...
         (cr(v_line,i_trans(i))+cr(v_line,i_trans(i)+1))/2.0;
      fit_func2 = zeros(1,nypts);
      fit_func2(1) = fit_func(1);
      for j=2:2:nypts-1
         fit_func2(j) = fit_func(2+(j-2)/2);
         fit_func2(j+1) = fit_func(2+(j-2)/2);
      end
      fit_func2(nypts) = fit_func(ncpts);
      cr(v_line,i_trans(i)-(ytsize2-1):i_trans(i)+ytsize2) = fit_func2;
   end
end
for v_line = ramp_end_line+1:num_rows
   i_trans = find(diff(cr(v_line,:)));
   num_trans = size(i_trans,2);
   for i=1:num_trans
      fit_func = (cr(v_line,i_trans(i))-cr(v_line,i_trans(i)+1))*cos(2*pi*ct)/2.0 + ...
         (cr(v_line,i_trans(i))+cr(v_line,i_trans(i)+1))/2.0;
      fit_func2 = zeros(1,nypts);
      fit_func2(1) = fit_func(1);
      for j=2:2:nypts-1
         fit_func2(j) = fit_func(2+(j-2)/2);
         fit_func2(j+1) = fit_func(2+(j-2)/2);
      end
      fit_func2(nypts) = fit_func(ncpts);
      cr(v_line,i_trans(i)-(ytsize2-1):i_trans(i)+ytsize2) = fit_func2;
   end
end

%  Want y
if (nargout == 1)
   varargout{1} = round(y);
end
%  Want cb, cr
if (nargout == 2)
        varargout{1} = round(cb);
   varargout{2} = round(cr);
end
%  Want y, cb, cr
if (nargout == 3)
        varargout{1} = round(y);
   varargout{2} = round(cb);
   varargout{3} = round(cr);
end
```

A.2 ROUTINE WRITE_YUV.M

```
function write_yuv(output_file,varargin)
%  WRITE_YUV(output_file,varargin)
%
%  Write a yuv (i.e., y, cb, cr) file in SMPTE 125M format.
%
%  If there are 3 input arguments in varargin [y, cb, cr], the input
%  variables cb and cr are subsampled by two in the column direction and a
%  yuv OUTPUT_FILE is generated where data is stored as bytes in the form:
%
%  (cb1 y1 cr1 y2 cb3 y3 cr3 y4 ...), where row 1 is output first, then
%  row 2, etc.
%
%  The file size in bytes is num_rows (or lines) x num_cols x 2 (i.e., 2 bytes
%  per pixel, since the cb and cr signals are subsampled by two).
%
%  If there is 1 input argument in varargin [y], a y OUTPUT_FILE is generated,
%  where data is stored as bytes in the form:
%
%  (y1 y2 y3 y4 ...), where row 1 is stored first, then row 2, etc.  The file
%  size in bytes is num_rows x num_cols.
