NTIA Report TR-06-433

Reduced Reference Video Calibration Algorithms

Margaret H. Pinson
Stephen Wolf

report series

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

NTIA Report TR-06-433

Reduced Reference Video Calibration Algorithms

Margaret H. Pinson
Stephen Wolf

U.S. DEPARTMENT OF COMMERCE
Carlos M. Gutierrez, Secretary

Michael D. Gallagher, Assistant Secretary
for Communications and Information

October 2005

DISCLAIMER

Certain commercial equipment and materials are identified in this report to specify adequately the technical aspects of the reported results. In no case does such identification imply recommendations or endorsement by the National Telecommunications and Information Administration (NTIA), nor does it imply that the material or equipment identified is the best available for this purpose.

The software algorithms described within were developed by an agency of the U.S. Government. No warranty, expressed or implied, is made by NTIA or the U.S. Government as to the accuracy, suitability, and functioning of the programs and related material, nor shall the fact of distribution constitute any endorsement by the U.S. Government.

CONTENTS

Page

FIGURES

Page

TABLES

Page

REDUCED REFERENCE VIDEO CALIBRATION ALGORITHMS

Margaret H. Pinson and Stephen Wolf [1]

This report describes four Reduced Reference (RR) video calibration algorithms of low computational complexity. RR methods are useful for performing end-to-end in-service video quality measurements since these methods utilize a low bandwidth network connection between the original (source) and processed (destination) ends. The first RR video calibration algorithm computes temporal registration of the processed video stream with respect to the original video stream (i.e., video delay estimation). The second algorithm jointly calculates spatial scaling and spatial shift. The third algorithm calculates luminance gain level offset of the processed video stream with respect to the original video stream. The fourth algorithm estimates the valid video region of the original or processed video stream (i.e., the portion of the video image that contains actual picture content). All the algorithms utilize only the luminance (Y) image plane of the video signal.

Key words: calibration; delay; gain; offset; spatial scaling; spatial shift; temporal shift; video

1 INTRODUCTION

This report describes a series of low computational complexity Reduced Reference (RR) video calibration algorithms. RR measurements are useful for in-service quality monitoring applications since they require only a small amount of reference information to make a performance measurement [1]. This RR information can thus be easily communicated between the original (i.e., source) and processed (i.e., destination) ends of a video system using commonly available network connections (e.g., Internet, Public Switched Telephone Network). In an RR measurement system, video calibration of the processed video stream (i.e., estimating gain, offset, temporal delay, spatial shift, spatial scaling, and valid region) is normally a prerequisite for estimating the quality of the processed video stream. While the algorithms described in this document can produce calibration estimates using a single video clip, these algorithms are most effective when applied in the order specified to a series of video clips associated with a single video system. Combining calibration results from multiple video clip samples provides a means to obtain more robust estimates.

The first algorithm estimates the overall temporal alignment (i.e., delay) of the processed video stream with respect to the original video stream. This temporal delay estimation algorithm utilizes three features that characterize the motion and luminance of the video scene. The second algorithm jointly estimates spatial scaling and spatial shift. This second algorithm uses randomly

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

selected pixels and image profiles (horizontal and vertical) that are extracted from the video scene. The third algorithm reuses these pixels and profiles to estimate luminance gain and level offset. The fourth algorithm examines the edges of images and estimates the portion of the video image (original or processed) that contains actual valid picture content. All algorithms utilize the luminance (Y) image plane of the video signal. Luminance values are presumed to range from 1 to 254 (e.g., black = 16, white = 235).

2 TEMPORAL REGISTRATION ALGORITHM

The temporal registration algorithm presented in this section is an RR method that requires a data transmission bit-rate of 1 to 6 kb/s (depending upon video frame rate and feature quantization accuracy[2]). This low bit-rate requirement, together with the computational simplicity of extracting the RR features, makes this method ideally suited for real-time in-service monitoring of end-to-end video delay. Three low bandwidth features track 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. This procedure produces one video delay estimate per video clip, which should have a minimal duration of 5 seconds.[3]

2.1 Default Valid Video Region

NTSC (525-line) and PAL (625-line) video sampled according to ITU-R Recommendation BT.601 [2] (henceforth abbreviated Rec. 601) may have a border of pixels and lines that do not contain valid picture. The original video from the camera may only fill a portion of the Rec. 601 frame. Some digital video compression schemes further reduce the area of the picture in order to save transmission bits. To prevent non-picture areas from degrading the performance of the temporal registration algorithm, they must be excluded.

Section 5 presents an automated method for determining valid region. However, since valid region is always referenced to the original video, we first need to know the spatial shift and scaling that is present in the processed video. Since spatial shift and scaling information is only available after the spatial registration algorithm is completed, and since temporal registration must be known before spatial registration, we must use some reasonable default values for valid region. Table 1 gives the reasonable default values that we use for the border of invalid pixels around the edge of common image sizes. Pixels in this invalid border region will be discarded by the temporal registration algorithm. Images in common intermediate format (CIF), source input format (SIF), and quarter resolution versions of these (QCIF and QSIF) typically do not have an invalid border, so no pixels are discarded.

Table 1. Default Invalid Border for Common Video Sizes

Video Type	Rows	Columns	Invalid Top	Invalid Left	Invalid Bottom	Invalid Right
NTSC (525-line)	486	720	18	22	18	22
PAL (625-line)	576	720	14	22	14	22

[2] 6 kb/s presumes interlaced video operating at 60 fields per second and single precision to code the features.
[3] The method presented here estimates the average video delay of the entire processed sequence, as opposed to methods that estimate the video delays of individual frames within the processed sequence. When individual frames within the processed sequence have different video delays, they will tend to vary about the average video delay found by this algorithm.

2.2 Description of Features

The temporal alignment algorithm uses three low-bandwidth features. The first temporal information (TI) feature, designated TI2, uses temporal differences between adjacent luminance images. For interlace video, fields spaced 2 fields apart in time (i.e., 1 frame apart) are differenced, and for progressive video, frames spaced one frame apart in time are differenced. This TI feature quantifies the amount of scene motion by summing the squared pixel differences between two sampled images. The second feature, designated TI10, is based on temporal differences spaced 10 interlaced fields or 5 progressive frames apart. 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. The second feature smoothes over these times of no motion in the processed video (i.e., repeated frames or fields). The third feature, designated Ymean, is based on the mean of the Y images. This feature quantifies the intensity, or brightness, of the video scene by measuring the average luminance level.

Each of the three 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 produce good temporal alignments where another one fails. The magnitude and shape of the correlation function are used as indicators for the reliability of that feature's temporal registration. Unreliable features are discarded, and the remaining results are considered jointly to estimate the best average video delay for the clip.

In the following feature definitions, the luminance image is noted as Y. For interlaced video, Y is a field and for progressive video, Y is a frame. The time when this field or frame 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 $\mathrm{Y}(i,j,t)$.

2.2.1 TI2 Feature: Two Field Difference Temporal Information

For interlaced video, 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 for each pixel

$$\mathrm{TI2}(i,j,t) = \mathrm{Y}(i,j,t) - \mathrm{Y}(i,j,t-2)) \tag{1}$$

For progressive video, consider frame $\mathbf{Y}(t)$ and the previous frame $\mathbf{Y}(t\text{-}1)$, and compute for each pixel

$$\mathrm{TI2}(i,j,t) = \mathrm{Y}(i,j,t) - \mathrm{Y}(i,j,t-1)) \tag{2}$$

Then, using the results from (1) or (2), compute

$$\mathrm{TI2}(t) = rms_{\mathrm{space}}[\mathrm{TI2}(i,j,t)] \equiv \sqrt{\frac{1}{\mathrm{R}}\sum_i\sum_j (\mathrm{TI2}(i,j,t))^2} \tag{3}$$

where rms_{space} is the root mean square function over space defined by the above equation, i and j are within the valid region defined in section 2.1, and R is the total number of pixels in the valid region (i.e., the number of pixels in the double summation). The TI2 calculation for interlaced fields is calculated as shown in Figure 1.

