COMMITTEE T1

CONTRIBUTION

Document Number: T1A1.5/93-152

****

STANDARDS PROJECT: VTC/VT Performance Standards Projects

****

TITLE: A Summary of Methods of Measurement for Objective Video Quality Parameters Based on the Sobel Filtered Image and the Motion Difference Image

****

ISSUE ADDRESSED: Objective Video Performance Testing

****

SOURCE: NTIA/ITS, Stephen Wolf, Margaret Pinson, Coleen Jones, Arthur Webster

****

DATE: November 8, 1993

****

DISTRIBUTION TO: T1A1.5

****

KEYWORDS: Objective Video Performance Testing, Video Quality Parameters

****

DISCLAIMER: This contribution references specific vendor’s video test equipment. This is not intended to be an endorsement of any particular vendor’s products.

****

1 Introduction

The Institute for Telecommunication Sciences (ITS) has developed a number of objective performance parameters for digital video systems over the past 2 years. Contributions that document several of these parameters include T1Q1.5/92-112, T1A1.5/92-112, T1A1.5/92-135, T1A1.5/92-136, T1A1.5/92-138, T1A1.5/92-139, T1A1.5/93-032, and T1A1.5/93-060. ITS continues to develop and refine the set of technology independent video quality parameters.

This contribution, together with a companion contribution (T1A1.5/93-153), summarize the detailed methods of measurement for video quality parameters that have demonstrated strong correlation to subjective evaluations, and hence should be included in the upcoming video quality tests that are being conducted by T1A1.5. The level of detail in this contribution should be sufficient for any interested laboratory to implement the video quality measurements. Detailed pseudo-code has been included for several of the more complex algorithms to assure that they will be implemented in a consistent manner across laboratories. To that end, questions regarding the methods of measurement in this contribution should be directed to:

Stephen Wolf
U.S. Department of Commerce
NTIA/ITS.N3
325 Broadway
Boulder, CO 80303

Phone: (303) 497-3771
Fax: (303) 497-5323
E-mail: steve@its.bldrdoc.gov

To the extent that committee T1A1.5 validates the proposed measurements in this contribution (and in contribution T1A1.5/93-153), the detailed methods of measurement could become content material for the VTC/VT draft standard. Any improvements that are made by committee T1A1.5 during the evaluation of the proposed video quality parameters could also be included. ITS is currently in the process of performing the objective measurements in this document on the hypothetical reference circuit video (see contribution T1A1.5/93-014R1) and will be making these results available to committee T1A1.5.

Figure 1 presents a conceptual block diagram of an in-service perception-based video quality measurement system. The measurement system is composed of two sub-systems -- a source instrument and a destination instrument. The source instrument attaches non-intrusively to the source video and extracts a set of source features that can be used as a reference to quantify perceptual video quality changes. The destination instrument attaches non-intrusively to the destination video and extracts an identical set of destination features. An objective quality estimate of the transmission system performance can then be obtained by comparing the source features with the corresponding destination features. The contents of this document contribute to the design and validation of the video quality measurement system shown in Figure 1.

The video quality parameters presented here are computed using two, low-bandwidth features that are directly measured from the digitized source and destination video. One of these low-bandwidth features is derived from the Sobel filtered image and the other is derived from the motion difference image. Parameters based on the Sobel filtered image have proven useful for measuring spatial distortions such as image blurring. Parameters based on the motion difference image have proven useful for measuring temporal distortions such as jerky motion. The perception-based video quality features can be communicated with a very low bit rate, much less than the bit rate of the source video. Thus, the quality features can be easily and economically transferred between the source and destination instrument, which may be separated by large distances. The practicality of the measurements proposed here have been demonstrated in a real time PC-based implementation (see T1A1.5/93-105).

This contribution is organized as follows:

1. The methods of measurement for the low-bandwidth source and corresponding destination features (shown in Figure 1) are described in section 2 of this contribution. Two methods of measurement for each low-bandwidth quality feature will be described, one that is believed to be optimal and a second that is an approximate form of the first, but more computationally efficient. The ultimate intent is to pick just one method of measurement for each low-bandwidth quality feature.

2. An automatable method for properly time-aligning the source and corresponding destination features is described in section 3 of this contribution. This method of measurement can be used for computing the video delay of the HRC shown in Figure 1.

3. The methods for computing the objective video quality parameters (shown in Figure 1) from the time-aligned source and corresponding destination features are described in section 4 of this contribution.

4. Several extensions have been made to the basic video quality parameters. These extensions, described in section 5, seem to provide significant additional information about HRC performance. However, we are not at this time recommending that these extensions be included in the first round of objective performance parameter testing. The extensions are described in this contribution to inform committee T1A1.5 of ongoing video quality research efforts at ITS.

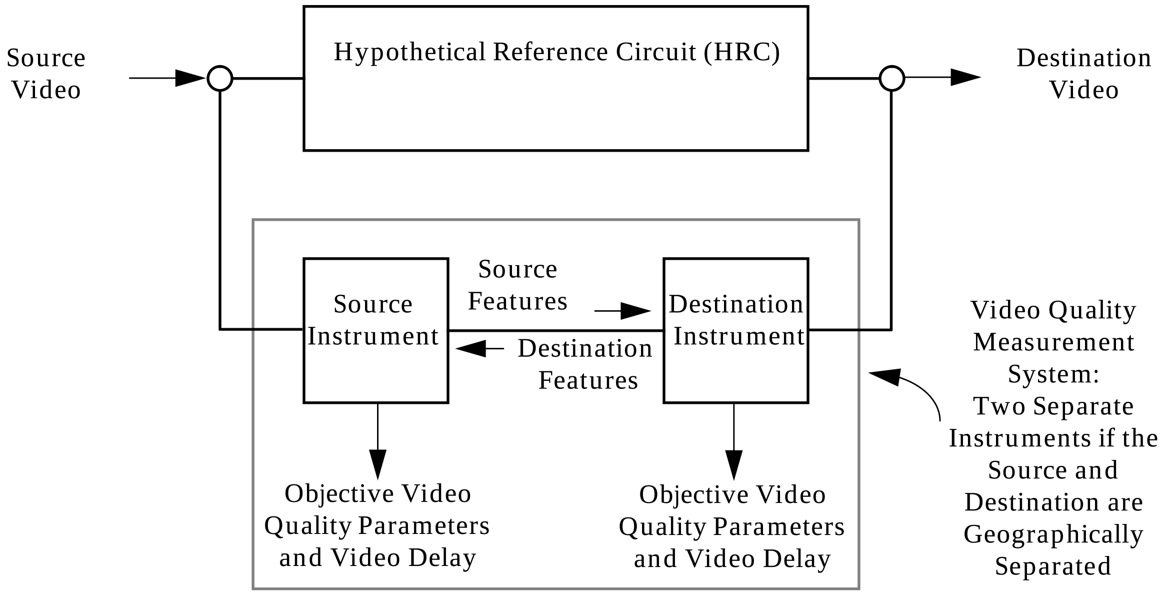

Figure 1 Conceptual View of Video Quality Measurement System

2 Methods of Measurement for Source and Destination Features

2.1 First Method of Measurement for Spatial Information of Source Video (SI_S) and Spatial Information of Destination Video (SI_D)

This section presents a method of measurement for the spatial information of the source video (SI_S) and the spatial information of the destination video (SI_D). SI_S and SI_D are features that are measured in an identical manner from the source video and the destination video in Figure 1. SI_S and SI_D characterize the spatial detail in the source and the destination video scene. Objective video quality parameters that measure spatial impairments in the destination video relative to the source video can be computed using SI_D and SI_S (see section 4). The preferred method of measurement for SI_S and SI_D is as follows.

1. Using CCIR Recommendation 601-2 (commonly referred to as D1 format), digitally sample the luminance signal of a source video frame and a destination video frame. Let $Y_S(t_n)$ represent a source luminance frame of video sampled at time t_n, and let $Y_D(t_m)$ represent a destination luminance frame of video sampled at time t_m. These luminance frames occur at the rate of approximately 30 times per second for NTSC video. ITS plans to process both the HRC tapes (which are in D2 format) and the subjective viewing tapes (which will be in Betacam SP format). The HRC tapes were first copied into Betacam SP format (since the highest quality format of the ITS video laboratory is Betacam SP), and the Y channel as output by the Betacam SP VCR was then sampled using CCIR Recommendation 601-2. The subjective viewing tapes can be processed as is, without any format conversion.

Note: Per CCIR Recommendation 601-2, the following attributes of the digitizing hardware should be checked before sampling: the frequency response should be flat (ideally within ± 0.05 db) from 0 to 5.75 MHz, 100 IRE (reference white) should be quantized at 235, and 7.5 IRE (reference black) should be quantized at 16.

2. Next, $Y_S(t_n)$ and $Y_D(t_m)$ are each filtered with the two 3 x 3 masks that are given in Figure 2. The filtering operation that detects horizontal edges is given by the convolution of the horizontal mask (H) with the luminance images, represented by $H*Y_S(t_n)$ and $H*Y_D(t_m)$. Likewise, the filtering operation that detects vertical edges is given by the convolution of the vertical mask (V) with the luminance images, represented by $V*Y_S(t_n)$ and $V*Y_D(t_m)$. The pseudo-sobel (PSobel) filtered luminance images combine the outputs of the two filtering operations as

$$PSobel(Y_S(t_n)) = |H*Y_S(t_n)| + |V*Y_S(t_n)|, \tag{1}$$

$$PSobel(Y_D(t_m)) = |H*Y_D(t_m)| + |V*Y_D(t_m)|, \tag{2}$$

where the absolute value and addition operations are performed on a pixel by pixel basis.