%

fid = fopen(output_file, 'w');

%  One input argument in varargin, the y image
if (length(varargin) == 1);
   [num_rows,num_cols] = size(varargin{1});
   y = round(varargin{1});
   fwrite(fid, y', 'uint8');
end

%  Three input arguments in varargin, [y, cb, cr]
if (length(varargin) == 3);
   [num_rows,num_cols] = size(varargin{1});
   y = round(varargin{1});
   cb = round(varargin{2});
   cr = round(varargin{3});
   c = zeros(num_rows,num_cols);
   %  Subsample cb and cr by 2 and stuff into array
   for i=1:2:num_cols
        c(:,i) = cb(:,i);
        c(:,i+1) = cr(:,i);
   end
   %  Merge the y and c arrays
   y = y';
   c = c';
   merge = cat(1, reshape(c,1,num_cols,num_rows), reshape(y,1,num_cols,num_rows));
   merge = reshape(merge,2*num_cols,num_rows);
   %
   %  Clip at 0 and 255 before storing
   %
   clip_high = 255*ones(2*num_cols,num_rows);
   clip_low = zeros(2*num_cols,num_rows);
   merge = min(merge,clip_high);
   merge = max(merge,clip_low);

   fwrite(fid, merge, 'uint8');
end

fclose(fid);
```

APPENDIX B: MULTIPLE DATA SET FITTING (MDSF) ALGORITHM

Stephen D. Voran [*]

We are seeking the best linear relationship between a set of p objective video quality estimation parameters (parameters) and a set of n subjective video quality scores (scores). In order to find the most robust relationship, we would like to use all n of the scores available to us. Unfortunately, the scores are dissimilar because they come from m different subjective tests, some of which use different test procedures, different testing environments, and even different test scales. In an attempt to homogenize these heterogeneous scores, we allow a single first-order correction for all of the scores in each test:

$$\tilde{\mathbf{s}}_i = a_i \mathbf{s}_i + b_i \mathbf{1}\,, \quad i = 1 \text{ to } m\,, \tag{1}$$

where $\mathbf{s}_i$ is a column vector of the n_i scores in the ith test and $\mathbf{1}$ is a column vector of n_i ones. Note that

$$n = \sum_{i=1}^{m} n_i\ . \tag{2}$$

We combine the corrected scores into a single n x 1 vector $\tilde{\mathbf{s}}$ for notational convenience:

$$\tilde{\mathbf{s}}^{\mathsf{T}} = \left[\tilde{\mathbf{s}}_1^{\mathsf{T}}, \tilde{\mathbf{s}}_2^{\mathsf{T}}, \tilde{\mathbf{s}}_3^{\mathsf{T}}, \ldots, \tilde{\mathbf{s}}_m^{\mathsf{T}}\right]\ . \tag{3}$$

We have n samples from each of p parameters. These n samples correspond to the n scores. We form the n x p+1 parameter matrix **P**, where columns 1 through p contain parameters, and column p+1 holds the constant value 1:

$$\mathbf{P} = \left[\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \ldots, \mathbf{q}_p, \mathbf{1}\right]\ . \tag{4}$$

The p+1 x 1 weight vector **w** allows us to form a linear combination of the columns of **P** with the goal of approximating $\tilde{\mathbf{s}}$:

$$\tilde{\mathbf{s}} \approx \hat{\mathbf{s}} = \mathbf{P}\mathbf{w}\ . \tag{5}$$

[*] The author is with the Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce, 325 Broadway, Boulder, CO 80305.

Finally, we introduce the $n \times n$ diagonal cost matrix **C**. **C** contains entries $0 < c_j$, $j = 1$ to n. This cost matrix allows us to shape the fitting errors. For example, we can force smaller errors where we have higher confidence in the data, if we are willing to accept larger errors where we have less confidence in the data.

If $\{a_i\}_{i=1}^{m}$ and $\{b_i\}_{i=1}^{m}$ were known, we could calculate $\tilde{\mathbf{s}}$ using (1) and then find the cost-weighted least-squares solution to (5):

