COMMITTEE T1

CONTRIBUTION

Document Number: T1A1.5/94-110

****

STANDARDS PROJECT: VTC/VT Performance Standards Projects

****

TITLE: Corrections and Extensions to T1A1.5/93-152

****

ISSUE ADDRESSED: Objective Video Performance Testing

****

SOURCE: NTIA/ITS, Stephen Wolf, Margaret Pinson

****

DATE: January 17, 1994

****

DISTRIBUTION TO: T1A1.5

****

KEYWORDS: Objective Video Performance Testing, Video Quality Parameters

****

DISCLAIMER:

****

1 Introduction

At the last T1A1.5 meeting (November, 1993 in San Jose, CA), contribution T1A1.5/93-152 was presented which summarized the methods of measurement for objective video quality parameters based on the sobel-filtered image and the motion difference image. This contribution presents:

1. One minor correction to the recommended value for the *fraction above threshold* in contribution T1A1.5/93-152.

2. A method for estimating the video delay uncertainty of the automated time alignment algorithm presented in section 3 of contribution T1A1.5/93-152. Non-zero video delay uncertainty may result when (1) dynamic time warping, or variable video delay, is present in the Hypothetical Reference Circuit (HRC), or (2) a substantial number of video frames are dropped by the HRC.

3. A method for using this video delay uncertainty in the computation of the parameters presented in T1A1.5/93-152.

4. An improved motion *spike* detector that could be used for computing parameters p_{10} and p_{11} in T1A1.5/93-152.

The correction to contribution T1A1.5/93-152 given by item 1 above should be implemented by all laboratories that are making the video quality measurements. The extensions to contribution T1A1.5/93-152 given by items 2 through 4 above should be included in the evaluation process that is being used by committee T1A1.5 to validate the objective video quality measurements. Questions regarding the material 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

2 Correction to T1A1.5/93-152

The recommended value for the *fraction above threshold* in contribution T1A1.5/93-152 was miscomputed by a factor of approximately 0.6. Thus, the last sentence on page 9 of contribution T1A1.5/93-152 should read

"[The recommended value for the *fraction above threshold* is 0.42]"

and the second to the last sentence in the last full paragraph on page 11 should read

"To assure this condition is met, the recommended value for the *fraction above threshold* is 0.42."

3 Video Delay Uncertainty

For low bit rate HRCs, the video delay of the HRC may change dynamically depending upon the information content of the source video. Typically, larger amounts of motion in the

source video produce longer coding and transmission delays, and hence longer video delays. This effect is illustrated in Figure 1, where the time history of the Temporal Information of the Source video (TI_S) is plotted against the time history of the Temporal Information of the Destination video (TI_D). By examining the figure, one can see that the TI_S and TI_D waveforms are time aligned during the low motion portion of the scene (from time 50 to time 120). However, the TI_S and TI_D waveforms are not time aligned for the high motion portion of the scene (from time 10 to time 40). During the high motion part, TI_D lags TI_S by up to 10 more frames. Thus, the video delay uncertainty for the 4 second piece of source video shown in Figure 1 is about 10 frames.