-1	-2	-1
0	0	0
1	2	1

H

-1	0	1
-2	0	2
-1	0	1

V

Figure 2 Horizontal and Vertical Edge Filters

3. SI_S at time t_n and SI_D at time t_m are then computed as the standard deviation (std) of the PSobel filtered images from step 2, i.e.,

$$SI_S(t_n) = std[PSobel(Y_S(t_n))], \tag{3}$$

and

$$SI_D(t_m) = \frac{std[PSobel(Y_D(t_m))]}{G}. \tag{4}$$

where the standard deviation is computed over all the samples in the *viewable region* of the processed D1 video frame (see Figure 3), and G is the gain of the transmission channel or HRC. Only the pixels contained within the *viewable region* of the D1 sampled video frame are used in equation (3) and equation (4) since SI_S and SI_D are perception-based. This *viewable region* is defined to consist of 672 pixels by 448 lines and is centered in the D1 video frame (which is composed of 720 horizontal pixels by 486 active lines).

Note: Normally, the video gain G will be very close to 1.0. The video gain G can be easily computed by comparing the white and black levels of the color bar signals on the D2 source tape with the corresponding white and black levels of the color bar signals on the D2 destination tape (i.e., HRC tape). If $source_{white}$ and $source_{black}$ are the 100 IRE (white) and 7.5 IRE (black) levels of the SMPTE color bar on the source tape, and $destination_{white}$ and $destination_{black}$ are the corresponding levels on the destination tape, then G is given by

$$G = \frac{destination_{white} - destination_{black}}{source_{white} - source_{black}}. \tag{5}$$

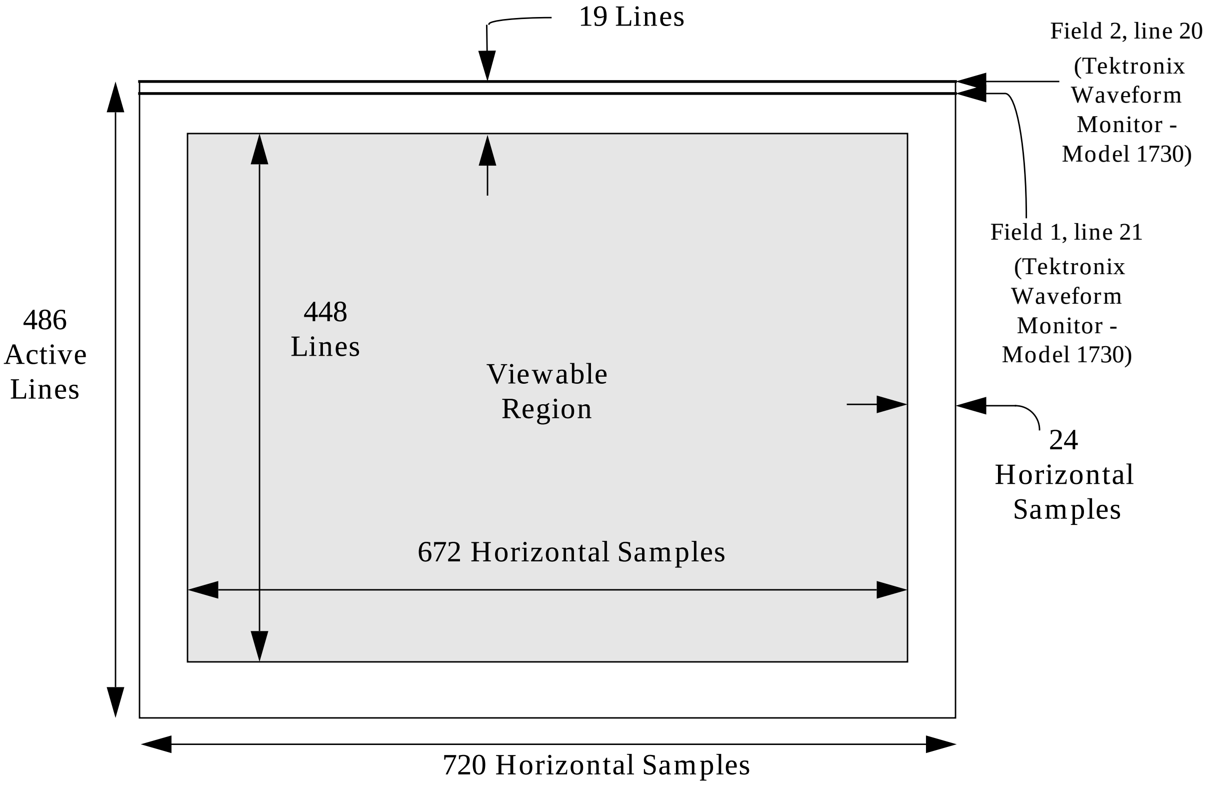

Figure 3 D1 *Viewable Region* for Computing Quality Features

4. For a video test scene, the time histories of $SI_S(t_n)$ and $SI_D(t_m)$ are computed as given in the above steps (i.e., SI_S and SI_D are computed for each frame of the source video and the destination video in the video test scene).

2.2 Alternate Method of Measurement for Spatial Information of Source Video (SI_S) and Spatial Information of Destination Video (SI_D)

This section presents an alternate method for computing the spatial information of source video (SI_S) and the spatial information of destination video (SI_D). This alternate method, previously proposed in contribution T1A1.5/93-105, is an approximation to the method of measurement presented in section 2.1 and is designed to run efficiently on real time image processing hardware. The alternate method presented here is determined by the characteristics of the ITS real time image processing hardware (documented in contribution T1A1.5/93-105). Although this alternate method is more computationally efficient, some degradation in performance may result.

1. Digitally sample the luminance portion of the source video and the destination video as described in step 1 of section 2.1.

2. As described in step 2 of section 2.1, compute the four images $H*Y_S(t_n)$, $H*Y_D(t_m)$, $V*Y_S(t_n)$, and $V*Y_D(t_m)$. These four, 10 bit images are right shifted by 2 bits (divided by four) before the absolute values are computed. The approximate pseudo-sobel (PSobel) filtered luminance images are then given by

$$PSobel(Y_S(t_n)) \approx \left|H*Y_S(t_n) >> 2\right| + \left|V*Y_S(t_n) >> 2\right|, \tag{6}$$

$$PSobel(Y_D(t_m)) \approx \left|H*Y_D(t_m) >> 2\right| + \left|V*Y_D(t_m) >> 2\right|, \tag{7}$$

where >>2 represents right shifting the images by 2 bits. When an overflow is encountered in equation (6) and equation (7), the data is clipped at 8 bits, or a value of 255.

3. SI_S at time t_n and SI_D at time t_m are then computed as given in step 3 of section 2.1, except that the approximated PSobel images of equation (6) and equation (7) are used.

4. SI_S at time t_n and SI_D at time t_m (from step 3) are multiplied by 4 to compensate for the divide by 4 in step 2.

5. For a video test scene, the time histories of $SI_S(t_n)$ and $SI_D(t_m)$ are computed as given in the above steps (i.e., SI_S and SI_D are computed for each frame of the source video and the destination video in the video test scene).

Note: Research conducted by ITS indicates that SI_S and SI_D can be sub-sampled in time by a factor of five (i.e., only computed for every fifth video frame) without appreciably affecting the measurement results. Thus, up to five frame times can be used to calculate SI_S or SI_D.

2.3 First Method of Measurement for Temporal Information of Source Video (TI_S) and Temporal Information of Destination Video (TI_D)

This section presents a method of measurement for the temporal information of the source video (abbreviated TI_S although the following discussion will consider three types, namely, $TImean_S$, $TIstd_S$, and $TIrms_S$) and the temporal information of the destination video (likewise abbreviated TI_D). TI_S and TI_D are features that are measured in an identical manner from the source video and destination video shown in Figure 1. The time histories of TI_S and TI_D characterize the flow of motion in the source and destination video scene. Objective video quality parameters that measure temporal impairments in the destination video relative to the source video can be computed using TI_D and TI_S (see section 4). This section will describe three types of temporal information, all of which have been shown to be useful. The first type of temporal information, denoted TImean, is measured from the mean of the absolute value of the motion difference image. The second type of temporal information, denoted TIstd, is measured from the standard deviation (std) of the absolute value of the motion difference image. From TImean and TIstd, the composite root mean square (rms) of the absolute valued motion difference image can be computed. This will be denoted TIrms. The preferred method of measurement for TImean, TIstd, and TIrms is as follows.

1. Digitally sample the luminance portion of the source video and the destination video as described in step 1 of section 2.1.

2. The components of the temporal information of the source video at time t_n ($TImean_S(t_n)$, $TIstd_S(t_n)$, and $TIrms_S(tn)$) and the components of the temporal information of the destination video at time t_m ($TImean_D(t_m)$, $TIstd_D(t_m)$, and $TIrms_D(t_m)$) are then computed as:

$$TImean_S(t_n) = mean\left|Y_S(t_n) - Y_S(t_{n-1})\right| \tag{8}$$

$$TIstd_S(t_n) = std\left|Y_S(t_n) - Y_S(t_{n-1})\right| \tag{9}$$

$$TIrms_S(t_n) = \sqrt{[TImean_S(t_n)]^2 + [TIstd_S(t_n)]^2} \tag{10}$$

and

$$TImean_D(t_m) = \frac{mean\left|Y_D(t_m) - Y_D(t_{m-1})\right|}{G}. \tag{11}$$

$$TIstd_D(t_m) = \frac{std\left|Y_D(t_m) - Y_D(t_{m-1})\right|}{G} \tag{12}$$

$$TIrms_D(t_m) = \sqrt{[TImean_D(t_m)]^2 + [TIstd_D(t_m)]^2} \tag{13}$$

The subtraction and absolute value in equation (8), (9), (11), and (12) are performed pixel by pixel and the mean and standard deviation (std) are computed using the resulting pixels that are in the *viewable region* (see Figure 3). G is the gain of the transmission channel or HRC and is computed as given in section 2.1, step 3.