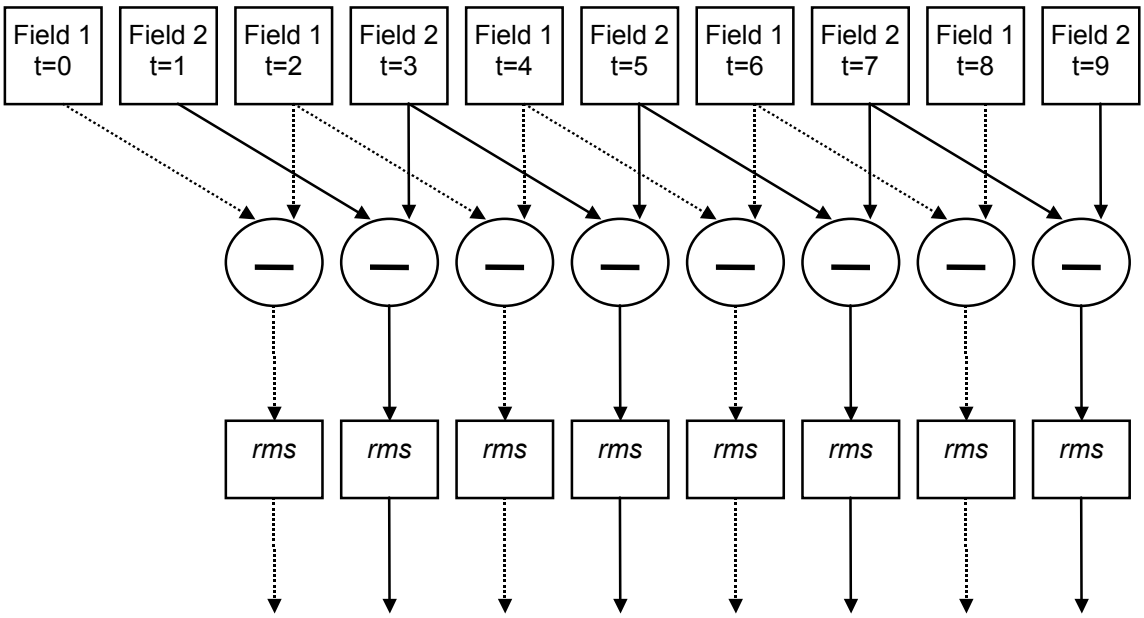

Figure 1. Diagram depicting the method of calculating TI2(t).

2.2.2 TI10 Feature: Ten Field Difference Temporal Information

The TI10 feature is based on a temporal difference spaced ten interlaced fields apart, or five progressive frames apart. This feature smoothes the temporal information using a wider filter than TI2 and eliminates the effect of frame repeats in the TI waveform for systems that have four or fewer consecutive frame repeats.

To compute TI10 at time t on interlaced sequences, consider field Y(t) and the field of the same type five frames ago Y(t-10), and compute for each pixel

$$TI10(i,j,t) = Y(i,j,t) - Y(i,j,t\text{-}10) \quad (4)$$

For progressive sequences, consider frame Y(t) and the frame five frames ago Y(t-5), and compute for each pixel

$$TI10(i,j,t) = Y(i,j,t) - Y(i,j,t\text{-}5) \quad (5)$$

Then, using the results from (4) or (5), compute

$$\mathrm{TI10}(t) = rms_{\mathrm{space}}[\mathrm{TI10}(i,j,t)] \tag{6}$$

where i and j are within the valid region defined in section 2.1.

2.2.3 Ymean Feature: Average Luminance Level

Ymean is calculated as the average luminance level of a field. To compute Ymean at time t on interlaced fields or frames, consider $\mathrm{Y}(t)$ and compute

$$\mathrm{Ymean}(t) = mean_{\mathrm{space}}[\mathrm{Y}(i,j,t)] \equiv \frac{1}{\mathrm{R}} \sum_i \sum_j Y(i,j,t) \tag{7}$$

where $mean_{\mathrm{space}}$ is the mean function over space defined by the above equation, i and j are within the valid region defined in section 2.1, and R is the total number of pixels in this valid region (i.e., the number of pixels in the double summation).

2.3 Feature Sequence Cross-Correlation[4] and Validation

The following steps describe the process that is used to cross-correlate original and processed feature streams in order to estimate their best temporal registration. The algorithm includes validation steps, where the features are examined to discard potentially unreliable alignment results. 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 feature, $a(t) = \mathrm{TI2}(t)$. Let M be the length of the feature streams, a_o and a_p. The feature stream a_o, given by $\{a_o(0), a_o(1), a_o(2), \ldots, a_o(M-1)\}$ will be abbreviated as $\{a_o(t)\}_{t=0}^{M-1}$.

We have found that temporal registration estimates based on the above features become unreliable when

$$std_{\mathrm{time}}[a_o(t)] \le threshold \tag{8}$$

or

$$std_{\mathrm{time}}[a_P(t)] \le threshold \tag{9}$$

where *threshold* = Y_THRESHOLD = 0.25 for the Ymean feature, *threshold* = TI_THRESHOLD = 0.15 for the TI2 and TI10 features, and std_{time} represents the standard deviation over time of the M samples in the feature stream.

When (8) or (9) are satisfied, the Ymean waveform has detected insufficient temporal changes in scene brightness to be useful for temporal registration (e.g., scene with a constant brightness level). Similarly when (8) or (9) are satisfied the TI2 and TI10 waveforms have detected

[4] 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.

insufficient temporal changes in the scene motion to be useful for temporal registration (e.g., still scene). Features that fall below the thresholds in (8) or (9) are considered "invalid" and no further calculations are performed using them. Furthermore, if all three features (TI2, TI10, and Ymean) fall below the thresholds in (8) or (9), then the video clip is considered "still" and no further temporal registration calculations are performed.

Feature sequence correlation is performed on the feature streams that pass the above test. The temporal registration uncertainty, U, will be specified in *fields* for interlaced systems and *frames* for progressive systems. U represents the maximum temporal shift (plus or minus) of the processed feature stream with respect to the original feature stream. The feature sequence correlation is performed as follows:

1. Given a sequence of M processed video features $\{a_p(t)\}_{t=1}^{M}$, we first discard the first and last U samples to form sequence $\{a_p(t)\}_{t=U}^{M-1-U}$. Normalize (divide) each element in the resulting sequence by the standard deviation of that sequence to form the normalized sequence $\{n_p(t)\}_{t=U}^{M-1-U}$.

2. For each alignment delay guess d (for all $-U \le d \le U$), we compute a corresponding original feature stream $\{a_o(t)\}_{t=U+d}^{M-1-U+d}$, and normalize (divide) each element in the sequence by the standard deviation of that sequence to form the normalized sequence $\{n_o(d,t)\}_{t=U}^{M-1-U}$. This original feature normalization is essentially the same as the processed feature normalization in step one, except computed for each delay, d.

3. Take the resulting normalized processed and original feature streams and compute the difference between those sequences:

$$\mathrm{Diff}(d,t) = \{n_o(d,t) - n_p(t)\}_{t=U}^{M-1-U} \tag{10}$$

4. Compute the sample standard deviation over time of the difference sequence for each delay offset d, namely

$$S(d) = std_{\mathrm{time}}(\mathrm{Diff}(d,t)) \tag{11}$$

5. The minimum $S(d)$ (denoted S_{min}) and its offset d_{min} is the best alignment indicated by this feature. Figure 2 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).

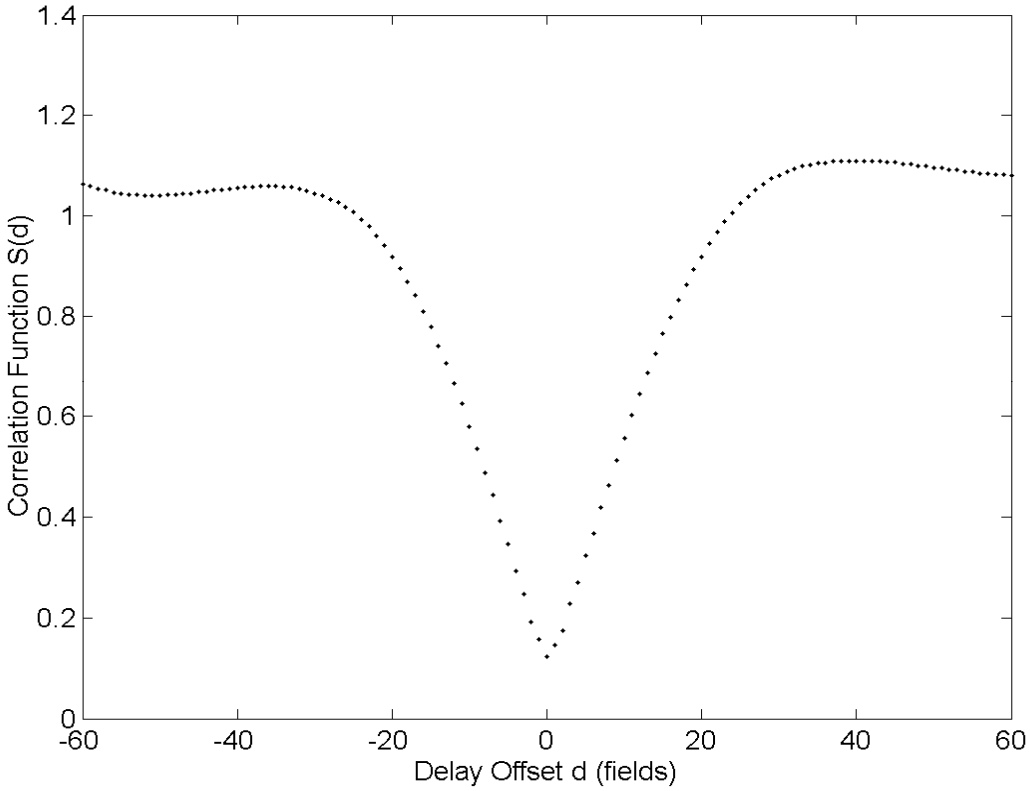

Figure 2. Example plot of correlation function S(*d*).