$$\min_{\mathbf{w}} \left|\mathbf{C}(\tilde{\mathbf{s}} - \hat{\mathbf{s}})\right|^2 \;\Rightarrow\; \mathbf{w} = \left(\mathbf{P}^{\mathsf{T}}\mathbf{C}^2\mathbf{P}\right)^{-1}\mathbf{P}^{\mathsf{T}}\mathbf{C}^2\tilde{\mathbf{s}} \;. \tag{6}$$

On the other hand, if **w** were known, we could calculate $\hat{\mathbf{s}}$ using (5) and then solve the m cost-weighted least-squares problems for $\{a_i\}_{i=1}^{m}$ and $\{b_i\}_{i=1}^{m}$:

$$\hat{\mathbf{s}}_i \approx \tilde{\mathbf{s}}_i = [\mathbf{s}_i, \mathbf{1}]\begin{bmatrix} a_i \\ b_i \end{bmatrix} = \mathbf{S}_i \begin{bmatrix} a_i \\ b_i \end{bmatrix}, \;\; i = 1 \text{ to } m\,, \tag{7}$$

where

$$\hat{\mathbf{s}}^{\mathsf{T}} = \left[\hat{\mathbf{s}}_1{}^{\mathsf{T}}, \hat{\mathbf{s}}_2{}^{\mathsf{T}}, \hat{\mathbf{s}}_3{}^{\mathsf{T}}, \ldots, \hat{\mathbf{s}}_m{}^{\mathsf{T}}\right]. \tag{8}$$

The cost-weighted least-squares solution to (7) is

$$\min_{a_i,\, b_i} \left|\mathbf{C}(\hat{\mathbf{s}}_i - \tilde{\mathbf{s}}_i)\right|^2 \;\Rightarrow\; \begin{bmatrix} a_i \\ b_i \end{bmatrix} = \left(\mathbf{S}_i{}^{\mathsf{T}}\mathbf{C}^2\mathbf{S}_i\right)^{-1}\mathbf{S}_i{}^{\mathsf{T}}\mathbf{C}^2\hat{\mathbf{s}}_i \;, \; i = 1 \text{ to } m. \tag{9}$$

In practice, neither **w** nor $\{a_i\}_{i=1}^{m}$ and $\{b_i\}_{i=1}^{m}$ is known *a priori*. The two least-squares problems do not combine into a single least-squares problem. One way to find a solution for **w**, $\{a_i\}_{i=1}^{m}$, and $\{b_i\}_{i=1}^{m}$ is to iterate between (6) and (9).

In general, both the scores and the parameters have errors in them, and a multiple data set fitting algorithm should acknowledge these two contributions to the total fitting error. We define parameter error vectors $\{\varepsilon_{pi}\}_{i=1}^{m}$ and score error vectors $\{\varepsilon_{si}\}_{i=1}^{m}$:

$$\tilde{\mathbf{s}}_i = a_i\left(\mathbf{s}_i + \varepsilon_{si}\right) + b_i\mathbf{1} = \hat{\mathbf{s}}_i + \varepsilon_{pi}, \;\; i = 1 \text{ to } m. \tag{10}$$

Joint error minimization (JEM) algorithms allow us to simultaneously minimize cost-weighted versions of these two errors and also allow the possibility of controlling their ratios:

$$\min_{a_i,\, b_i} \left|\mathbf{C}_i \varepsilon_{si}\right|^2 + \left|\mathbf{C}_i \varepsilon_{pi}\right|^2, \text{ such that } \frac{\left|\mathbf{C}_i \varepsilon_{si}\right|^2}{\left|\mathbf{C}_i \varepsilon_{pi}\right|^2} = r^2,\ i = 1 \,\text{to}\, m. \tag{11}$$

The n_i x n_i diagonal cost matrix $\mathbf{C}_i$ is the appropriate portion of the n x n cost matrix $\mathbf{C}$.

The error ratio $0 < r$ allows us to distribute the total fitting error between score error and parameter error in accordance with any prior knowledge of the error processes. Four JEM algorithms are described in [29]. These include an approach that uses total least squares [22], an approach that minimizes

$$\left|\mathbf{C}_i \varepsilon_{si}\right|^2 + \frac{r}{|a_i|}\left|\mathbf{C}_i \varepsilon_{pi}\right|^2,\ i = 1 \,\text{to}\, m, \tag{11a}$$

an approach that minimizes

$$\left|\mathbf{C}_i \varepsilon_{si}\right|^2 + \left|\mathbf{C}_i \varepsilon_{pi}\right|^2 + \lambda \left(\left|\mathbf{C}_i \varepsilon_{si}\right|^2 - r^2 \left|\mathbf{C}_i \varepsilon_{pi}\right|^2\right),\ i = 1 \,\text{to}\, m, \tag{11b}$$

using the Lagrange multiplier λ, and a direct estimation approach.

In practice, we have used $r = 1$, resulting in equal score and parameter errors. We use the notation $[a_i, b_i] = \text{JEM}(\mathbf{s}_i, \hat{\mathbf{s}}_i, r)$ to refer to the JEM algorithm that solves (10) and (11) for a_i and b_i.