3. For a video test scene, the time histories of $TIrms_S(t_n)$, and $TIrms_D(t_m)$ are computed as given in the above steps (i.e., these values are computed for each frame of source video and destination video in the video test scene).

2.4 Alternate Method of Measurement for Temporal Information of Source Video (TI_S) and Temporal Information of Destination Video (TI_D)

This section presents an alternate method for computing $TImean_S$ and $TImean_D$ defined in section 2.3. This alternate method, previously proposed in contribution T1A1.5/93-105, is an approximation to the method of measurement presented in section 2.3 and is designed to run efficiently on real time image processing hardware. The alternate method presented here is determined by the characteristics of the ITS real time image processing hardware (documented in contribution T1A1.5/93-105). Only TImean_S and TImean_D are considered here since the standard deviation type of temporal information requires pixel squaring, a very computationally expensive operation. Although this alternate method is more computationally efficient than that presented in section 2.3, some degradation in performance may result.

1. Digitally sample the luminance portion of the source video and the destination video as given in step 1 of section 2.1.

2. Sub-sample the luminance images from step 1 above by a factor of two in both the horizontal and vertical directions. The sub-sampling is performed such that the first horizontal line (top line) and the first vertical column (left column) of the image in Figure 3 is retained. The sub-sampled frames of video are 360 horizontal pixels by 243 lines, with a sub-sampled *viewable region* of 336 pixels by 224 lines. Let $F_S(t_n)$ represent a source luminance frame of video sub-sampled at time t_n, and let $F_D(t_m)$ represent a destination luminance frame of video sub-sampled at time t_m (these sub-sampled frames are actually NTSC fields sub-sampled by 2 in the horizontal direction).

3. The temporal information of the source video at time t_n ($TI_S(t_n)$) and the temporal information of the destination video at time t_m ($TI_D(t_m)$) are then computed as:

$$TImean_S(t_n) \approx mean\left|F_S(t_n) - F_S(t_{n-1})\right| \tag{14}$$

and

$$TImean_D(t_m) \approx \frac{mean\left|F_D(t_m) - F_D(t_{m-1})\right|}{G} \tag{15}$$

The subtraction and absolute value in equation (14) and (15) are performed pixel by pixel and the mean is computed using the resulting pixels that are in the sub-sampled *viewable region* (see step 2 above). G is the gain of the transmission channel or HRC and is computed as given in section 2.1, step 3. Equations (14) and (15) may yield values that differ significantly from equations (8) and (11) if the same motion is not present in both NTSC fields, such as for film generated NTSC video (see section 5.2), or HRC frame updates that do not occur on a frame boundary.

4. For a video test scene, the time histories of $\mathrm{TImean}_S(t_n)$ and $\mathrm{TImean}_D(t_m)$ are computed as given in the above steps (i.e., these values are computed for each frame of source video and destination video in the video test scene).

3 Method for Time Alignment of Source and Destination Features (i.e., Video Delay)

The time histories of the source features (SI_S and TI_S) and destination features (SI_D, TI_D) must be aligned before the video quality parameters presented in section 4 are computed. The motion energy in the source and destination video scenes (i.e., the time histories of TI_S and TI_D) can be used to perform this time alignment. Either the TIrms_S and TIrms_D temporal information components, or the TImean_S and TImean_D temporal information components can be used for time alignment. The time alignment may either be performed manually or with an automated algorithm. This section presents an automated algorithm for performing the time alignment. This algorithm can be used to measure the video delay of the transmission channel since this delay is given by the backward time shift of the destination features relative to the source features when they are properly aligned. The algorithm used for the time alignment is an improved version of the algorithm that was presented in a prior contribution (T1A1.5 / 92-139).

This section is divided into four sub-sections. Definitions of alignment terms are covered in section 3.1. A maximum filtering sub-algorithm that filters the TI_S and TI_D waveforms before they are time aligned is described in section 3.2. Another sub-algorithm for computing the time alignment based on the minimum standard deviation of the (*max filtered* TI_S - *max filtered* TI_D) difference waveforms is detailed in section 3.3. Finally, a robust alignment algorithm is described in section 3.4 that utilizes the two sub-algorithms of section 3.2 and section 3.3.

The behavior of the alignment algorithms described here are determined by the values of several constants (or thresholds). Recommended values for these constants presented in this document have been determined heuristically.

3.1 Definitions

This section contains a list of definitions for terms that will be used in the description of the time alignment algorithms (see Figure 4).

Spike: If the current time sample of TI is greater than both the previous time sample and the following time sample, then a *spike* is said to have occurred at the current time.

Spike Height: The *spike height* is the difference between the current time sample of TI and the larger of the two adjacent samples.

Scene Cut: Any *spike* in TI_S that has a *spike height* greater than 15.0 is called a *scene cut*.

Scene Width: The length (in frames) of the scene portion that is used for time alignment. [For the T1A1.5 tests, a non-drop frame mode, nine second scene portion will be 270 frames long.]

Vector: Any time history of TI samples of a length equal to *scene width*.

Source Vector: A *vector* of $\mathrm{TI}_\mathrm{S}(\mathrm{t}_\mathrm{n})$ samples (or *max filtered* $TI_S(t_n)$ samples) measured from the source scene.

Destination Vector: A *vector* of $\mathrm{TI}_\mathrm{D}(\mathrm{t}_\mathrm{m})$ samples (or *max filtered* $TI_D(t_m)$ samples) measured from the destination scene.

Reference Vector: The specific *source vector* which is measured from the portion of the source scene that is of interest to the user (specified by the user). [For the T1A1.5 tests, the *reference vector* is measured from the 270 frame portion of the source scene that is subjectively viewed.]

Guess Vector: The specific *destination vector* which forms the user’s best time alignment guess to the *reference vector*.

Uncertainty: Uncertainty of the *guess vector’s* alignment, in frames. The true alignment must be within ± *uncertainty* samples from the user’s guess. [The recommended setting for *uncertainty* is 60 samples, which allows for an unknown HRC video delay of 4 seconds if the *guess vector* is chosen to have a video delay of 2 seconds.]

Window: The time alignment of any *destination vector* that is within ± *window* samples from the *guess vector* can be computed by the alignment subroutines. For robustness, a ± *window* of *destination vectors* around the *guess vector* are examined by the summarized alignment algorithm in section 3.4. [The recommended setting for *window* is *uncertainty* / 2, or 30 samples.]

Offset: The distance (in frames) between a *source vector* and the *reference vector* or between a *destination vector* and the *guess vector*. *Offsets* may be positive or negative. A positive *offset* means that the *source vector* occurs later in time than the *reference vector* (or that the *destination vector* occurs later in time than the *guess vector*).

Max Filtered TI_D: A maximum filtered version of $\mathrm{TI_D}$ (see section 3.2).

Max Filtered TI_S: A maximum filtered version of $\mathrm{TI_S}$ (see section 3.2).

Smoothed TI_D*:* A smoothed, low pass filtered version of $\mathrm{TI_D}$ that can be used to separate $\mathrm{TI_D}$ *spikes* of jerky motion from $\mathrm{TI_D}$ samples that do not contain motion. $\mathrm{TI_D}$ *spikes* of jerky motion have been eliminated in the *smoothed* TI_D. For low bit rate HRCs, a centered Hanning filter with a *filter width* of 63 frames works well provided the HRC transmits at least one frame every 2 seconds. To properly apply this Hanning filter to a *destination vector*, at least 32 $\mathrm{TI_D}$ samples must be available before and after the *destination vector*.

Fraction Above: Measures the fraction of $\mathrm{TI_D}$ that is on or above the corresponding *smoothed* TI_D. The fraction computation is based on (*scene width* + 2**window* + 2**uncertainty*) data points. The *fraction above threshhold* specifies the minimum *fraction above* that must be met before minimum standard deviation alignment is performed on the resulting *max filtered* TI_D and *max filtered* TI_S waveforms. [A *fraction above threshold* from 0.60 to 0.90 may be used; the recommended value is 0.70]