6. If the normalized original and processed feature streams were identical, then they would cancel at correct alignment (i.e., S_{min} would be 0.0). On the other hand, S_{min} can be at most 2.0, since the normalized original and processed waveforms each have a variance of 1.0. If the normalized original and processed waveforms are independent, their variances will add and S_{min} will be approximately equal to sqrt(2) ≈ 1.414. A value for S_{min} greater than sqrt(2) indicates that the two waveforms are negatively correlated. We have found that if $S_{min} \leq$ CORRELATION_VALID = 0.25, then the correlation between the processed feature stream and the original feature stream is probably reliable and these features are therefore considered valid. However, if $S_{min} \geq$ CORRELATION_INVALID = 1.40, then the correlation between the processed feature stream and the original feature stream is unreliable for the above reasons and these features are therefore considered invalid. For correlations in between (e.g., CORRELATION_VALID ≤ $S_{min} \leq$ CORRELATION_INVALID), S_{min} yields ambiguous results with respect to accuracy, so other criteria must be used (see step 7).

7. If (CORRELATION_VALID ≤ $S_{min} \leq$ CORRELATION_INVALID), find the earliest (minimum) delay, d_1, where $S(d_1) < S_{min}$ + DELTA_THRESHOLD, where DELTA_THRESHOLD = 0.04. Also find the latest delay, d_2, where $S(d_2) < S_{min}$ + DELTA_THRESHOLD. Compute the distance between those two delays, $w = d_2 - d_1 + 1$. Notice that no restrictions are placed on values of S for delays between d_1 and d_2. This width w

discriminates between correlations with a well-defined minimum, and correlations with multiple nearly-identical minimum values (e.g., a sharp correlation function as given in Figure 2 versus a broad correlation function). TI features with $w \leq$ TI_WIDTH are reliable, and Ymean features with $w \leq$ Y_WIDTH are reliable, where TI_WIDTH = 3 and Y_WIDTH = 4. Features that meet these criteria are considered valid whereas features that do not meet these criteria are considered invalid.

Table 2 provides a summary of the recommended threshold values that are used by the temporal registration algorithm. These recommended threshold values were obtained by minimizing alignment errors between the current algorithm and the frame based temporal alignment method in section 3.4.2 of [3].

Table 2. Recommended Values for Thresholds Used by Temporal Registration Algorithm

Threshold	Recommended Value
TI_THRESHOLD	0.15
Y_THRESHOLD	0.25
CORRELATION_VALID	0.25
CORRELATION_INVALID	1.40
DELTA_THRESHOLD	0.04
TI_WIDTH	3
Y_WIDTH	4

2.4 Estimation of Temporal Offset from Features

The following describes how to apply the three features of section 2.1 and the correlation algorithm of section 2.3 to achieve the final estimate of the temporal alignment offset.

1. Compute the original and processed TI2 feature streams, TI10 feature streams, and Ymean feature streams in section 2.1. When operating in-service, transmit the original features to the processed video location. At most, single precision will be required (i.e., 4-byte floating point), and further bandwidth savings can be obtained through quantization (e.g., 16 bits per value).

2. For each original and processed feature stream pair, compute the correlation function S(d) according to section 2.3 and record whether that feature is valid or invalid, and if invalid whether that feature is still (e.g., motionless).

3. If one or more features are valid, average those valid correlation functions together. For progressive video sequences, find the delay offset S_{min} that minimizes the averaged correlation function. For interlaced video sequences, restrict the search to field one delays or field two delays, and then find the delay offset S_{min} that minimizes the averaged correlation function.

Note: This algorithm cannot be used to determine with 100% reliability whether field one of the processed video sequence best aligns to field one or field two of the original sequence (i.e., indicative of interlaced reframing by the video system under test – see section 3.1.2 of [3]). We have found that such a determination is approximately 90% accurate. Thus, to detect reframing we recommend the use of some other, external algorithm such as the one that will be described in section 3.

4. If all of the features are invalid, a delay cannot be computed for this video clip. Furthermore, if all features have been marked still, then the clip contains still video and this extra information may be reported to the user. Temporal alignment or registration is not required to estimate quality for still scenes.

2.5 Observations and Conclusions

The in-service video delay measurement algorithm presented here uses a set of very low bandwidth features extracted from 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, 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, these original and processed 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 video scene delay as 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? Because the three features operate at different levels of temporal granularity, the video delays measured by these features may differ slightly for video systems that have field or frame repeats. By combining the results from all three features, a more robust delay estimate is obtained.

The length of the time window that is 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 (e.g., a four second head and shoulders scene wherein the only change in scene content is due to lip motion). Shorter time windows can be reliably 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.

This temporal registration algorithm is robust when other calibration errors are present. For instance, the temporal registration estimates are not significantly impacted by video system luminance gain and level offset errors. This was verified by inserting random luminance gains from 0.8 to 1.2 and random luminance level offsets from -20 to +20 into 2772 processed video sequences. The resulting temporal registration values were unchanged.

This temporal registration algorithm is only minimally impacted by spatial shifts and spatial scaling. This was tested by inserting random spatial scaling from 5% shrinkage to 5% expansion in both the horizontal and vertical directions, while simultaneously inserting random spatial shifts from -20 to +20 both horizontally and vertically. Figure 3 compares the temporal registration results from 2772 processed video sequences containing these spatial calibration errors with the temporal registration results from fully calibrated processed video. Notice that most of the video clip's alignments agree within plus or minus one frame, even in the presence of extreme calibration errors. Since many of the video clips contained frame repeats, some of the differences between running this temporal registration algorithm on calibrated and un-calibrated video clips may be indicative of truly ambiguous temporal registration. Spatial calibration errors increased the number of video clips with all features invalid from 159 to 165.

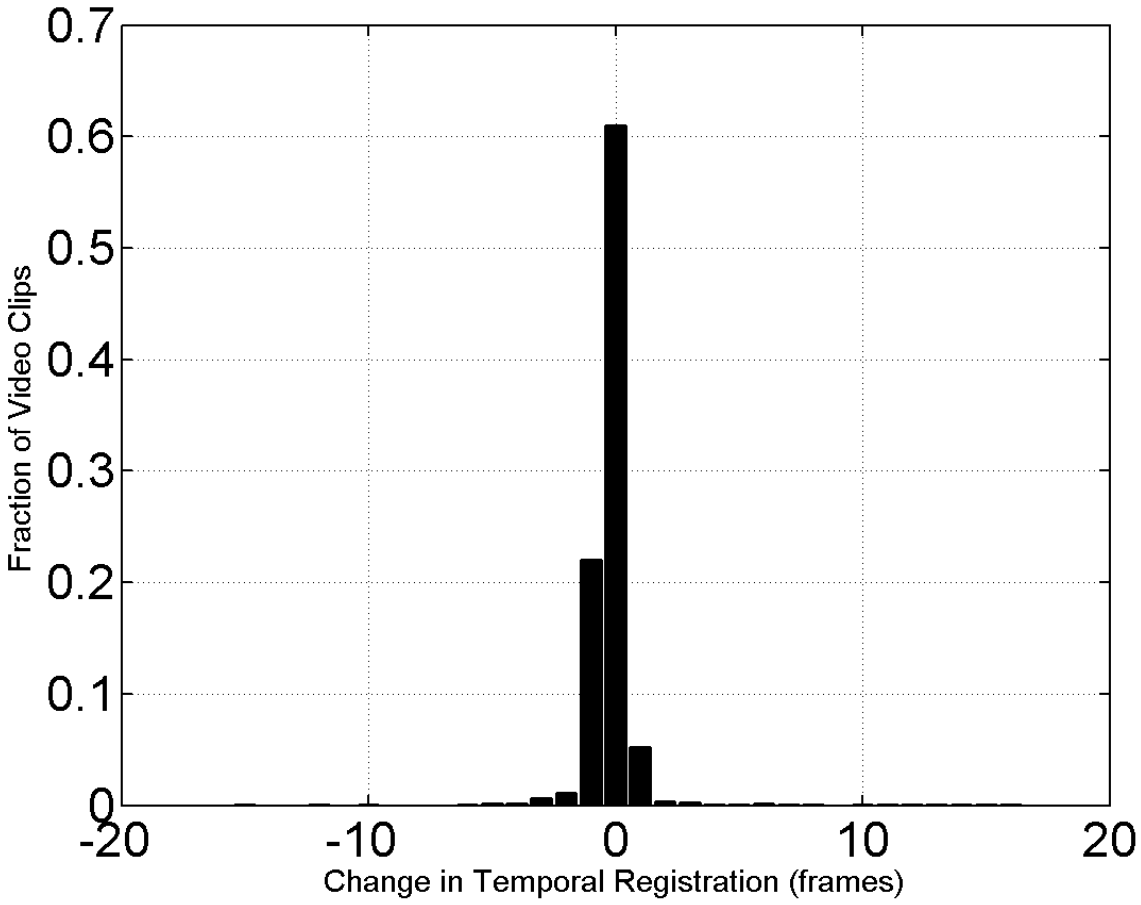

Figure 3. Impact of spatial calibration errors on RR temporal registration.