The problem as we have defined it so far has two excess degrees of freedom, which prohibits a unique solution. This situation is easily remedied by constraining a_j=1, b_j=0 for some value of $1 \le j \le m$. This means that for the jth subjective test, the first-order correction to the data is null. Thus the multiple data set fitting algorithm will transform all other data to the scale of the jth subjective test. For this reason we refer to the jth subjective test as the reference test. For simplicity, and without loss of generality, we will assume j=1, so that the first subjective test is the reference subjective test. The constraints a_1=1, b_1=0 can be enforced by shifting and scaling $\mathbf{w}$, $\{a_i\}_{i=1}^{m}$, and $\{b_i\}_{i=1}^{m}$ at each iteration of the multiple data set fitting algorithm:

$$
\begin{aligned}
\tilde{w}_i &= w_i / a_1, && i = 1 \text{ to } p, \\
\tilde{w}_{p+1} &= \left(w_{p+1} - b_1\right) / a_1, && \\
\tilde{b}_i &= \left(b_i - b_1\right) / a_1, && i = 1 \text{ to } m, \\
\tilde{a}_i &= a_i / a_1, && i = 1 \text{ to } m.
\end{aligned}
\tag{12}
$$

We are now equipped to return to the original problem of fitting video quality parameters and scores. We select initial values of $\{a_i\}_{i=2}^{m}$ and $\{b_i\}_{i=2}^{m}$ using prior knowledge of the scales used in the *m* subjective tests. One simple and intuitive rule for selecting these initial values of a_i and b_i is that the resulting first-order correction should map the low quality end of the *i*th subjective test to the low quality end of the reference subjective test, and it should map the high quality end of the *i*th subjective test to the high quality end of the reference subjective test. Once this initialization is completed, the multiple data set fitting algorithm has three major steps per iteration. First it solves the least squares problem given in (6) yielding **w**. Next it uses a JEM algorithm to find $\{a_i\}_{i=1}^{m}$ and $\{b_i\}_{i=1}^{m}$ consistent with (10) and (11). Finally the normalization procedure in (12) is applied. These three steps are repeated until convergence criteria on one or more of a_i, b_i, and w_i are satisfied. We offer no mathematical proof that this iterative approach will always converge, but we note that in our applications it always has converged.

A multiple data set fitting algorithm is summarized below. As necessary, the notation used above is augmented with an iteration number parameter. For example, $\tilde{\mathbf{s}}_i(j)$ is the value of $\tilde{\mathbf{s}}_i$ in the *j*th iteration of the algorithm.

Inputs: $\{\mathbf{s}_i\}_1^m$, score vectors

P, parameter matrix

C, cost matrix

$\{\tilde{a}_i(0)\}_{i=1}^{m}, \{\tilde{b}_i(0)\}_{i=1}^{m}$, initial data set correction factors

```
j=0
While convergence criteria on w, and/or $\{\tilde{a}_i\}_{i=2}^{m}$, and/or $\{\tilde{b}_i\}_{i=2}^{m}$ are not satisfied