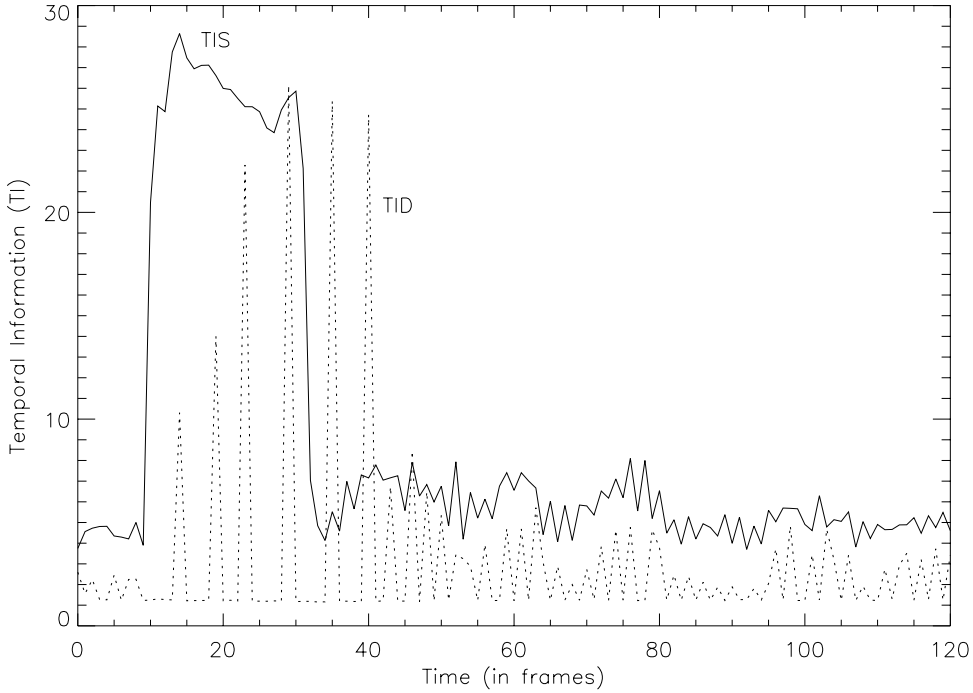

Figure 1 Plot Demonstrating Variable Video Delay

Another cause of non-zero video delay uncertainty is when the HRC drops so many source video frames that the temporal information is greatly reduced. This condition typically occurs for very high motion scenes that have been transmitted through low bit-rate HRCs. One example is plotted in Figure 2. By observing and comparing TI_D to TI_S, one notes that the HRC is only transmitting about one frame out of every 15 to 30 source video frames. Thus, the video delay uncertainty for the 10 second piece of source video shown in Figure 2 is on the order of 30 frames.

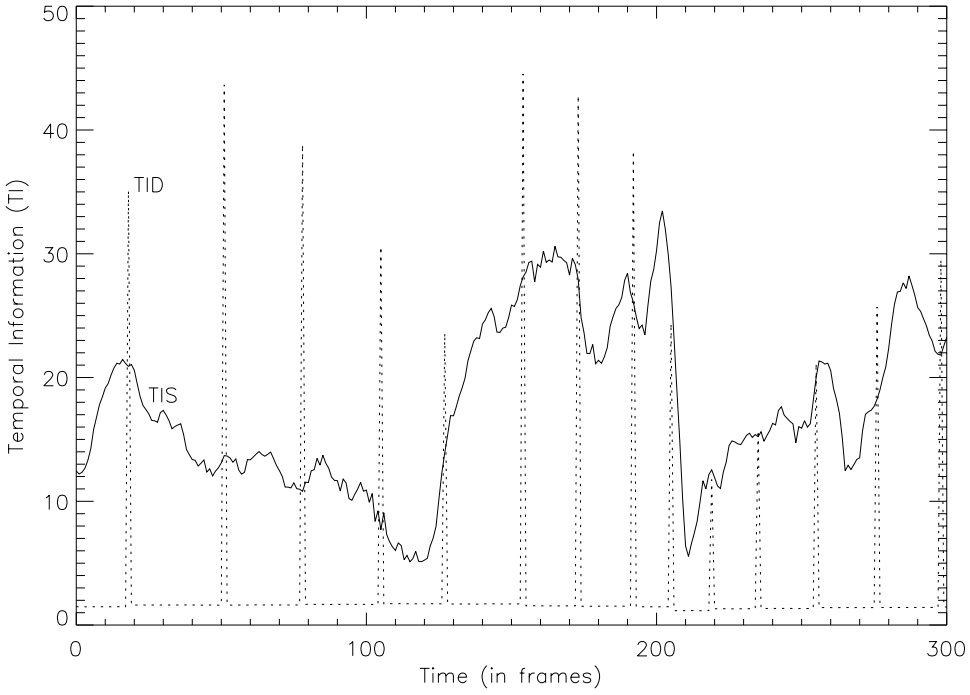

Figure 2 Plot Demonstrating Low Frame Rate HRC

One could visually inspect the TI_S and TI_D time histories such as those given in Figure 1 and Figure 2 to determine the video delay uncertainty. However, since video delay is a statistical quantity that changes dynamically, it would be preferable to have an automated algorithm for estimating this video delay uncertainty. This remainder of this section presents the current research and methodology for automating the estimation of video delay uncertainty.

The minimum standard deviation alignment presented in section 3.3 of T1A1.5/93-152 calculates the video delay of one piece of destination video (this piece of destination video is called a *destination vector*). The alignment algorithm of section 3.4 of T1A1.5/93-152 computes the video delays of many different pieces of destination video and forms a histogram of the results. This is performed in step two (or step three) of the alignment algorithm (see page 16 of T1A1.5/93-152). This histogram tells how many votes each possible video delay received. In T1A1.5/93-152 the *best alignment* (i.e., the best estimate of the video delay), is chosen as that video delay which received the most votes. However, if one considers a range of video delays (R) that includes every video delay (d_v) in the histogram which received at least one vote, i.e.

$$R = [d_{vmin}, d_{vmax}] \quad (1)$$

where d_{vmin} is the minimum video delay with at least one vote and d_{vmax} is the maximum video delay with at least one vote. This range R provides an estimate of the video delay uncertainty.