3 SPATIAL REGISTRATION ALGORITHM

The spatial registration algorithm requires as input the temporal offset between the original and processed video streams. This temporal offset is obtained from the temporal registration algorithm in section 2. The spatial registration algorithm simultaneously estimates horizontal and vertical spatial scaling and spatial shift. This searching utilizes RR features that are extracted from one frame every second. All used RR features of the video clip are considered simultaneously, to enhance the estimation accuracy. The spatial registration algorithm requires an RR data transmission bit-rate of 4 to 30 kb/s (depending upon image size and feature quantization accuracy).

The algorithm presented here is a downstream RR algorithm. Low bandwidth RR information from the original video sequence is transmitted to the processed video sequence location. More information is required from the processed video sequence than the original video sequence to perform the spatial searches. This algorithm can also be implemented as an upstream monitoring system with no loss in accuracy (i.e., the RR information is transmitted from the processed end to the original end, and extra information is used from the original video sequence rather than from the processed video sequence).

3.1 Preconditions

For the spatial registration algorithm to operate reliably, the original and processed video sequences must be temporally aligned. Since the temporal registration algorithm described in section 2 is robust against spatial shift and scaling errors that are present in the processed sequence, it may be used to temporally align the processed video before proceeding with spatial registration. The output of temporal registration is a single estimate of delay between the original and processed video sequences. For processed video sequences that exhibit constant or variable frame rate encoding, the temporal registration algorithm will produce an estimate of the average video delay for the clip.

The spatial registration algorithm does not perform *any* temporal search of its own. Temporal registration using the estimated average delay is presumed to be sufficiently accurate. This approach was found to yield accurate results for processed video sequences exhibiting constant or variable frame rate encoding. The training data used to develop the spatial registration algorithm included transmission errors, and these errors produced pausing with loss, pausing without loss, and corrupted images. The training data also included a wide range of frame rates, including variable frame rate encoding that occasionally was slower than 1 fps.

Because this algorithm does not do a temporal search, it is able to treat progressive and interlaced video sequences identically. Complications resulting from interlaced field reframing only arise when field streams are slid past each other. The elimination of temporal registration search from the spatial registration algorithm eliminates the need to treat interlaced video differently from progressive video.

As with temporal registration, a border of pixels and lines that do not contain valid picture should be eliminated (see section 2.1).

3.2 Algorithm Overview

A search over all of these available pixels in space and time would be prohibitively time consuming. Additionally, preliminary results indicate that these exhaustive searches can produce inaccurate results. Distortions present in the processed video sequence can result in erroneous spatial shift/scale values being selected. Instead, this spatial registration algorithm uses three types of RR information extracted from one frame every second: (1) a small number of pixels chosen at random, (2) the horizontal profile of pixels (average value of each column), and (3) the vertical profile of pixels (average value of each row). Due to averaging, the profiles are less susceptible to noise and thus provide a robust estimation of spatial shift and scaling, but cannot give a precise estimate. The randomly selected individual pixels yield more precise estimates, but are less robust. Taken together, the profiles and pixels yield robust and precise estimates of spatial shift and scaling.

Before continuing, we would like to explain the use of randomized pixel selection. Intuitively, it seems appropriate to use a heuristic to intelligently select pixels that are well suited to the task. However, it is quite difficult to develop a heuristic that provides an accurate response to the infinite variety of scene content and system impairments. During algorithm development, many heuristics were tried but none proved as robust as random choice. For example, pixels near strong edges are appropriate for some content, but misleading for other content. When selecting original pixels, it is difficult to guess the portions of the image that will be heavily impaired (and thus less useful) and the portions that will be accurately rendered – this depends too much upon the inner workings of the unspecified coding and transmission scheme. Selecting pixels randomly provides a Monte Carlo method for finding a near optimal solution.

3.3 Scaling and Shift Algorithm

The following constants are used in the spatial registration algorithm. HSHIFT is the maximum amount of horizontal shift to be searched, and VSHIFT is the maximum amount of vertical shift to be searched. These constants are specified in pixels for horizontal shift, and frame-lines for vertical shift. Both must be even numbers. HSCALE is the maximum horizontal scaling to be searched, and VSCALE is the maximum vertical scaling to be searched. These constants divided by 1000 yield the maximum fraction of image expansion or shrinkage. Thus, 60 indicates the expansion or shrinkage is limited to 6% (i.e., the scale factor is in the range 94% to 106%). Larger or smaller values can be used for both of these constants where appropriate. Table 3 gives reasonable default values of these constants for common image sizes.

Table 3. Default HSHIFT, VSHIFT, HSCALE, and VSCALE Values

Resolution	QCIF, QSIF	CIF, SIF	VGA, NTSC, PAL
HSHIFT	4	8	20
VSHIFT	4	8	20
HSCALE	60	60	100
VSCALE	60	60	100

The original RR video information that is transmitted must not come from pixels near the border of the valid region, as those pixels in the processed video are required for scaling and shifting within searches. To eliminate these pixels, an additional border is required of width given by:

$$\text{HPIXELS} = \textit{evenup}\ (\text{HSHIFT} + (\text{HSCALE}/1000) * \textit{cols}) \quad (12)$$

$$\text{VPIXELS} = \textit{evenup}\ (\text{VSHIFT} + (\text{VSCALE}/1000) * \textit{rows}) \quad (13)$$

where *rows* and *cols* are the number of rows and columns in the original image after eliminating the invalid border given in Table 1. Function *evenup* rounds a value up to the next even integer. For NTSC/525-line using the above suggested values, *rows* = 450, *cols* = 676, and thus HPIXELS = 88 and VPIXELS = 66 as shown in Figure 4.

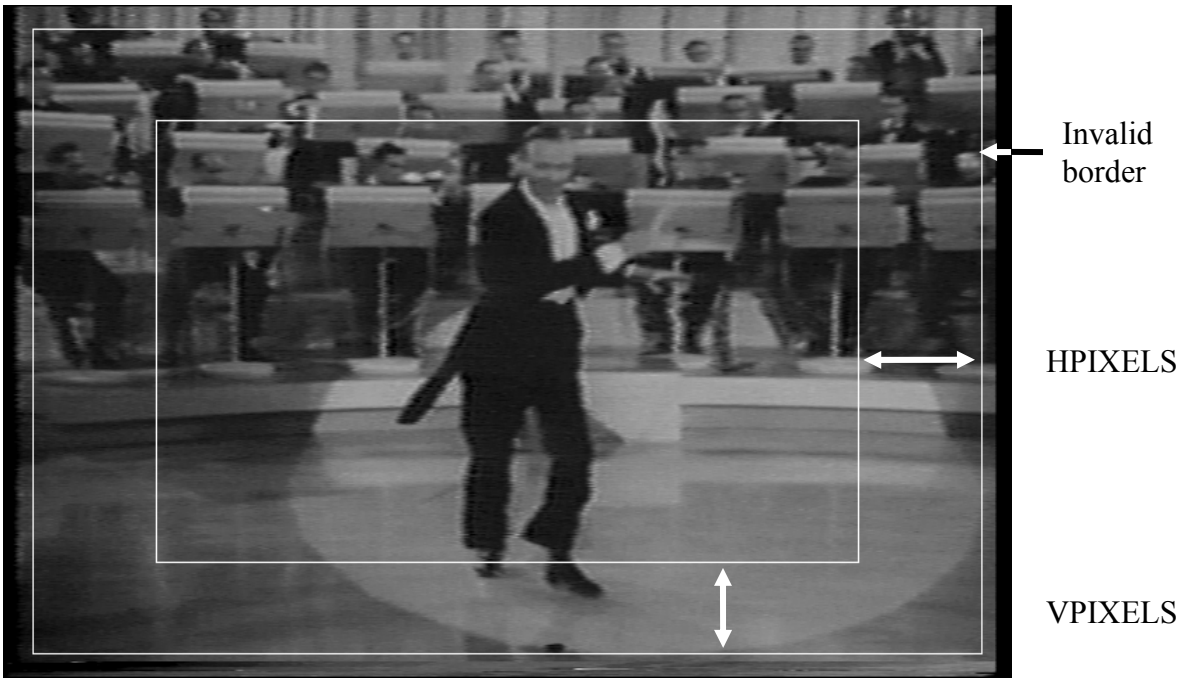

Figure 4. HPIXELS and VPIXELS visually demonstrated.

For the entire original video clip (e.g., 8 to 15 seconds long), the images to be used by the scaling and shift algorithms are first selected. It is recommended that the selected images be uniformly sampled in time and spaced one second apart. Thus, for an N second long sequence, N images are chosen. Let Y_n be the n^{th} luminance image in an original video clip that contains these N images (i.e., $n = 1, 2, \ldots, N$). Let $Y_n(i,j)$ be the coordinates of a pixel, where i is the vertical row index and j is the horizontal column index, and the upper-left coordinate of the image is $i = 1$, $j = 1$. Eliminate the invalid border shown in Table 1 and also a border of HPIXELS on left and right, and VPIXELS on top and bottom. Let O_n be the resulting image. For our NTSC/525-line example, O_n now contains 318 rows and 500 columns.