    $j=j+1$
    $\tilde{\mathbf{s}}_i(j)=\tilde{a}_i(j-1)\mathbf{s}_i+\tilde{b}_i(j-1)\mathbf{1}$ ,  $i=1 \text{ to } m$
    $\tilde{\mathbf{s}}(j)^{\mathsf{T}} = \left[\tilde{\mathbf{s}}_1(j)^{\mathsf{T}}, \tilde{\mathbf{s}}_2(j)^{\mathsf{T}}, \tilde{\mathbf{s}}_3(j)^{\mathsf{T}}, \ldots, \tilde{\mathbf{s}}_m(j)^{\mathsf{T}}\right]$
    $\mathbf{w}(j)=\left(\mathbf{P}^{\mathsf{T}}\mathbf{C}^2\mathbf{P}\right)^{-1}\mathbf{P}^{\mathsf{T}}\mathbf{C}^2\tilde{\mathbf{s}}(j)$
    $\hat{\mathbf{s}}(j)=\mathbf{P}\mathbf{w}(j)$
```

```
        Extract $\{\hat{\mathbf{s}}_i\}_{i=1}^{m}$ from $\hat{\mathbf{s}}(j)$, consistent with $\hat{\mathbf{s}}(j)^{\mathsf{T}} = [\hat{\mathbf{s}}_1(j)^{\mathsf{T}}, \hat{\mathbf{s}}_2(j)^{\mathsf{T}}, \hat{\mathbf{s}}_3(j)^{\mathsf{T}}, ..., \hat{\mathbf{s}}_m(j)^{\mathsf{T}}]$
        $[a_i(j), b_i(j)] = \mathrm{JEM}(\mathbf{s}_i, \hat{\mathbf{s}}_i(j), r)$,    $i = 1 \text{ to } m$
        $\tilde{b}_i(j) = (b_i(j) - b_1(j)) / a_1(j)$,    $i = 1 \text{ to } m$
        $\tilde{a}_i(j) = a_i(j) / a_1(j)$,    $i = 1 \text{ to } m$
    End

    $\tilde{w}_i(j) = w_i(j) / a_1(j)$,    $i = 1 \text{ to } p$
    $\tilde{w}_{p+1}(j) = (w_{p+1}(j) - b_1(j)) / a_1(j)$
    $\tilde{\mathbf{s}}_i(j) = \tilde{a}_i(j)\mathbf{s}_i + \tilde{b}_i(j)\mathbf{1}$,    $i = 1 \text{ to } m$
    $\hat{\mathbf{s}}(j) = \mathbf{P}\tilde{\mathbf{w}}(j)$
```

Outputs: $\{\tilde{a}_i(j)\}_{i=1}^{m}$, $\{\tilde{b}_i(j)\}_{i=1}^{m}$, final data set correction factors

$\tilde{\mathbf{s}}(j)$ corrected score vector

$\tilde{\mathbf{w}}(j)$ parameter weights

$\hat{\mathbf{s}}(j)$ parameter-based approximation to corrected score vector

In terms of enforcing the constraints a_1=1, b_1=0, the algorithm defined above may seem somewhat indirect. A seemingly more direct approach would be to select a_1=1, b_1=0 for the initial values, and then never apply the JEM algorithm to the reference data set. This "skip the reference data set" variation of the multiple data set fitting algorithm eliminates the need for the normalization steps described in (12). We have found that this variation also converges to a solution, but that convergence can be significantly slower than the convergence of the original algorithm. When skipping the JEM step on the reference data set, direct information about the relationship between the reference data set and the parameters is ignored. This information is apparently extracted in a slower and less direct way through the remaining m-1 data sets, leading to slower convergence. Depending on the scores and parameters, the two algorithms may converge to the same solution or to different solutions. There appears to be no general result on the relative optimality of the two solutions. The "skip the reference data set" variation of the multiple data set fitting algorithm is summarized below.