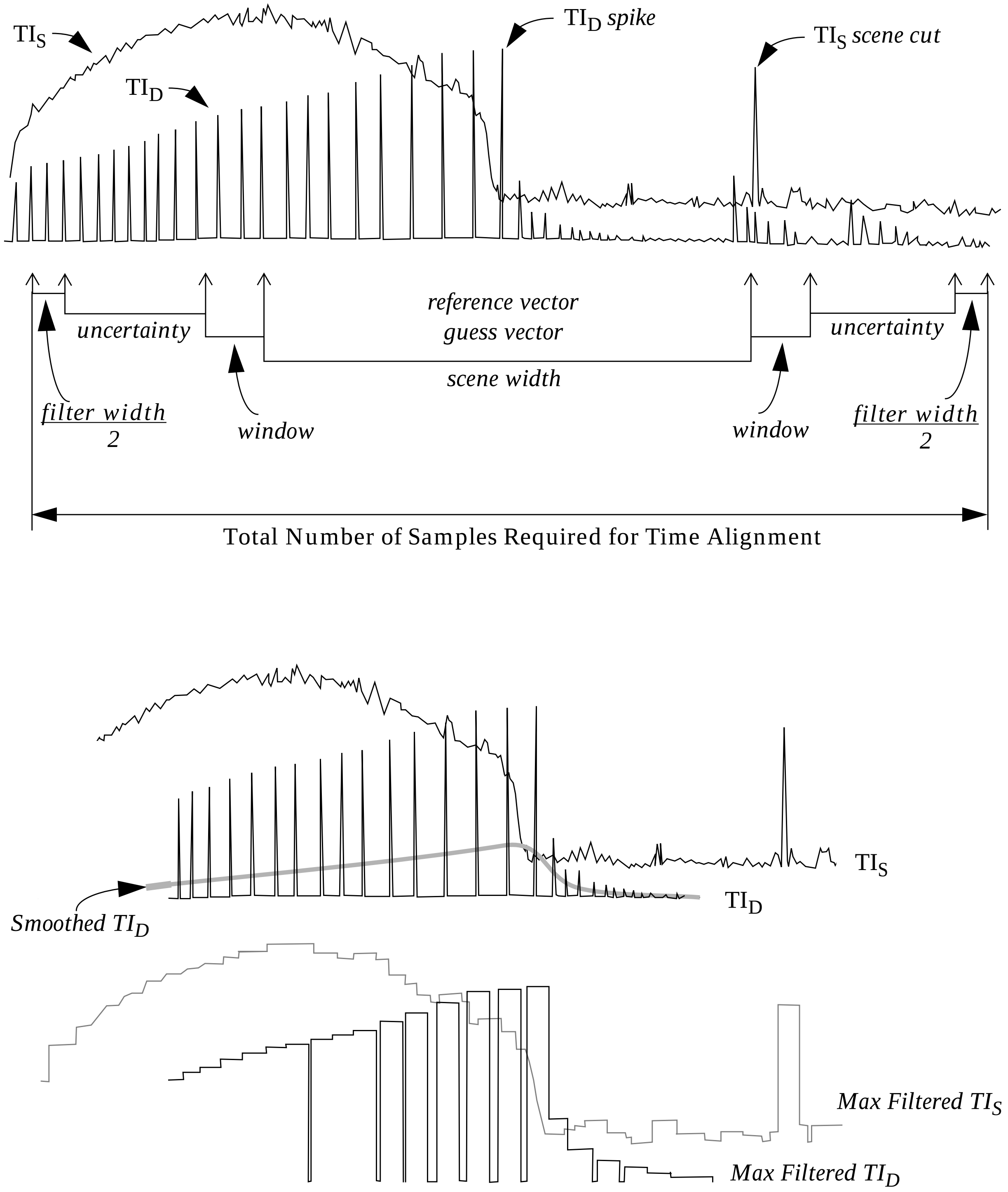

Figure 4 Definitions of Alignment Terms

3.2 Description of Maximum Filtering Algorithm

The plot shown in Figure 4 depicts a TI_D waveform that is typical of a frame repeating codec (a codec that updates the video less than 30 frames per second). As seen in the figure, samples of TI_D that contain motion appear as spikes. These motion spikes, which are visually perceived as jerky motion, are followed by one or more low or non-motion samples of TI_D where frame repetition is occurring. The non-motion samples of TI_D that correspond to the repeated video frames do not contain information relevant to alignment. The TI_D motion spikes, however, form a shape similar to the TI_S plot. The purpose of the maximum filter is to increase the similarity between the TI_D and TI_S waveforms by replacing the frame repeat samples of TI_D with the nearest motion spike samples. After the TI_D and TI_S waveforms are maximum filtered as described here, an algorithm for computing the time alignment based on the minimum standard deviation of the (*max filtered TI_S - max filtered TI_D*) difference waveforms can be applied (section 3.3).

The maximum filter algorithm has three components. One component, *fraction above,* measures the fraction of TI_D that does not contain frame repeats. A second component is a three wide maximum filter that iteratively increases the *fraction above* of TI_D and TI_S (TI_D and TI_S are updated with their maximum filtered versions on each iteration). A third component, the *fraction above threshhold*, establishes the minimum threshhold for the *fraction above* of TI_D such that resulting *max filtered TI_D* contains sufficiently few frame repeats for alignment to begin.

The first component of the maximum filtering algorithm, *fraction above,* measures approximately how many samples of TI_D are frame repeats and how many samples contain motion information for alignment. *Fraction above* estimates the fraction of samples that are not frame repeats. *Fraction above* applies a low pass filter to TI_D to produce the corresponding *smoothed TI_D* waveform. The fraction of TI_D that lies on or above the corresponding *smoothed TI_D* will increase as the number of frames repeats in TI_D decreases. The number of frame repeats in TI_D will decrease as TI_D is iteratively replaced with its three wide maximum filtered version.

The second component of the maximum filtering algorithm, a three wide maximum filter, provides a means of replacing frame repeat samples with the nearest motion spike samples, thereby raising the frame repeat samples to the level of the surrounding motion-spike curve. To that end, the three wide maximum filter iteratively replaces TI_D with TI_D', where $TI_D'(t_m)$ is the maximum of $TI_D(t_{m-1})$, $TI_D(t_m)$, and $TI_D(t_{m+1})$. Similarly, TI_S is iteratively replaced with TI_S'. Repeated application of the three wide maximum filter will spread motion spikes in TI_D into the surrounding frame repeats, as well as emphasizing scene cuts in TI_D and TI_S. This three wide maximum filter has the property that the *fraction above* for TI' is greater than the *fraction above* for TI.

The third component of the maximum filtering algorithm, the *fraction above threshhold,* establishes when the iterative max filtering operation of the three wide maximum filter can stop. To that end, the three wide maximum filter is applied repeatedly to both TI_S and TI_D until the *fraction above* for TI_D is greater than or equal to the *fraction above threshhold.* A TI_D *vector* with *fraction above* greater than the *fraction above threshhold* contains few frame repeat samples. To assure this condition is met, the *fraction above threshold* should be set from 0.60 to 0.90, with 0.70 being the recommended value. Example *max filtered TI_S* and *max filtered TI_D* waveforms produced by the maximum filtering algorithm are depicted in Figure 4.

3.2.1 Pseudo Code

Function **max_filter** replaces TI_D and TI_S arrays with their maximum filtered versions (i.e., *max filtered TI_D* and *max filtered TI_S*). The function **max_filter** calls functions **percent_above()** and **three_filter()**, which are defined below.

function **max_filter** (source, destination, source_length, destination_length,
fraction_above_threshhold, filter_width)

source	The TI_S array numbered from 0 to (source_length - 1).
destination	The TI_D array numbered from 0 to (destination_length - 1).
source_length	Length of the TI_S array source (*scene width* + 2**uncertainty* + (*filter width* - 1) + 2**window*).
destination_length	Length of the TI_D array destination (*scene width* + 2**uncertainty* + (*filter width* - 1) + 2**window*)
fraction_above_threshhold	Specifies the *fraction above threshhold.*
filter_width	Specifies the *filter width* for *smoothed TI_D*, an odd integer.

```
BEGIN
    fraction = fraction_above (destination, destination_length, filter_width)
    WHILE (fraction < fraction_above_threshhold) DO
    BEGIN
        three_filter (source, source_length)
        three_filter (destination, destination_length)
        fraction = fraction_above (destination, destination_length, filter_width)
    END
END
```

Function **fraction_above** applies a *filter width* wide Hanning filter to a TI_D array and then computes the number of TI_D samples which are on or above the resulting *smoothed TI_D*. Notice that the first and last coefficients of the Hanning filter are zero; these coefficients are included for the sake of clarity.

function **fraction_above** (destination, length, filter_width)

destination	The TI_D array.
length	Length of the TI_D array destination.
filter_width	Specifies the *filter width* for *smoothed TI_D*, an odd integer.

```
BEGIN

    /* N is the size of the Hanning filter, an odd integer > 1 */

    N = filter_width
    above = 0
    frame = (N-1) / 2
    WHILE (frame < length - (N-1)/2)
    BEGIN
```

```
        /* Apply a Hanning filter, centered around the current frame */

        col = frame - (N-1) / 2
        cnt = 0
        han= 0
        WHILE (cnt < N) DO
        BEGIN
            han = han + $^1/_2$ (1 - cos($2\pi$ * cnt / (N - 1)) * destination[col]
            cnt = cnt + 1
            col = col + 1
        END

        /* Normalize Hanning coefficients to add to one for a unit gain filter */

        han = han / ((N-1)/2)

        /* Update the number of frames on or above the Hanning filter */

        IF (destination[frame] $\geq$ han) THEN
            above = above + 1
        frame = frame+1
    END
    RETURN (above / (length - N + 1))
END
```

Function **three_filter** applies an in place, three wide maximum filter to an array. It calls function **maximum**(a,b,c), which returns the largest of the three numbers **a**, **b**, and **c**.