Compute the vertical profile of each original image (i.e., average each row) and join the profiles together into a single profile array, $VP_O(i,n)$, given by

$$VP_O(i,n) = \frac{1}{C_o}\sum_{j=1}^{C_o} O_n(i,j) \tag{14}$$

where C_o is the total number of columns in O_n.

Compute the horizontal profile of each original image (i.e., average each column) and join the profiles together into a single profile array, $HP_O(j,n)$, given by

$$HP_O(j,n) = \frac{1}{R_o}\sum_{i=1}^{R_o} O_n(i,j) \tag{15}$$

where R_o is the total number of rows in O_n.

A random number sequence will now be used to select random pixels from the N original video images O_n. Let S be an integer in the range [0, 255], chosen at random. S will be used as a seed for the pseudo-random number generator and can be transmitted in one byte. Since each computer uses the same seed S and the same pseudo-random number generator, each computer will produce the same random number sequence. As given by equation (16) below, we will use 80% as many random individual pixels as profile values. This 80% number is based on empirical evaluations of the algorithm performance and attaches slightly more weight to the horizontal and vertical profile samples versus the random pixel samples. Let M be the number of individual pixels to be selected from the video sequence.

$$M = round\ (0.8 * N * (R_o + C_o)) \tag{16}$$

$$orig\_i = round\ (rand(M) * R_o + 0.5) \tag{17}$$

$$orig\_j = round\ (rand(M) * C_o + 0.5) \tag{18}$$

$$orig\_n = round\ (rand(M) * N + 0.5) \tag{19}$$

where $rand(M)$ is a vector of M numbers from a random number generator with a uniform distribution over the range (0, 1) and *round* is a function that rounds to the nearest integer. Arrays *orig_i, orig_j, and orig_n* contain the row, column, and time coordinates for each of the M pixels, respectively. Each pixel location is chosen independently. Thus, more pixels may be selected from some images and less pixels from other images. For our NTSC/525-line example, *round* (0.8 * N * 818) pixels will be selected.

The next task is to find the best combination of horizontal scaling, vertical scaling, horizontal shift and vertical shift. An exhaustive search over these four dimensions would be prohibitively time consuming. Therefore, a randomized, iterative search strategy is used instead.

We begin by defining our initial guesses. Let *hm* be the current horizontal scaling factor (multiplicative), *ha* the current horizontal shift factor (additive), *vm* the current vertical scaling factor (multiplicative), and *va* the current vertical scaling factor (additive). At all times,

-HSHIFT < *ha* < HSHIFT (20)
-VSHIFT < *va* < VSHIFT
-HSCALE < *hm* < HSCALE
-VSCALE < *vm* < VSCALE

For each of the *N* luminance images in the processed video sequence, eliminate the invalid border shown in Table 1. Let P_n be the resulting processed image, R_p the number of rows in that image and C_p the number of columns. Notice that $R_p = R_o + 2 *$ VPIXELS; and $C_p = C_o + 2 *$ HPIXELS. Recall that P_n and O_n have already been temporally aligned by some external temporal registration algorithm such as that presented in section 2. For our NTSC/525-line example, P_n contains 450 rows (R_p) and 676 columns (C_p).

One could scale the processed video images by factors *hm* and *vm* before extracting the horizontal and vertical profiles. However, that approach is not very computationally efficient as it requires multiple two-dimensional resamplings of the processed images. Instead, the horizontal and vertical profiles are extracted from the processed video images and these extracted profiles are rescaled by factors *hm* and *vm*.

Compute the vertical profile of each processed image (i.e., average each row) and join the profiles together into a single profile array, $VP_p(i,n)$, given by

$$VP_p(i,n) = \frac{1}{C_p}\sum_{j=1}^{C_p} P_n(i,j) \tag{21}$$

Compute the horizontal profile of each processed image (i.e., average each column) and join the profiles together into a single profile array, $HP_p(j,n)$, given by

$$HP_p(j,n) = \frac{1}{R_p}\sum_{i=1}^{R_p} P_n(i,j) \tag{22}$$

To further speed computations, rescaling of the processed horizontal and vertical profiles is performed using nearest neighbor interpolation. We will first scale and then shift the original coordinates according to equations (23) and (24) below, and these scaled coordinates will then be used to index the processed profiles calculated by equations (21) and (22). The new vertical and horizontal profile indices (*prof_i* and *prof_j*, respectively) are given by

$$prof\_i = round\left(\frac{i}{vms+1} + va + \text{VPIXELS} + vms\,\frac{R_o}{2}\right),\ i = 1, 2, \ldots, R_o \tag{23}$$

$$prof\_j = round\left(\frac{j}{hms+1} + ha + \text{HPIXELS} + hms\,\frac{C_o}{2}\right),\ j = 1, 2, \ldots, C_o \tag{24}$$

where

$$vms = \frac{vm}{1000},\ hms = \frac{hm}{1000} \tag{25}$$

Similarly, the scaled and shifted image coordinates for the randomly sampled pixels are:

$$proc\_i = round\left(\frac{orig\_i}{vms + 1} + va + \text{VPIXELS} + vms\,\frac{R_o}{2} \right) \tag{26}$$

$$proc\_j = round\left(\frac{orig\_j}{hms + 1} + ha + \text{HPIXELS} + hms\,\frac{C_o}{2} \right) \tag{27}$$

$$proc\_n = orig\_n \tag{28}$$

Since the processed and original images are assumed to have been temporally aligned, the time index *orig_n* does not need to be scaled or shifted and thus remains unchanged.

The next task is to jointly consider the randomly chosen pixels, the vertical profiles, and the horizontal profiles. Let $\underline{O}$ be a column vector containing a concatenation of the original pixels from O_n at locations (*orig_i*, *orig_j*, *orig_n*), the original vertical profiles VP_o for $n = 1, 2, \ldots, N$, and the original horizontal profiles HP_o for $n = 1, 2, \ldots, N$. Similarly, let $\underline{P}$ be a column vector containing a concatenation of the processed pixels from P_n at locations (*proc_i*, *proc_j*, *proc_n*), the processed vertical profiles VP_p at locations *prof_i* for $n = 1, 2, \ldots, N$, and the processed horizontal profiles HP_p at locations *prof_j* for $n = 1, 2, \ldots, N$.

Let us now define our search criteria, V, which depends upon *hm*, *ha*, *vm* and *va*. V is the standard deviation of the difference between $\underline{O}$ and $\underline{P}$, namely

$$V = stdev(\underline{O} - \underline{P}) \tag{29}$$

Smaller values of V indicate a closer match between original and processed.

The search strategy contains two stages. The first stage searches randomly and with uniform density across the entire search space. The second stage refines the results of the first stage. It uses a 4-dimensional Gaussian distribution to focus the search in the vicinity of the current best point that minimizes V in equation (29). Each time a new best point is found, the second stage search is recentered about that point.

Let us define six variables: *W, min_W, min_hm, min_ha, min_vm,* and *min_va*. *W*(*hm, ha, vm, va*) will hold V for each horizontal scale *hm*, horizontal shift *ha*, vertical scale *vm*, and vertical shift *va*. *W*(*hm, ha, vm, va*) is initialized to NaN (Not-A-Number). *min_W* will hold the minimum V, whose value will be associated with horizontal scale *min_hm*, horizontal shift *min_ha*, vertical scale *min_hm*, and vertical shift *min_ha*. Initialize *min_W* to infinity. Note that *hm* will range from -HSCALE to HSCALE, while *ha* will range from -HSHIFT to HSHIFT. Likewise, *vm* will range from -VSCALE to VSCALE, while *va* will range from -VSHIFT to VSHIFT. Finally, let us choose TRIES, the number of evaluations to be performed before the

algorithm declares that a solution has been found. A value of TRIES = 15000 seems to work well and is the recommended default value.

For a number of iterations equal to TRIES / 10, choose values for *hm*, *ha*, *vm*, and *va* randomly over the range to be searched, using a uniform distribution of random values.

$$hm = round\ (\ \text{-HSCALE} + ((\text{HSCALE} * 2) * rand)), \tag{30}$$

$$ha = round\ (\ \text{-HSHIFT} + ((\text{HSHIFT} * 2) * rand)) \tag{31}$$

$$vm = round\ (\ \text{-VSCALE} + ((\text{VSCALE} * 2 * rand)) \tag{32}$$

$$va = round\ (\ \text{-VSHIFT} + ((\text{VSHIFT} * 2) * rand)) \tag{33}$$

where *rand* is a random number generator that yields numbers from the uniform distribution over the range (0, 1).

For each randomly chosen coordinate (*hm, ha, vm, va*), compute *V* as shown in equation (29) which will give the value for W(*hm, ha, vm, va*). Update the values of *W, min_W, min_hm, min_ha, min_vm, min_va* in equations (34) and (35).

$$W(hm, ha, vm, va) = V \tag{34}$$

If $V < min\_W$, then

$$min\_W = V,\ min\_hm = hm,\ min\_ha = ha,\ min\_vm = vm, \text{ and } min\_va = va \tag{35}$$

If a coordinate (*hm, ha, vm, va*) is chosen twice, then the calculation of *V* is skipped. Duplicate coordinates are detected by testing whether *W*(*hm, ha, vm, va*) contains NaN. Duplicate coordinates are counted in the number of evaluations to be tried.