Inputs: $\{\mathbf{s}_i\}_1^m$, score vectors

P, parameter matrix

C, cost matrix

$a_1(0)=1,\ b_1(0)=0,\ \{a_i(0)\}_{i=2}^{m},\ \{b_i(0)\}_{i=2}^{m},$ initial data set correction factors

```
$j=0$
While convergence criteria on $\mathbf{w}$, and/or $\{a_i\}_{i=2}^{m}$, and/or $\{b_i\}_{i=2}^{m}$ are not satisfied
    $j=j+1$
    $\tilde{\mathbf{s}}_i(j)=a_i(j-1)\,\mathbf{s}_i+b_i(j-1)\,\mathbf{1}$ ,   $i=1$ to $m$
    $\tilde{\mathbf{s}}(j)^{\mathsf{T}}=\left[\tilde{\mathbf{s}}_1(j)^{\mathsf{T}},\tilde{\mathbf{s}}_2(j)^{\mathsf{T}},\tilde{\mathbf{s}}_3(j)^{\mathsf{T}},\ldots,\tilde{\mathbf{s}}_m(j)^{\mathsf{T}}\right]$
    $\mathbf{w}(j)=\left(\mathbf{P}^{\mathsf{T}}\mathbf{C}^2\mathbf{P}\right)^{-1}\mathbf{P}^{\mathsf{T}}\mathbf{C}^2\tilde{\mathbf{s}}(j)$
    $\hat{\mathbf{s}}(j)=\mathbf{P}\mathbf{w}(j)$
    Extract $\{\hat{\mathbf{s}}_i\}_{i=1}^{m}$ from $\hat{\mathbf{s}}(j)$, consistent with $\hat{\mathbf{s}}(j)^{\mathsf{T}}=\left[\hat{\mathbf{s}}_1(j)^{\mathsf{T}},\hat{\mathbf{s}}_2(j)^{\mathsf{T}},\hat{\mathbf{s}}_3(j)^{\mathsf{T}},\ldots,\hat{\mathbf{s}}_m(j)^{\mathsf{T}}\right]$
    $[a_i(j),\ b_i(j)]=\mathrm{JEM}\left(\mathbf{s}_i,\hat{\mathbf{s}}_i(j),r\right)$,   $i=2$ to $m$
End

$\tilde{\mathbf{s}}_i(j)=a_i(j)\,\mathbf{s}_i+b_i(j)\,\mathbf{1}$,        $i=1$ to $m$
```

Outputs: $\{a_i(j)\}_{i=1}^{m},\ \{b_i(j)\}_{i=1}^{m},$ final data set correction factors

$\tilde{\mathbf{s}}(j)$ corrected score vector

$\mathbf{w}(j)$ parameter weights

$\hat{\mathbf{s}}(j)$ parameter-based approximation to corrected score vector

APPENDIX C: ROOT CAUSE ANALYSIS (RCA) EXPERIMENT

C.1 GOAL OF EXPERIMENT

The objective of the root cause analysis (RCA) experiment was to determine relationships (i.e., mappings) between the ITS video quality parameters and specific perceived video artifacts. The experiment considered artifact types that are normally associated with digital video compression and transmission systems. The desired mappings were more qualitative than quantitative in that we wanted the viewers to provide the same RCA information regardless of the level of impairment (i.e., viewers were not asked to rate the severity of the artifacts, just their significance as contributors to the perceived quality level, whatever that level might be).

C.2 CLIP SELECTION AND VIEWING

A subset of 110 clips from the subjective data sets described in section 7 was randomly selected with the added requirement that the chosen clips have a normalized subjective quality rating of at least 0.1 (on a scale from 0.0 to 1.0, where 0.0 is no impairment and 1.0 is maximum impairment). The clips were also chosen to be approximately uniformly distributed in quality from 0.1 to 1.0. Viewers saw the original version first and then the processed version, and they were given the opportunity to replay the clip pair as many times as they felt necessary to complete the score form. Viewers were given the opportunity to take breaks as required and to continue their scoring when they returned. Each viewer saw a different randomized ordering of the 110 clips.

All clip pairs were stored in Rec. 601 format [13] and played back using a computer-based system. The tests used a calibrated professional grade 20-inch monitor with SMPTE 125M inputs [23] and viewers were positioned at a viewing distance of three times the picture height.

C.3 VIEWERS

The subjective experiment used six expert viewers from ITS. Due to the limited number of experts that were available, the results of this experiment should be viewed as preliminary.

C.4 ARTIFACT TYPES AND THEIR SIGNIFICANCE AS CONTRIBUTORS TO QUALITY

The expert viewers examined the video for the presence of the following artifacts: blurring, block distortion (i.e., tiling or error blocks), jerky/unnatural motion (i.e., dropped frames), added noise, and color distortions. See reference [2] for definitions of these artifacts. Viewers were asked to select the artifact that was the most significant contributor to quality degradation (i.e., one primary artifact must be selected). In addition, viewers were given the option to select secondary artifacts that were less significant contributors to quality degradation (i.e., zero or more secondary artifacts could be selected).