4 Parameter Computation Using Video Delay Uncertainty

In addition to computing the video quality parameters for that time alignment which received the most votes (i.e., the *best alignment* in T1A1.5/93-152), one could compute the video

quality parameters for every time alignment within the range R specified by equation (1) above. If this is done for some selected piece of source video (for the T1A1.5 video teleconferencing tests, the selected piece of source video is 9 seconds long), a quality parameter (p) could also be assigned to a range of values. The question then becomes how to best compute a single number from this range of values. One possibility would be to take the minimum over the range of values. If $p(d_v)$ is the parameter value for a video delay of d_v, then we could define a p', such that

$$p' = \min_{d_v \in R} [p(d_v)] . \quad (2)$$

Since the parameters measure impairment, this criterion would select the minimum impairment over the range of video delay uncertainty. Improvements have been observed in the performance of the video quality parameters that are most sensitive to time alignment shifts. In particular, this is true of parameters p_1, p_3, and p_7 in contribution T1A1.5/93-152 since they apply maximum and/or minimum time collapsing functions to the frame by frame parameter samples.

5 Improved Spike Detector

Parameters p_{10} and p_{11} in contribution T1A1.5/93-152 utilize a *spike* detector to detect *spikes* of motion energy in the TI_D and TI_S waveforms. These motion *spikes* occur during scene cuts and when jerky motion is present in the HRC output video. The *spike* detector described in T1A1.5/93-152 works well for detecting *spikes* that are one TI sample wide. However, if a scene cut or an HRC frame update occurs halfway through the NTSC video frame, or if an HRC takes two frames to update the video, then a two TI sample width *spike* detector is required to properly detect these motion *spikes*. [Note: Scene cuts or HRC frame updating halfway through the NTSC video frame will create two sample wide spikes in the TI metrics based on frame differences presented in section 2.3 of T1A1.5/93-152. These same motion conditions will only create one sample wide spikes in the TI metrics based on field differences presented in section 2.4 of T1A1.5/93-152.]

This section presents pseudo code for an improved *spike* detector that measures the *spike height* of a motion *spike* that is one or two TI samples wide. Parameters p_{10} and p_{11} in contribution T1A1.5/93-152 could be modified to utilize this improved *spike* detector. These modifications would involve not only the spike detection logic of parameters p_{10} and p_{11} but also the frame repeat detection and counting logic of parameter p_{10}. Function **spike_height** returns the *spike height* of a TI sample point within the TI array.

Function **spike_height** (array, frame)

array	An array that contains the time history of the TI samples. At least two points prior to this frame and two points after this frame must be present in the array.
frame	An integer that specifies the frame within the TI array where the *spike height* is to be measured.

/* Check to see if this frame is a *spike*; if not, return a *spike height* of 0.0 */

```
IF ((array[frame] < array[frame+1]) OR (array[frame] < array[frame-1])) THEN
    RETURN (spike_height = 0.0)
```

/* Left leg of *spike* (earlier in time) has less height than the right leg */

```
IF (array[frame-1] > array[frame+1]) THEN
BEGIN
```

/* Left leg slopes down into a 2 wide *spike* */

```
    IF (array[frame-2] < array[frame-1]) THEN
    BEGIN
        IF (array[frame-2] > array[frame+1]) THEN
            spike_height = array[frame] - array[frame-2]
        ELSE
            spike_height = array[frame] - array[frame+1]
    END
```

/* Left leg shorter, 1 wide *spike* */

```
    ELSE
        spike_height = array[frame] - array[frame-1]
END
```

/* Right leg of *spike* (later in time) has less height than the left leg */

```
ELSE BEGIN
```

/* Right leg slopes down into a 2 wide *spike* */

```
    IF (array[frame+2] < array[frame+1]) THEN
    BEGIN
        IF (array[frame+2] > array[frame-1]) THEN
            spike_height = array[frame] - array[frame+2]
        ELSE
            spike_height = array[frame] - array[frame-1]
    END
```

/* Right leg shorter, 1 wide *spike* */

```
    ELSE
        spike_height = array[frame] - array[frame+1]
END

RETURN (spike_height)
```