After the TRIES / 10 = 1500 iterations, the coordinate (*min_hm, min_ha, min_vm, min_va*) will be a fairly close estimate of the actual coordinate of the minimum of *V*. Perform an additional TRIES * 9 / 10 iterations as shown above but with a modified distribution of random values. The new random distribution increases the likelihood of the chosen coordinate being closer to the current best point in the search space.

$$hm = min\_hm + round\ (2 * rand\_norm) \tag{36}$$

$$ha = min\_ha + round\ (2 * rand\_norm) \tag{37}$$

$$vm = min\_vm + round\ (2 * rand\_norm) \tag{38}$$

$$va = min\_va + round\ (2 * rand\_norm) \tag{39}$$

In equations (36)-(39), *rand_norm* is a random number generator that yields a normal distribution with zero mean and unity variance. If a random coordinate (*hm, ha, vm, va*) is outside the range to be searched, then another random coordinate is chosen instead. The long tails of the normal distribution help prevent the algorithm from locking in on a local minimum

rather than the global minimum. The quick handling of duplicate coordinates allows TRIES to be set to a large number without negatively impacting run speed. Note that equations (36)-(39) continually recenter the search about the current best point in the search space.

The final values for *min_hm, min_ha, min_vm,* and *min_va* indicate the horizontal and vertical scaling and shift factors. The *min_hm* and *min_vm* spatial scaling factors are used by a resampling routine to resample the processed video images.[5] After this spatial scaling, the resultant images are then shift corrected by *min_ha* and *min_va*. The user should verify that the processed video after calibration is properly registered to the original video.[6]

3.4 Median Filtering Values from Multiple Video Sequences

Spatial shift and scaling results for individual processed video clips contain occasional outliers. Still or nearly still scenes are particularly problematic, since estimates are essentially based upon one image rather than a sequence of images. If multiple processed video sequences that have passed through the same video system (i.e., all video sequences can be considered to have the same calibration numbers, except for temporal registration) are available, then spatial shift and spatial scaling results should be filtered across scenes to achieve increased accuracy. We have found that median filtering across scenes produces robust estimates for spatial scaling and spatial shift. If possible, we recommend median filtering across results from seven or more different video scenes.

Spatial shift was verified by comparing the results for 2565 processed video clips for this algorithm and the frame-based spatial shift algorithm in section 3.1.5 of [3]. The number of clips for the spatial registration validation (2565) was different than the number of clips for the temporal registration validation (2772) because spatial registration errors were introduced into some clips from tape editing, and these clips were discarded. Each video system represented by those 2565 video clips has at least two different original video clips, so that a limited amount of median filtering using different scenes could be performed for this study. Figure 5 shows the distance between the spatial shifts produced by this algorithm and those produced by the frame-based spatial registration algorithm in section 3.1.5 of [3]. The plot in Figure 5 is for individual clips and does not include any median filtering over scenes to increase accuracy.

Spatial scaling was verified by comparing the results for the above video clips to the spatial scaling algorithm in [4]. Figure 6 shows the distance between the spatial scaling of this algorithm and those produced by the algorithm in [4]. These spatial scaling results were visually verified when scaling was detected as being present. The plot in Figure 6 is for individual clips and does not include any median filtering over scenes to increase accuracy.

Figure 7 shows the increased agreement between the two sets of spatial shift results given in Figure 5 when median filtering over scenes is performed for both algorithms. Similarly, Figure 8

---

[5] This report makes no recommendations for how to perform high accuracy resampling of the processed video images using the final spatial scaling factors.

[6] A small shift (e.g., -0.4) may have to be added inside the *round* functions in equations (23), (24), (26), and (27) to properly compensate for shift differences between the nearest neighbor resampling algorithm presented here and more sophisticated resampling algorithms that use Finite Impulse Response (FIR) filters.

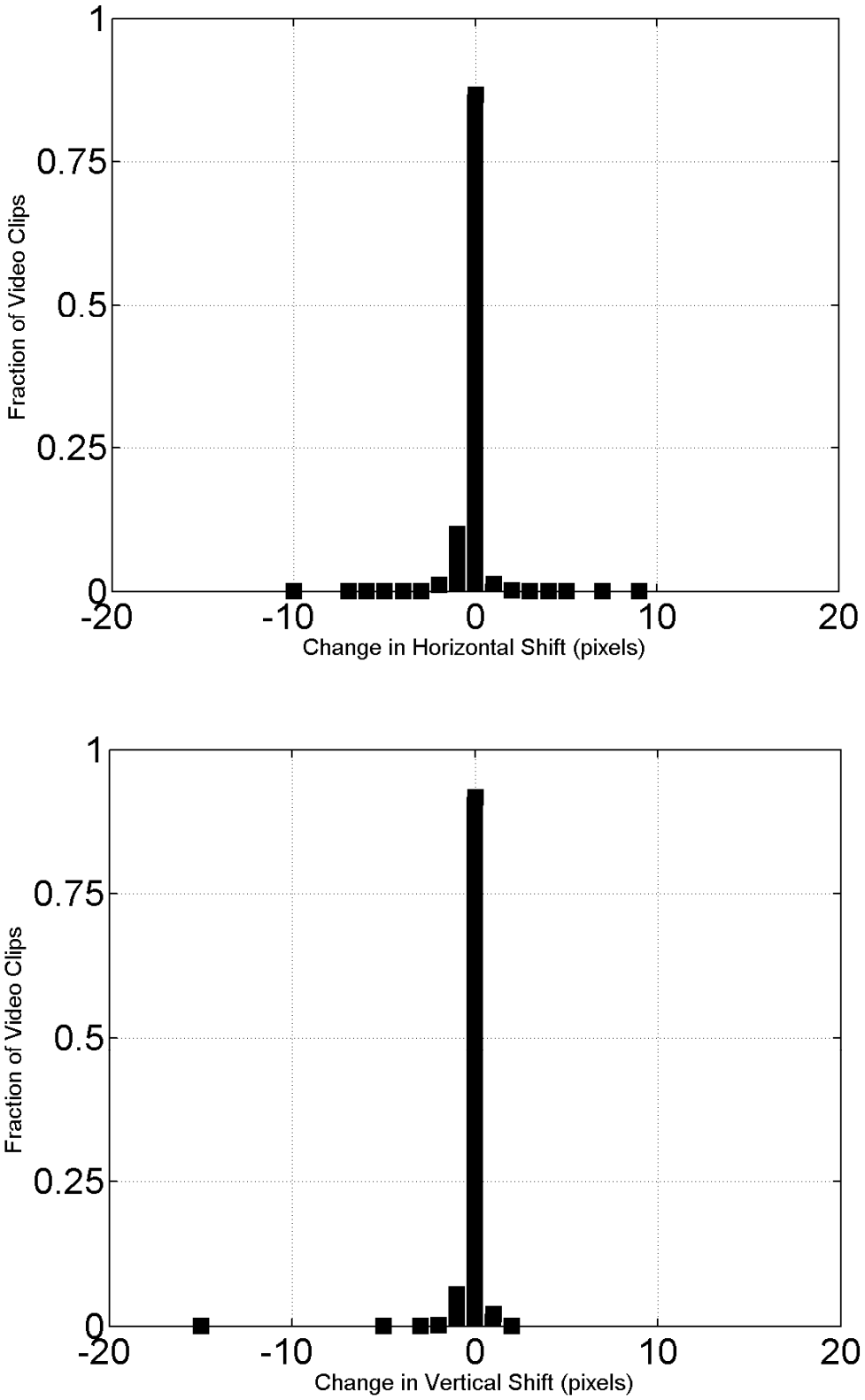

Figure 5. Difference between RR spatial shifts and the algorithm in section 3.1.5 of [3].