Function **three_filter** (temporal, length)

temporal	The array holding the TI samples.
length	Length of the array temporal.

```
BEGIN
    frame = 1
    overlap = temporal[0]
    WHILE (frame < length - 1) DO
    BEGIN
        max = maximum (overlap, temporal[frame], temporal[frame+1])
        overlap = temporal[frame]
        temporal[frame] = max
        frame = frame + 1
    END
END
```

3.3 Minimum Standard Deviation Alignment

The minimum standard deviation alignment algorithm compares one *max filtered* TI_D *destination vector* with (2**uncertainty* + 1) different *max filtered* TI_S *source vectors.* The *destination vector* input to the algorithm may be *offset* from the *guess vector* by ± *window* samples. Thus, for a given *destination vector,* the *source vectors* that are used are those vectors *offset* from the *reference vector* by (the *destination vector offset* ± *uncertainty* samples). For each *source vector*, take the difference between the *source vector* and the *destination vector.* Compute the standard deviation of this difference *vector.* The *source vector* with the minimum standard deviation is the best alignment to the *destination vector.*

When the standard deviation measure does not yield a clear minimum, the best alignment is *ambiguous.* To identify this case, find the *source vector* with the second smallest standard deviation. If this *source vector* (1) is far away from the minimum standard deviation *source vector* and, (2) has a standard deviation less than *comparable magnitude* times the minimum standard deviation, then alignment is said to be *ambiguous.* [For low bit rate HRCs, the recommended setting for *far away* is five samples (one sixth of a second) and the recommended setting for *comparable magnitude* is 1.5.]

3.3.1 Pseudo Code

Function **std_align** returns (the *offset* of the *source vector* that is the minimum standard deviation alignment to a particular *destination vector* minus the *offset* of that *destination vector*). Thus, the function returns the direction (+ or -) and the number of samples to shift the TI_D waveform so that it is time aligned to the TI_S waveform. The function is called with the maximum filtered versions of TI_D and TI_S (i.e., *max filtered* TI_D and *max filtered* TI_S) if the HRC is transmitting less than 30 frames per second.

Function **std_align** (source, destination, filter_width, uncertainty, window, scene_width, d_offset)

source Array that holds the *source vectors* (contains TI_S samples numbered from 0 to (*scene width* + 2**uncertainty* + (*filter width* - 1) + 2**window* - 1).

destination Array that holds the *destination vector*s (contains TI_D samples numbered from 0 to *scene width* + 2**uncertainty* + (*filter width* - 1) + 2**window* - 1).

filter_width Same as *filter width* defined in section 3.1, an odd integer.

uncertainty Same as *uncertainty* defined in section 3.1.

window Same as *window* defined in section 3.1.

scene_width Same as *scene width* defined in section 3.1.

d_offset Specifies the *offset* of the particular *destination vector* within the array destination to use for alignment. This *offset*, defined in section 3.1, must fall within the range of -*window* to +*window*, inclusive.

```
BEGIN
    minimum_std = MAXIMUM_POSSIBLE_STANDARD_DEVIATION
    second_std = MAXIMUM_POSSIBLE_STANDARD_DEVIATION

    /* start is the first element of the destination vector specified by d_offset */

    start = (filter_width-1)/2 + uncertainty + window + d_offset
    /* loop is the first element of the source vector currently being examined */

    loop = start - uncertainty
    WHILE (loop ≤ start+uncertainty) DO
    BEGIN

        /* Sum difference and difference squared */

        frame = 0
        total = 0
        mean = 0.0
        std = 0.0
        WHILE (frame < scene_width) DO
        BEGIN
                total = total + 1
                mean = mean + source[frame+loop] - destination[start+frame]
                std = std + (source[frame+loop] - destination[start+frame])$^2$
                frame = frame + 1
        END

        /* Approximate standard deviation from sum & sum of squares */

        mean = mean / total
        std = sqrt (std / total - mean * mean)

        /* Update minimum & second smallest standard deviation alignments */

        IF (std < minimum_std) THEN
        BEGIN
            second_std = minimum_std
            second_align = minimum_align

            minimum_std = std
            minimum_align = loop
        END
        ELSE IF (std < second_std) THEN
        BEGIN
            second_std = std
            second_align = loop
        END
    END

    /* Alignment ambiguous if second smallest is far way and of comparable magnitude */

    IF ((|second_align - minimum_align| > 5)
        AND (second_std / minimum_std ≤ 1.5)) THEN
            RETURN (AMBIGUOUS)
    ELSE    RETURN (minimum_align - start)
END
```

3.4 Alignment Algorithm

The maximum filtering and minimum standard deviation sub-algorithms described in section 3.2 and section 3.3 can be combined to produce a very robust alignment algorithm. This alignment algorithm is described in detail below. In summary, the alignment algorithm consists of three steps. First, the maximum filter described in section 3.2 is applied to the TI_S and TI_D waveforms. Second, the minimum standard deviation alignment algorithm is computed for every *destination vector* that is within ± *window* of the *guess vector*. By examining the histogram of the alignment results from this step, one can determine if the alignment with the most votes is *ambiguous* or not. If the alignment is deemed *ambiguous*, then step three is performed. *Ambiguous* alignment can result from step 2 when the *max filtered* TI_D waveform is significantly different in magnitude from the *max filtered* TI_S waveform. Step three performs a square root scaling of the *max filtered* TI_S and *max filtered* TI_D waveforms before repeating the operations of step 2. The purpose of the square root scaling is to weight TI variations resulting from small motion changes more than TI variations resulting from large motion changes, thereby increasing the similarity between the *max filtered* TI_S and *max filtered* TI_D waveforms.

The complete alignment algorithm will return the direction (+ or -) and the number of samples to shift the TI_D *guess vector* so that it is time aligned to the TI_S *reference vector*. If $TI_S(t_{n=N})$ and $TI_D(t_{m=M})$ are two corresponding time aligned TI_S and TI_D samples, then the video delay (d_v) is calculated as

$$d_v = t_M - t_N. \tag{16}$$

Step ONE: Apply the maximum filter algorithm in section 3.2 to the TI_S and TI_D waveforms. This step produces *max filtered* TI_S and *max filtered* TI_D.

Step TWO: For every *destination vector* from -*window* before the *guess vector* to +*window* after the *guess vector*, run the minimum standard deviation alignment algorithm in section 3.3. Tally the returned alignments as votes for the true alignment, discarding individual *ambiguous* alignment results that are returned from the minimum standard deviation alignment algorithm. The alignment which received the most votes is called the *best alignment*.

There are three conditions under which the *best alignment* from step two is still considered to be *ambiguous*. If any of these three conditions are detected, then the minimum standard deviation results from this step are discarded and step three is performed. The first condition is if the *best alignment* receives less votes than the *voting threshhold* times the 2**window* possible votes. [For low bit rate HRCs, the recommended setting for *voting threshold* is 0.20 or 20%.] The second condition is if the *best alignment* was +*uncertainty* or -*uncertainty*, which is beyond the range of an acceptable alignment. The third condition is if plausible alternate alignments (other than *best alignment*) are detected. An alternate alignment is detected if it received more than *comparable votes* times the number of votes that the *best alignment* received and if it is *far away* from the b*est alignment*. [For low bit rate HRCs, the recommended setting for *comparable votes* is 0.5 and the recommended setting for *far away* is 5 video frames.]

Step THREE: Take the square root of *max filtered* TI_S and *max filtered* TI_D. Then, for every *destination vector* starting at -*window* before the *guess vector* to +*window* after the *guess vector*, run the minimum standard deviation alignment algorithm in section 3.3. Tally the returned alignments as votes for the true alignment, discarding *ambiguous* alignment results. Return the alignment that received the most votes as the *best alignment*. However, if the *best alignment* is +*uncertainty* or -*uncertainty*, then alignment remains *ambiguous*.

3.4.1 Pseudo Code

Step ONE

```
max_filter (source, destination,
            scene_width + 2*window + 2*uncertainty + (filter_width - 1),
            scene_width + 2*window + 2*uncertainty + (filter_width - 1),
            fraction_above_threshhold, filter_width)
```

Step TWO

/* Initialize the votes for each alignment to zero */

```
offset = - uncertainty
WHILE (offset ≤ uncertainty) DO
BEGIN
      vote[offset] = 0
      offset = offset + 1
END
```

/* Compute alignment for different *destination vectors*; tally results*/

```
loop = - window
WHILE (loop ≤ window) DO
BEGIN
      offset = std_align (source, destination, filter_width, uncertainty, window, scene_width,
                          loop)
      IF (offset ≠ AMBIGUOUS) THEN
            vote[offset] = vote[offset] + 1
      loop = loop + 1
END
```

/* Find the best alignment */

```
best = - uncertainty
loop = - uncertainty
WHILE (loop ≤ uncertainty) DO
BEGIN
      IF (vote[loop] > vote[best]) THEN
            best = loop
      loop = loop + 1
END
```

/* alternate alignment exists if second best has *comparable votes* and is *far away* */

```
loop = - uncertainty
alternate_exists = FALSE
WHILE (loop ≤ uncertainty) DO
BEGIN
      IF (vote[loop] / vote[best] ≥ 0.5 AND |loop-best| > 5) THEN
            alternate_exists = TRUE
END
```

/* Stop if no alternatives exist & the best alignment received *voting threshhold* votes */

```
IF ((alternate_exists ≡ FALSE) AND (vote[best] > 0.20 * (2*window+1))
        AND (best ≠ uncertainty) AND (best ≠ -uncertainty)) THEN
                RETURN (best)
```

Step THREE

/* Initialize the votes for each alignment to zero */

```
offset = - uncertainty
WHILE (offset ≤ uncertainty) DO
BEGIN
        vote[offset] = 0
        offset = offset + 1
END
```

/* Apply square root function to *max filtered* TI_S and *max filtered* TI_D */

```
loop = 0
WHILE (loop < scene_width + 2*window + 2*uncertainty + (filter_width-1)) DO
BEGIN
        source[loop] = sqrt (source[loop])
        destination[loop] = sqrt (destination[loop])
        loop = loop + 1
END
```