C.5 ARTIFACT SPATIAL EXTENT AND TIME DURATION

For each selected artifact (i.e., primary, secondary), viewers were asked to provide estimates of their spatial extent (local or global) and time duration (short or long). This allows for finer root cause analysis to be performed. For instance, block distortion that is very localized in space and time is normally referred to as error blocks (for a definition of error blocks, see [2]). It was also hoped that added information on the spatial and temporal extent of the impairment would provide insight into the optimal spatial and temporal collapsing functions (sections 5.3 and 5.4) for the objective quality parameters and the impairment root cause analysis estimators (section 9.2).

C.6 AUDIO USED FOR VIEWING SESSION

The following audio was used for the RCA experiment:

> *The Institute for Telecommunication Sciences is conducting a study to determine how different video impairments, or artifacts, influence video quality. We are interested in measuring the effects of five specific video artifacts; blurring, block distortion, jerky or unnatural motion, added noise, and color distortion.*
>
> *You will be shown a sequence of clip pairs that contain two versions of the same video scene. The first version in the clip pair will be the original scene. The second version in the clip pair will be the original scene passed through a video system that may have introduced one or more of the previously mentioned artifacts. For each clip pair, please select one primary artifact that you think is the most significant contributor to quality degradation. You may also select one or more secondary artifacts that you feel are lesser contributors to quality degradation. For each selected artifact, record if the artifact is present in a small part of the video picture (i.e., local) or if the artifact is present throughout the entire video picture (i.e., global). Also record the time duration of the artifact in relation to the total clip length (i.e., short versus long). Note, we are not interested in measuring the severity of the artifacts, just their presence.*
>
> *You may repeat each clip pair by pressing r on the keyboard, followed by return. To go to the next clip pair, just press return. Remember, there are no right or wrong answers. We are interested in how you, personally, perceive the presence of video artifacts.*
>
> *We now begin the viewing session.*
>
> *Clip 1, Clip 2, …., Clip 110*
>
> *This completes the viewing session. Thank you for your cooperation.*

C.7 SAMPLE SCORING FORM

The following is a sample of the scoring form that was used for the RCA experiment:

Clip	**Artifact**	**Significance**		**Spatial Extent**		**Time Duration**	
		Primary	Secondary	Local	Global	Short	Long
1	Blurring	○	○	○	○	○	○
	Block Distortion	○	○	○	○	○	○
	Jerky Motion	○	○	○	○	○	○
	Added Noise	○	○	○	○	○	○
	Color Distortion	○	○	○	○	○	○

C.8 POST PROCESSING OF RESULTS

The data was entered as ones and zeros into a spreadsheet (one if the oval was marked and zero otherwise). After re-ordering each of the viewers' responses to have the same sequential clip order (remember, each viewer saw a different randomized order), the following mathematical processes were performed:

1. The average response was computed by averaging the ones and zeros across viewers.
2. The total amount of each of the five artifacts (e.g., blurring) was computed as the primary response plus 0.5 times the secondary response.
3. Two sub-categories of the five artifacts were created using the spatial extent information (i.e., local or global). The "error block" artifact score was created as the product of the "spatial extent (local)" score and the "block distortion" score. The "global noise" artifact score was created as the product of the "spatial extent (global)" score and the "added noise" score.

Post-processing the subjective results in the above manner creates subjective scores that fall between zero and one for each artifact, where a one means that the artifact was perceived as being the primary artifact by all the viewers, a 0.5 means that the artifact was perceived as being a secondary artifact, and a 0.0 means the artifact was not perceived. For example, if all the viewers scored blurring as the primary artifact (the viewers could select only one primary artifact), the final blurring score would be 1.0. If all the viewers scored blurring as a secondary artifact (the viewers could select as many secondary artifacts as they wanted), the final blurring score would be 0.5. The final post-processed subjective data was used to develop the impairment RCA models given in section 9.2.