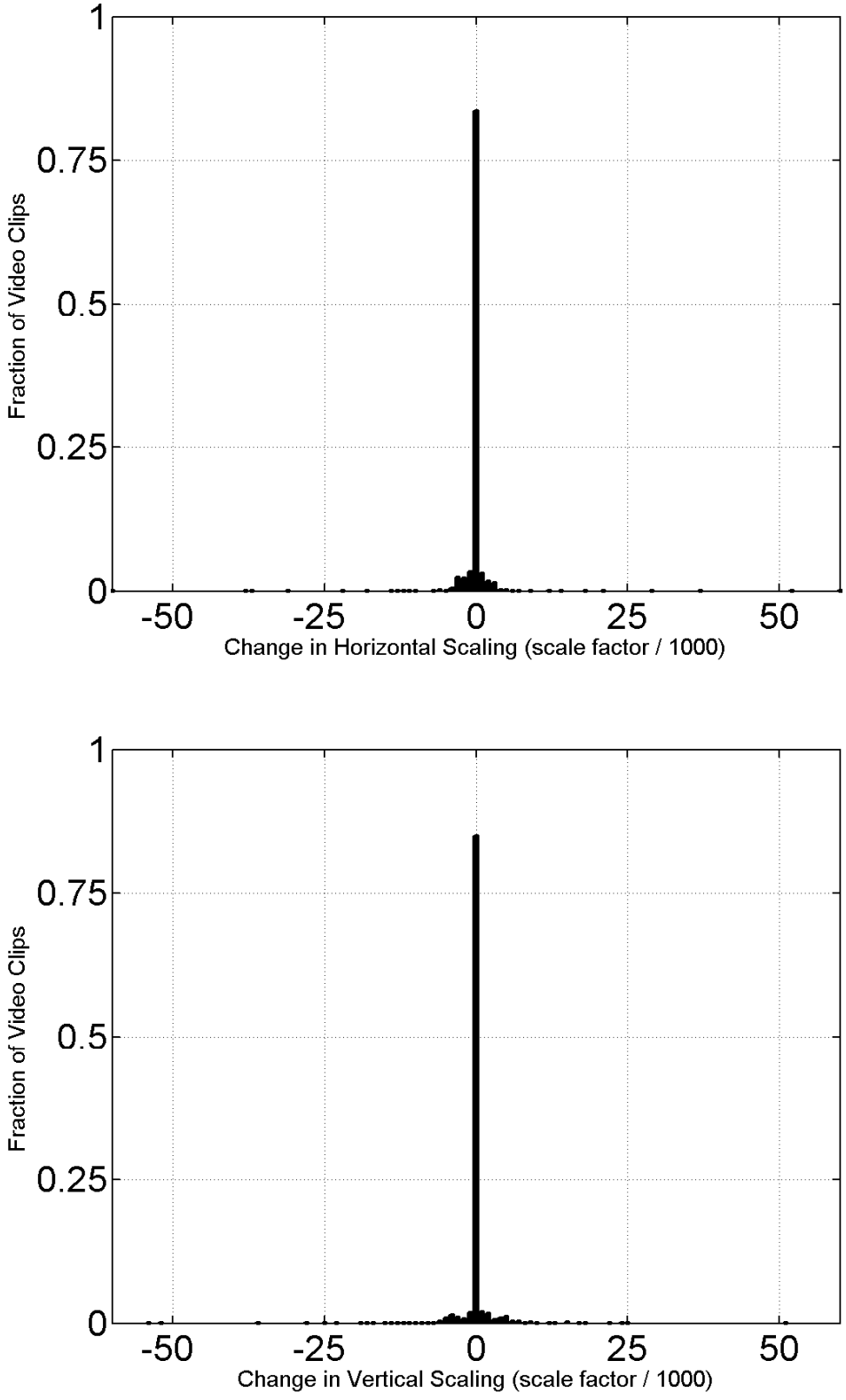

Figure 6. Difference between RR spatial scaling and the algorithm in [4].

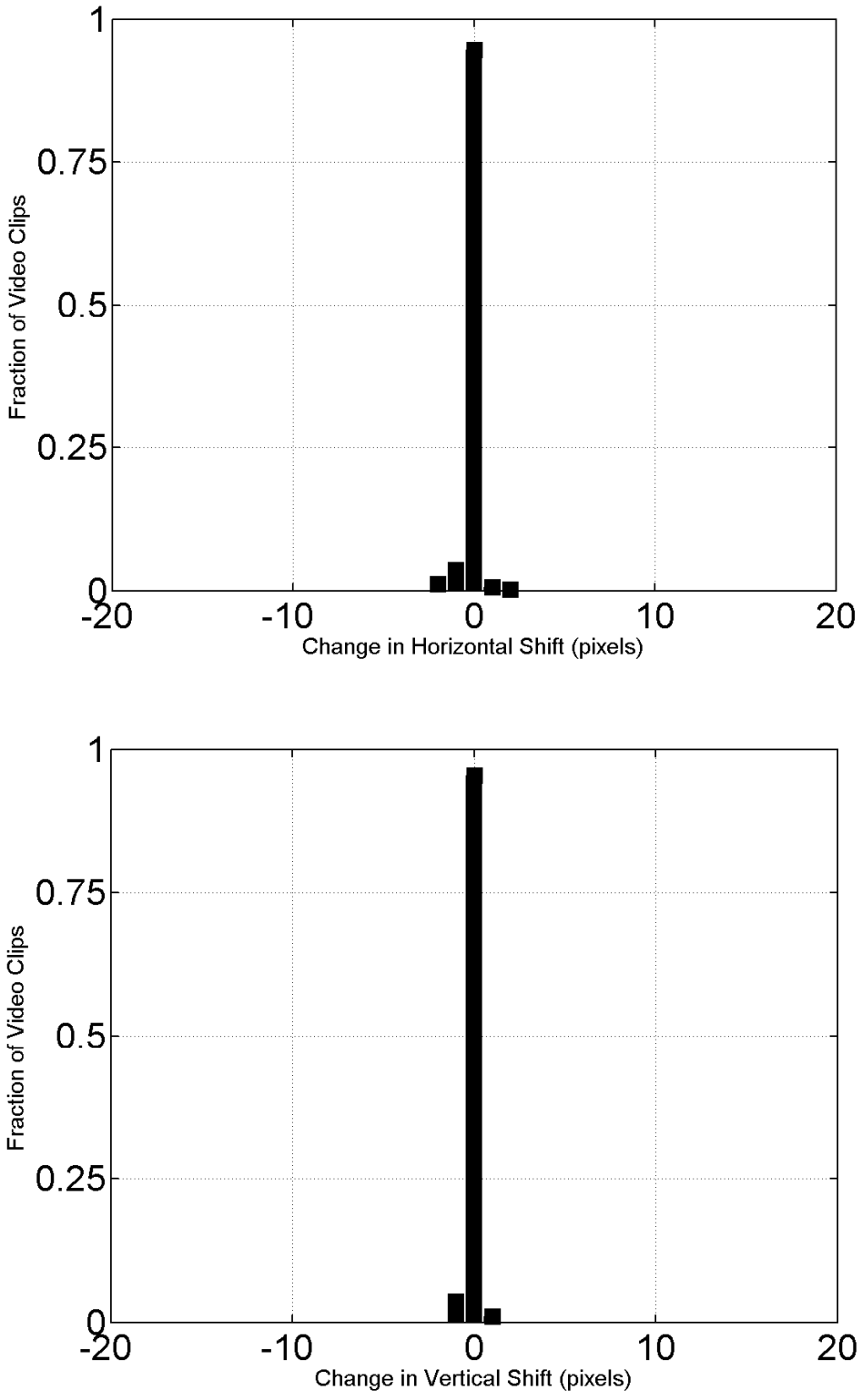

Figure 7. Difference between RR spatial shifts with median filtering (over scenes) and the algorithm in section 3.1.5 of [3].