/* Compute alignment for different *destination vectors*; tally results*/

```
loop = - window
WHILE (loop ≤ window) DO
BEGIN
        offset = std_align (source, destination, filter_width, uncertainty, window, scene_width,
                                        loop)
        IF (offset ≠ AMBIGUOUS) THEN
                vote[offset] = vote[offset] + 1
        loop = loop + 1
END
```

/* Find the best alignment */

```
best = - uncertainty
loop = - uncertainty
WHILE (loop ≤ uncertainty) DO
BEGIN
        IF (vote[loop] > vote[best]) THEN
                best = loop
        loop = loop + 1
END
IF (best ≠ uncertainty) AND (best ≠ -uncertainty)) THEN
                RETURN (best)
ELSE            RETURN(AMBIGUOUS)
```

4 Methods for Computing Video Quality Parameters

The time histories of the source features ($SI_S(t_n)$, $TI_S(t_n)$) and destination features ($SI_D(t_m)$, $TI_D(t_m)$), are first time aligned, either manually or by the automated method presented in section 3. This time alignment will determine the video delay (d_v) between the destination features and the source features. Hence, for all parameters computed in this section, the time indices on the destination features (t_m) are set equal to $t_n + d_v$, i.e.

$$t_m = t_n + d_v . \tag{17}$$

In describing the video quality parameters, the subscript $_{time}$ will be used extensively. When a function is subscripted by time (such as the root mean square function, denoted $rms_{time}(\bullet)$), this means that the function is applied using the time history of the argument values. This time history will include that portion of the scene-HRC combination that is subjectively viewed (e.g., for the TI_S and TI_D values, this is given by the *reference vector* and the corresponding time aligned *destination vector,* see section 3).

For quality parameters that are based on the time histories of TI_S and TI_D (parameters P_1 to P_6, P_{10}, and P_{11} described below), the preferred component of TI to use is the root mean square component ($TIrms_S$ and $TIrms_D$, measured as given in section 2.3). The computationally efficient mean component of TI ($TImean_S$ and $TImean_D$, measured as given in section 2.4) may also be used. However, use of the mean component of TI could result in some degradation in performance. Similarly, for quality parameters that are based on the time histories of SI_S and SI_D (parameters P_7 to P_9), the preferred method of measurement is given in section 2.1 while a computationally efficient alternative is given in section 2.2.

All of the parameters have been designed to measure perceived impairments. Therefore, the values of the parameters range from zero (for no impairment) to some positive number.

4.1 P_1: Max of TI Ratio

One TI comparison function that has proven useful is the log10 ratio of the quantity TI_D divided by TI_S. The instantaneous value of TI Ratio (t_n) is defined as

$$TI\ ratio\ (t_n) = \log 10\left(\frac{TI_D\,(t_n + d_v)}{TI_S\,(t_n)}\right). \tag{18}$$

When TI ratio (t_n) is positive, there is more temporal information in the destination than the source (i.e., $TI_D(t_n+d_v) > TI_S(t_n)$). This condition can result from impairments such as added noise, jerky motion, and error blocks. When TI ratio (t_n) is negative, there is less temporal information in the destination than the source (i.e., $TI_D(t_n+d_v) < TI_S(t_n)$). This condition results when the HRC is repeating frames (see section 4.10).

Parameter P_1 measures the maximum amount of added temporal information, where the maximum is computed over the time history of the video scene. Prior contributions have described versions of this parameter as MAFNLR (T1A1.5/93-60), and m_3 (T1A1.5/93-32, T1A1.5/92-112). Parameter P_1 is given by

$$P_1 = max\,[max_{time}\,(TI\ ratio\ (t_n)\,)\,, 0]\ . \tag{19}$$

4.2 P_2: Root Mean Square (RMS) of TI Ratio

The root mean square (rms) of TI ratio (t_n) provides an average measure of how different the motion in the destination video is from the source video. Parameter P_2 is given by

$$P_2 = rms_{time}\,[TI\,ratio\,(t_n)]\,. \tag{20}$$

4.3 P_3: Max - Min of TI Ratio

Parameter P_3 is similar to P_1, except that lost motion energy is also considered. Motion energy is lost when the HRC performs frame repetition (see section 4.10). A version of this parameter was described in a paper presented at the 1993 IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (Stephen Voran and Stephen Wolf, "An Objective Technique for Assessing Video Impairments"). Parameter P_3 is given by

$$P_3 = max\,[max_{time}(TI\,ratio\,(t_n)), 0] - min\,[min_{time}(TI\,ratio\,(t_n)), 0]\,. \tag{21}$$

4.4 P_4: Positive Mean - Negative Mean of TI Ratio

This parameter is a form of parameter P_3 that is not as sensitive to the TI ratio (t_n) extrema. Parameter P_4 is given by

$$P_4 = posmean_{time}\,[TI\,ratio\,(t_n)] - negmean_{time}\,[TI\,ratio\,(t_n)]\,, \tag{22}$$

where posmean is the mean computed as the sum of the positive values of TI ratio (t_n) divided by the number of positive values, and negmean is the mean computed as the sum of the negative values of TI ratio (t_n) divided by the number of negative values. If there are no positive values, posmean = 0 and if there are no negative values, negmean = 0.

4.5 P_5: Root Mean Square (RMS) of TI Error Ratio

In addition to TI ratio (t_n), TI error ratio (t_n) is another comparison function that has proven useful. TI error ratio (t_n) is the ratio of the error in TI between the source and destination, normalized by the source. The instantaneous value of TI error ratio (t_n) is defined as:

$$TI\,error\,ratio\,(t_n) = \frac{TI_S(t_n) - TI_D(t_n + d_v)}{TI_S(t_n)} \tag{23}$$

Parameter P_5 is similar to parameter P_2, except the root mean square (rms) of TI error ratio (t_n) is computed. Parameter P_5 is given by

$$P_5 = rms_{time}\,[TI\,error\,ratio\,(t_n)]\,. \tag{24}$$

4.6 P_6: Root Mean Square (RMS) of Positive Part of TI Error Ratio (Lost Motion Energy)

TI error ratio (t_n) is positive when the destination video has lost temporal information, such as during frame repetition by the HRC. Thus, the root mean square (rms) of the positive part of TI error ratio (t_n) provides a measure of how much frame repetition is being performed by the HRC. This parameter P_6 is given by

$$P_6 = rms_{time}\,[max\,(TI\,error\,ratio\,(t_n), 0)]\,, \tag{25}$$

where max (TI error ratio (t_n), 0) performs a frame by frame maximum of TI error ratio (t_n) with the zero value. Parameter P_6 contains information that is similar to parameter m_2 presented in prior contributions (T1A1.5/93-32, T1A1.5/92-112).

4.7 P_7: Max Absolute Value of SI Error Ratio

One SI comparison function that has proven useful is SI error ratio (t_n), defined as the ratio of the error in SI between the source and destination, normalized by the source. The instantaneous value of SI error ratio (t_n) is defined as

$$SI\ error\ ratio\ (t_n) = \frac{SI_S(t_n) - SI_D(t_n + d_v)}{SI_S(t_n)}. \tag{26}$$

When SI error ratio (t_n) is positive, there is less spatial information in the destination than the source (i.e., $SI_D(t_n+d_v) < SI_S(t_n)$). This condition can result from impairments such as blurring. When SI error ratio (t_n) is negative, there is more spatial information in the destination than the source (i.e., $SI_D(t_n+d_v) > SI_S(t_n)$). This condition can result from impairments such as added noise, edges, or blocks.

Parameter P_7 measures the maximum absolute value of SI error ratio (t_n). Parameter P_7 is identical to parameter m_1 in prior contribution T1Q1.5/92-112. Parameter P_7 is given by

$$P_7 = max_{time}(|SI\ error\ ratio\ (t_n)|). \tag{27}$$

4.8 P_8: Root Mean Square (RMS) of SI Error Ratio

The root mean square (rms) of SI error ratio (t_n) provides an average measure of the difference in spatial information between the source and destination video. Parameter P_8 is given by

$$P_8 = rms_{time}[SI\ error\ ratio\ (t_n)]. \tag{28}$$

Parameter P_8 is identical, except for a scaling constant, to parameter m_1 presented in prior contributions (T1A1.5/93-32, T1A1.5/92-112).

4.9 P_9: Alternate form of P_8

Parameter P_9 is a parameter that has been shown to have similar information to parameter P_8. Parameter P_9 is identical, except for a scaling constant, to parameter m_1' presented in prior contribution (T1A1.5/92-135) and parameter AFCEE presented in contribution (T1A1.5/93-60). Parameter P_9 is given by

$$P_9 = \left|\frac{rms_{time}[SI_S(t_n)] - rms_{time}[SI_D(t_n + d_v)]}{rms_{time}[SI_S(t_n)]}\right| \tag{29}$$

4.10 P_{10}: Frame Repeat Rate

The time history of TI_D visually displays frame repeats as low or non-motion samples between motion *spikes* (see Figure 4). The frame repeat rate for a portion of a video scene can be computed by finding two motion *spikes* in that portion of the video scene and then counting the number of low or non-motion samples between these two motion *spikes*. The frame repeat rate, defined as this number of low or non-motion samples plus one (so that one motion *spike* is included) would give a value of 1 for 30 frames per second, a value of 2 for 15 frames per second, etc.

In general, the frame repeat rate for low bit rate HRCs is adaptive and changes with time. Therefore, it is desirable to compute a composite frame repeat rate for a video scene. Prior contribution T1A1.5/92-138 proposed the Average Frame Rate (AFR) for this composite measure. However, research conducted by ITS indicates that a composite frame repeat rate that is at the 75% value (i.e., the point where 75% of the frame repeat rates in the scene are below and 25% are above) yields a quality parameter that more closely correlates with subjective quality. This research further indicates that the logarithm of this 75% value should be used as the quality parameter. This section presents an automated algorithm for computing this 75% frame repeat rate parameter.

4.10.1 Definitions

The following definitions will be required when describing how to compute the composite frame repeat rate (the reader may refer to Figure 4 and Figure 5).

Spike: If the current time sample of TI is greater than both the previous time sample and the following time sample, then a *spike* is said to have occurred at the current time.

Spike Height: The *spike height* is the difference between the current time sample of TI and the larger of the two adjacent samples.

Scene Cut: Any *spike* in TI_S that has a *spike height* greater than 15.0 is called a *scene cut*.

Frame Repeat Spike: A motion *spike* in the TI_D waveform that results from frame repeating by the HRC.

Frame Repeat Delta: The distance between two consecutive *frame repeat spikes*, equal to the number of intervening TI_D samples plus one (see Figure 5).

Source Variation: A threshold used for detecting *frame repeat spikes* in the TI_D waveform. Defined as the maximum *spike height* in the TI_S waveform, excluding any *scene cuts,* multiplied by a *small increase*. The purpose of increasing the threshold is to decrease false detections of *frame repeat spikes* in the TI_D waveform. [The recommended setting for *small increase* is 1.2]

Maximum Repeat Delta:The largest *frame repeat delta* expected. [For the T1A1.5 tests, the recommended setting for *maximum repeat delta* is 60 to accommodate low bit rate HRCs that transmit only one frame every 2 seconds.]

Min Number Deltas: The minimum number of *frame repeat deltas* in the video scene that should be detected if the HRC is repeating frames at the *maximum repeat delta* rate. [For the T1A1.5 tests, the recommended value for *min number deltas* is 4 provided the *maximum repeat delta* is set to 60 since 270/60 = 4, when rounded down to the nearest integer.]

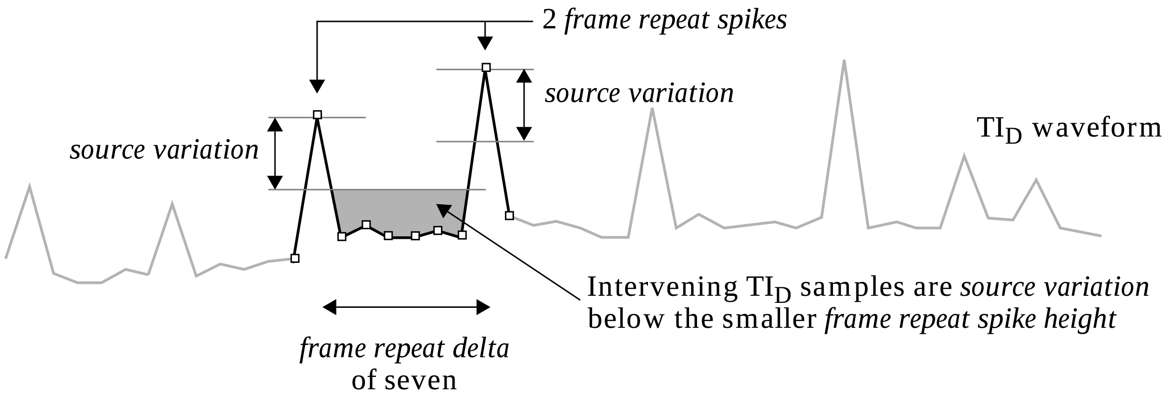
2 *frame repeat spikes*
source variation
source variation
TI_D waveform
Intervening TI_D samples are *source variation* below the smaller *frame repeat spike height*
frame repeat delta of seven

Figure 5 Definitions of Frame Repeat Rate Terms

4.10.2 Frame Repeat Rate Algorithm

The first step in the algorithm is to differentiate between *frame repeat spikes* in the TI_D waveform and *spikes* caused by motion variations in the source scene. Figure 4 contains an example plot of TI_S and TI_D waveforms. Notice that TI_S contains a number of spikes. The largest of these, having a height greater than 15.0, is a *scene cut*. The other *spikes* are motion variations in the source scene. Any *spike* in TI_D of comparable size to these source scene motion variations is not considered a *frame repeat spike*. Therefore, the *source variation* is used as a threshhold for detecting *frame repeat spikes*. The *source variation* is the *spike height* of the largest *spike* in TI_S, excluding *scene cuts*, multiplied by a *small increase*. By increasing the threshold slightly, the chance of falsely detecting *frame repeat spikes* is reduced. [The recommended setting for *small increase* is 1.2, which is a 20% increase in the threshhold.]

As shown in Figure 5, two *frame repeat spikes* in TI_D (herein called *spike* 1 and *spike* 2) are declared if all of the following three conditions are met; (1) *spike* 1 and *spike* 2 have *spike heights* greater than the *source variation*, (2) there are no *spikes* with *spike heights* greater than the *source variation* between *spike* 1 and *spike* 2, and (3) all intervening TI_D samples between *spike* 1 and *spike* 2 have values that are at least *source variation* below the smaller of *spike height* 1 and *spike height* 2. Whenever the above three conditions are met in the TI_D time history, the *frame repeat delta* is computed and saved to an array. A counter keeps track of how many *frame repeat deltas* are saved. The array of *frame repeat deltas* is rank sorted from low to high.

The frame repeat rate is set equal to 1 if (1) the number of *frame repeat deltas* found in the time history of TI_D is less than the number of *scene cuts* in the time history of TI_S plus the *min number deltas* for a video scene of this *scene width*, or if (2) the 75% *frame repeat delta* value of the rank sorted array is above the *maximum repeat delta*. Otherwise, the frame repeat rate is set equal to the 75% *frame repeat delta* value of the rank sorted array.

The log, base ten, of the frame repeat rate is the quality parameter.

4.10.3 Pseudo Code

Function **repeat_rate** computes the log, base 10, of the 75% *frame repeat delta* value of a *destination vector* that is time aligned to the *reference vector*. The 75% *frame repeat delta* value is an integer ranging between one and the *maximum repeat delta*.

Function **repeat_rate** (source, destination, scene_width, maximum_repeat_delta,
min_number_deltas)

source
Contains the *reference vector*. Array elements are numbered from zero to (*scene_width* - 1).

destination
Contains the *destination vector* which best aligns to the *reference vector* in array source. Array elements are numbered zero to (*scene_width* - 1).

scene_width
Same as the *scene width* defined in section 3.1.

maximum_repeat_delta
Same as the *maximum repeat delta* defined in section 4.10.1.

min_number_deltas
Same as the *min number deltas* defined in section 4.10.1.

/* Compute *source variation* using *small increase* = 1.2; tally number of *scene cuts* in TI_S */

```
source_variation = 0.0
scene_cuts = 0
frame = 1
WHILE (frame ≤ scene_width-2) DO
BEGIN
    IF (source[frame+1] > source[frame-1]) THEN
            height = source[frame] - source[frame+1]
    ELSE    height = source[frame] - source[frame-1]
    IF (height > 15.0) THEN
        scene_cuts = scene_cuts + 1
    ELSE IF (height > source_variation) THEN
        source_variation = height
    frame = frame + 1
END
source_variation = source_variation * 1.2
```

/* num_repeats is the number of *frame repeat deltas* found so far */

```
num_repeats = 0
```

/* prev_spike is the array element of the last *frame repeat spike* found. */

```
prev_spike = 0
```

/* Loop through all frames looking for *frame repeat spikes* */

```
frame = 1
WHILE (frame ≤ scene_width-2) DO
BEGIN
```

/* Assume this frame is a *spike* & compute the *spike height*.*/

```
    IF (destination [frame+1] > destination [frame-1])THEN
            height = destination [frame] - destination [frame+1]
    ELSE    height = destination [frame] - destination [frame-1]
    /* Consider spikes with spike heights greater than source variation */

    IF (height > source_variation) THEN
    BEGIN

        /* First time, initialize prev_spike as this frame */

        IF (prev_spike ≡ 0) THEN
            prev_spike = frame

        /* Otherwise, check whether the prev_spike and this spike are frame repeat spikes */

        ELSE BEGIN

            /* Compute threshhold using prev_spike, this spike, & source variation */

            IF (destination[prev_spike] < destination[frame]) THEN
                    threshhold = destination[prev_spike] - source_variation
            ELSE    threshhold = destination[frame] - source_variation
            between = prev_spike + 1
            is_a_repeat = TRUE
            WHILE (between < frame) DO
            BEGIN
                IF (destination[between] > threshhold)
                    is_a_repeat = FALSE
                between = between + 1
            END

            /* Array delta holds the num_repeats frame repeat deltas found. */

            IF (is_a_repeat ≡ TRUE)
            BEGIN
                num_repeats = num_repeats + 1
                delta[num_repeats] = frame - prev_spike
            END

            /* Update the location of the prev_spike. */

            prev_spike = frame
        END
    END
    frame = frame + 1
END

/* Sort array delta from smallest at delta[1] to largest at delta[num_repeats] */

sort (delta)

/* Repeat rate is one if there were too few frame repeat deltas found. Return $\log_{10}(1.0) = 0$ */

IF (num_repeats ≤ scene_cuts + min_number_deltas) THEN
        RETURN (0)
```

```
/* Repeat rate is one if the 75% delta value is unreasonably large. Return $\log_{10}(1.0) = 0$ */

ELSE IF (delta [num_repeats * 0.75] > maximum_repeat_delta)
            RETURN (0)

/* Otherwise, return $\log_{10}$ of the 75% delta value */

ELSE        RETURN (log10 (delta [num_repeats * 0.75]))
```

4.11 P_{11}: Max TI Spike Increase, Masking Scene Cuts

Both TI_D and TI_S contain *spikes*. Some of these are *scene cuts*, some are *frame repeat spikes*, and some result from natural motion variations in the video scene. The increase in non *scene cut spike height* from TI_S to TI_D determines the value of this quality measure. Frames near *scene cuts* are not used because relatively large motion variations can occur within a few frames of a scene cut and be relatively unnoticed due to the masking effects of the eye and brain. [For parameter P_{11} described below, frames from 5 before to 10 after a scene cut are masked.]

To compute P_{11}, first find all *spikes* in the TI_S and TI_D waveforms. Next, discard all TI_S and TI_D *spikes* from five frames before to ten frames after each *scene cut*. This will eliminate most of the *scene cut* effects from the measure. From the remaining *spikes*, find the highest TI_D *spike* and the highest TI_S *spike*. Take the *spike height* of the tallest TI_D *spike* minus the *spike height* of the tallest TI_S *spike*. Set this difference to zero if it falls below zero, and return $\log_{10}$ (difference + 1).

4.11.1 Pseudo Code

Function **spike_increase** computes the $\log_{10}$ of the largest increase in temporal motion spikes between the *reference vector* and the time aligned *destination vector*.

Function **spike_increase** (source, destination, scene_width)

source — Contains the *reference vector*. Array elements are numbered from zero to (*scene_width* - 1).

destination — Contains the *destination vector* which best aligns to the *reference vector* in array source. Array elements are numbered zero to (*scene_width* - 1).

scene_width — Same as the *scene width* defined in section 3.1.

```
BEGIN

/* is_a_scene_cut is a boolean array, valid from 1 to (scene width - 2); marks scene cuts */

frame = 1
WHILE (frame ≤ scene_width - 2) DO
BEGIN
      is_a_scene_cut [frame] = FALSE
      frame = frame + 1
END

/* Find the location of all scene cuts. Extend scene cut mask 5 before, 10 after */

frame = 1
```

```
WHILE (frame ≤ scene_width - 2) DO
BEGIN
    IF (source [frame+1] > source [frame-1]) THEN
            height = source [frame] - source [frame+1]
    ELSE    height = source [frame] - source [frame-1]
    IF (height > 15.0) THEN
    BEGIN
        cnt = frame - 5
        WHILE (cnt ≤ frame + 10) DO
        BEGIN
            IF (1 ≤ cnt AND cnt ≤ scene_width - 2) THEN
                is_a_scene_cut [cnt] = TRUE
            cnt = cnt + 1
        END
    END
    frame = frame + 1
END
```

/* Find maximum *reference vector spike* & maximum *destination vector spike* */

```
max_source_spike = 0.0
max_destination_spike = 0.0
frame = 1
WHILE (frame ≤ scene_width-2) DO
BEGIN
    IF (is_a_scene_cut [frame] ≡ FALSE) THEN
    BEGIN
        IF (source [frame+1] > source [frame-1]) THEN
                height = source [frame] - source [frame+1]
        ELSE    height = source [frame] - source [frame-1]
        IF (height > max_source_spike) THEN
            max_source_spike = height

        IF (destination [frame+1] > destination [frame-1]) THEN
                height = destination [frame] - destination [frame+1]
        ELSE    height = destination [frame] - destination [frame-1]
        IF (height > max_destination_spike) THEN
            max_destination_spike = height
    END
    frame = frame + 1
END
```

/* Return $\log_{10}$ of the spike difference plus one; minimum zero. */

```
IF (max_destination_spike > max_source_spike) THEN
        RETURN (log10 (max_destination_spike - max_source_spike + 1.0))
ELSE    RETURN (0.0)
```

5 Extensions to the Video Quality Parameters

Several extensions have been made to the basic ITS video quality parameters. These extensions, described in this section, seem to provide significant additional information about HRC performance. However, we are not at this time recommending that these extensions be included in the first round of objective performance parameter testing. The extensions are described in this contribution to inform committee T1A1.5 of ongoing video quality research efforts at ITS.

5.1 Scene Cut Masking

A number of psychological studies have been performed that demonstrate that the eye and brain have trouble perceiving scene detail immediately before and after a scene cut (for instance, see William E. Glenn and Karen G. Glenn, “High Definition Television Compatible Transmission System,” IEEE Transactions on Broadcasting, Vol. BC-33, No. 4, December, 1987). This can be accommodated in the video quality measurements by introducing scene masking, where the objective video quality measurements for the video frames around a scene cut are weighted less than the objective video quality measurements for other video frames that are not around a scene cut. This technique was, in fact, used for parameter P_{11} in section 4.11.

5.2 Film Generated NTSC Video

The 3-2 pulldown process used to convert 24 frames per second (fps) film to 30 fps NTSC video causes every 5th field of NTSC video to have zero motion. For film generated NTSC video, this is readily observed by plotting the time history of TImean as measured in section 2.4. In some cases, it may be desirable to pre-filter the TI_S and TI_D waveforms to remove the effects of 3-2 pulldown before the objective video quality parameters in section 4 are computed. This is an issue for further study.

5.3 Sub-regional Video Quality Measurements

Research at ITS has revealed that additional information can be obtained about the HRC performance if the objective video quality parameters are measured using sub-regions that are smaller than the *viewable region* (in Figure 3). Such was the case for the MALNLR parameter presented in T1A1.5/93-60. There are three types of sub-regions that provide additional useful information. They are:

1. Minimum motion sub-region - the video quality parameters in section 4 are measured using the minimum motion sub-region. This provides a measurement of impairments that are more noticable on the stationary portion of the video scene (e.g., noise, error blocks).
2. Maximum motion sub-region - the video quality parameters in section 4 are measured using the maximum motion sub-region. This provides a measurement of impairments that are more noticable on the moving portion of the video scene (e.g., blurring of moving lips, eyes, or head).
3. Maximum spatial detail sub-region - the video quality parameters in section 4 are measured using the maximum spatial detail sub-region. This provides a measurement of impairments that are more noticable on the spatially detailed portion of the video scene, regardless of whether this detailed portion is moving or not (e.g., blurring of details on a map).

To see how the above three types of parameters are computed, a simple example will be provided. This example is shown in Figure 6. Here the *viewable region* of Figure 3 has been sub-divided into 6 smaller sub-regions. These are denoted 1 through 6 in the figure. For each sub-region shown in the figure, compute the temporal information and spatial information measurements that have been presented in this contribution. These will be denoted $TI_{Sj}(t_n)$, $TI_{Dj}(t_n+d_v)$, $SI_{Sj}(t_n)$, and $SI_{Dj}(t_n+d_v)$, where j denotes the sub-region number (j = 1 through 6) and the rest of the notation is as previously defined. The algorithms for finding the above three sub-regions as a function of time t_n are then simply

1. Minimum motion sub-region - the sub-region j such that $TI_{Sj}(t_n) < TI_{Sk}(t_n)$, for all $k \neq j$.
2. Maximum motion sub-region - the sub-region j such that $TI_{Sj}(t_n) > TI_{Sk}(t_n)$, for all $k \neq j$.
3. Maximum spatial detail sub-region - the sub-region j such that $SI_{Sj}(t_n) > SI_{Sk}(t_n)$, for all $k \neq j$.

The video quality parameters of section 4 can then be computed using the above time varying sub-regions of the source and destination video frames.

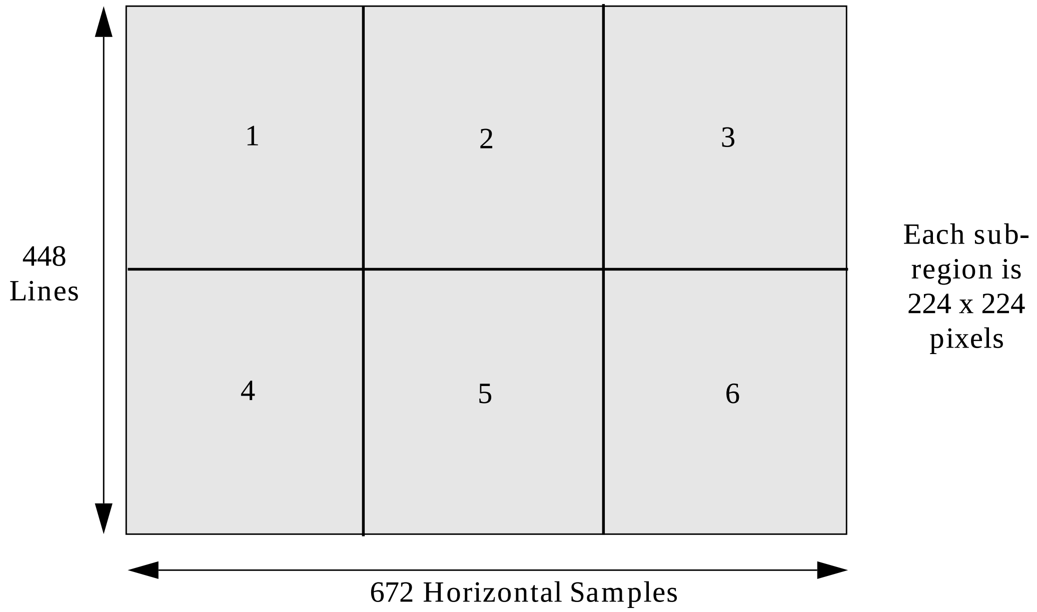

Figure 6 Example Showing how the Viewable Region can be Sub-divided