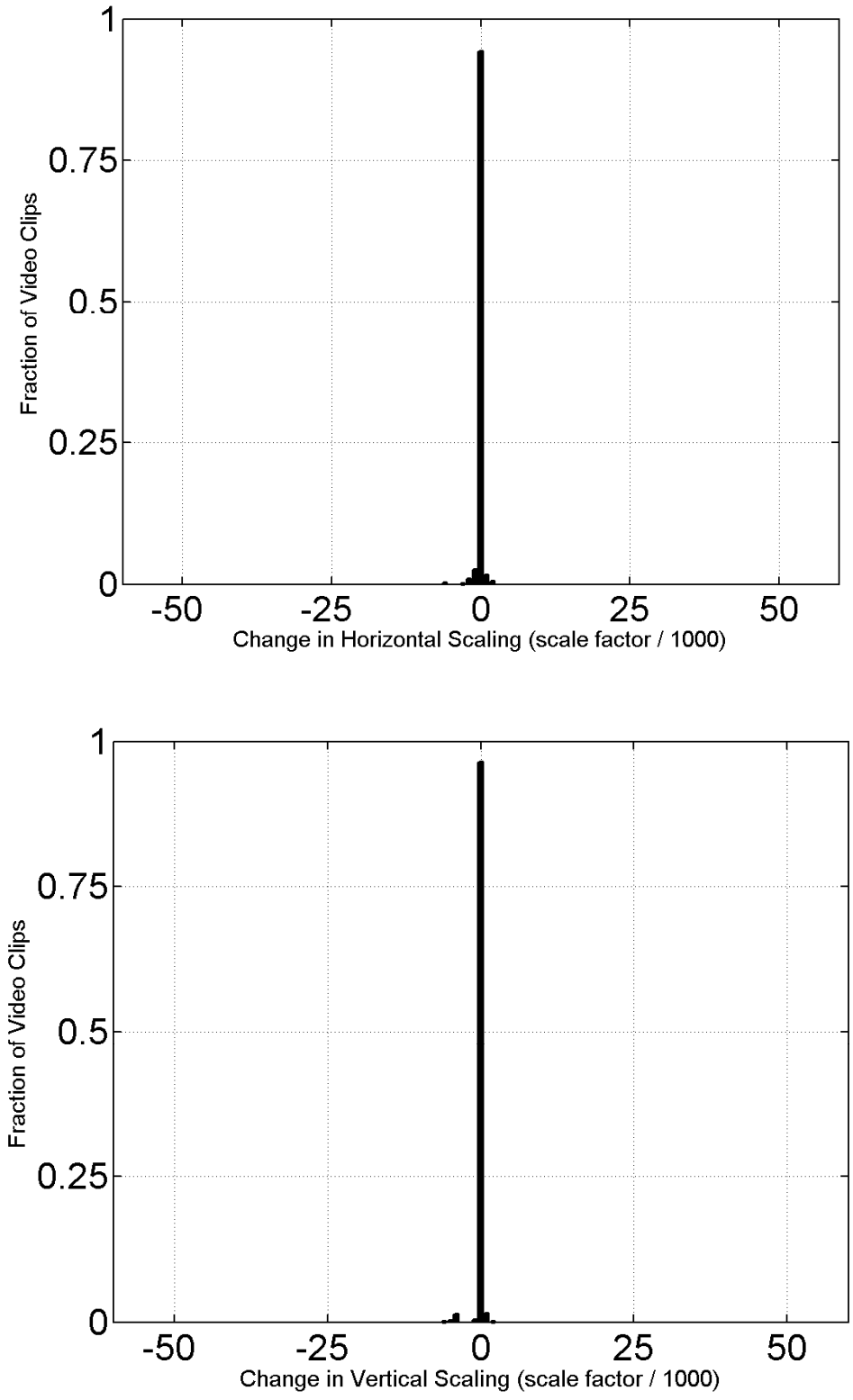

Figure 8. Difference between RR spatial scaling with median filtering (over scenes) and the algorithm in [4].

shows the increased agreement between the two sets of spatial scaling results given in Figure 6. For Figure 7 and Figure 8, there were a total of 247 different video systems. Notice that the median filtered results are tightly clustered around zero.

In general, the vast majority of the median filtered shifts agree within plus or minus one pixel; and almost all of the median filtered scaling factors agree within ±2 (i.e., ±0.2% scaling). Most of the non-zero differences between the algorithms come from video systems with four or fewer video scenes, or video systems that contained frequent digital transmission errors (e.g., significant transmission errors in every clip). Some of these differences may be indicative of truly ambiguous spatial registration. For example, if the video system blurs the video, performs field repetition (e.g., copies field one into field two), or utilizes lower resolution (e.g., 525-line video compressed to CIF and then up-converted and displayed at 525-line resolution), then spatial registration becomes ambiguous. Additionally, the reference algorithms used to define the "truth" data (i.e., the spatial registration algorithm in section 3.1.5 of [3], and the spatial scaling algorithm in [4]) may produce some errors.

It is reasonable to ignore median filtered horizontal and vertical scaling factors (*vm* and *hm*) that are within ±2 of 0 (i.e., ±0.2% rescaling). Since most video systems do not perform this kind of minor rescaling, scaling factors within ±2 of 0 are more likely to be indicative of ambiguous spatial registration than true rescaling. Thus, for these cases we recommend replacing the rescaling results with no rescaling.

3.5 Observations and Conclusions

The reduced reference spatial registration algorithm presented here uses a set of low bandwidth RR features extracted from the original video sequence. One byte each is required to transmit each randomly chosen original pixel, and at most single precision will be required (i.e., 4-byte floating point) to transmit the original horizontal and vertical profiles. With this level of quantization, the algorithm requires an RR data transmission bit-rate of approximately 27 kb/s for 525-line video, and 30 kb/s for 625-line video. Further bandwidth savings can be obtained through quantization (e.g., 16 bits per profile value).

This algorithm is suitable for measuring spatial shift and spatial scaling in a fully automated, in-service, real-time monitoring system or for an out-of-service environment, prior to performing video quality measurements. The video spatial scaling algorithm utilizes two types of features, pixels that track small local changes in scene content, and profiles that track overall trends. Accuracy is obtained by simultaneously estimating spatial shift and spatial scaling using the features extracted from multiple video frames. Results from multiple video sequences may be combined to further improve accuracy. This algorithm has been demonstrated to provide reliable estimates of spatial registration for a wide range of video scenes and systems.

4 LUMINANCE GAIN AND OFFSET ALGORITHM

The luminance gain and level offset estimation will use the column vectors $\underline{O}$ and $\underline{P}$ in equation (29) as calculated at the final values of *min_hm, min_ha, min_vm,* and *min_va*. This algorithm requires no extra bit-rate, as it utilizes data from section 3.

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

$$\underline{P} = g\underline{O} + l \tag{40}$$

There are only two unknowns (i.e., g and l) but Z equations, where

$$Z = M + N * (R_o + C_o) \tag{41}$$

Therefore, we must solve the over-determined system of linear equations given by:

$$\underline{P} \approx \underline{\hat{P}} = A \begin{bmatrix} l \\ g \end{bmatrix} \tag{42}$$

where A is a Z x 2 matrix given by

$$A_{Z\text{x}2} = \begin{bmatrix} \underline{1} & \underline{O} \end{bmatrix} \tag{43}$$

and $\underline{1}$ is a Z-element column vector of '1s' given by

$$\underline{1}_{Z\text{x}1} = \begin{bmatrix} 1_1 \\ . \\ . \\ . \\ 1_Z \end{bmatrix} \tag{44}$$

$\hat{P}$ 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 is given by

$$\begin{bmatrix} l \\ g \end{bmatrix} = \left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}} P \tag{45}$$

where the superscript "T" denotes matrix transpose and the superscript "-1" denotes matrix inverse. Problems with using equation (45) directly include using processed video samples that are highly distorted from the original video samples, and using processed video samples that are not correctly registered in space and time to the original video samples (e.g., the processed video may contain frame repeats and variable video delay). Thus, an iterative least squares solution with a cost function is used to help minimize the weight of these outliers in the fit.

The following iterative algorithm is applied to the original and processed samples:

1. Use the normal least squares solution, equation (45), to generate the initial estimate of the level offset and gain.

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:

$$\underline{E} = \left| \underline{P} - \hat{\underline{P}} \right| \tag{46}$$

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):

$$\underline{C} = \frac{1}{\underline{E} + \varepsilon} \tag{47}$$

   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:

$$\begin{bmatrix} l \\ g \end{bmatrix} = \left(A^{\mathrm{T}} C^2 A \right)^{-1} A^{\mathrm{T}} C^2 \underline{P} \tag{48}$$

8. Repeat steps 2 through 7 until the level offset and gain estimates converge to four decimal places.

To remove gain and level offset from the processed Y image plane, the following formula is applied to each processed pixel:

$$\text{New } Y(i,j,t) = [\ Y(i,j,t) - l\ ] / g \tag{49}$$

Similar to that found for spatial registration in section 3.4, we have found that median filtering across scenes produces more robust estimates for luminance gain and offset.

5 VALID REGION ALGORITHM

The valid region algorithm presented here uses a set of extremely low bandwidth RR features extracted from the original video sequence. Four integers are transmitted every time a valid region update is required (e.g., 16 b/s to update the valid region estimate every four seconds).

5.1 Valid Region when Over-Scan is Present

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. For instance, 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 receiving 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 may corrupt several lines at the bottom of the picture in the over-scan area). 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.

5.1.1 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 scaling and shift imposed by the video system must have been removed from the luminance image before applying the core algorithm (see section 3).

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.

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 is 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, the algorithm is the same but the directions are reversed (e.g., J is decremented instead of incremented). 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.

5.1.2 Applying the Core Valid Region Algorithm to a Video Sequence

Original Video

The core algorithm is first applied to the original sequence of images. For NTSC video sampled according to Rec. 601, 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 a uniform sub-sampling of images that follow. For example, the core algorithm could examine image numbers 0, 15, 30, 45, and so forth. When all selected 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 ensure that spatial-temporal sub-regions (from which video quality 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.

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 video scene. This maximum valid region is then reduced in size by any pixels and lines that are considered invalid due to spatial shift correction of the processed video frames. The core algorithm is then run on the first image in the processed video sequence and a uniform sub-sampling of the images that follow.

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, the bottom is decremented; likewise, if the region contains an odd number of pixels (e.g., horizontally), the right is decremented. The resulting valid region is returned as the processed valid region.

5.1.3 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 the valid region would quite likely result in 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 video 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 is not able 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.

5.2 Valid Region when Entire Picture is Displayed

CIF, SIF, QCIF, QSIF, and VGA are intended for computer display. These video sequences contain a valid picture all the way out to each edge. However, many of these sequences were originally shot using an NTSC, PAL or HDTV camera. As a result, these progressive video sequences occasionally have a border of pixels and lines that do not contain valid picture. This border is usually black, but a single line may sometimes contain values that are halfway between black and valid picture content (e.g., where the black border and valid picture values were averaged together). Often, this invalid border will occur on only one edge of the image. Digital video transmission systems usually do not further reduce the valid picture area, but may distort the sharp line from black to valid picture more than the rest of the picture. Viewers are used to ignoring black values around the edge of a picture, and will generally ignore this impairment. Thus, these border areas and related edge transitions should be excluded from video quality measurements. This case is further distinguished from the over-scan case because the number of pixels and lines can be quite small. Thus, extra care must be taken to preserve the maximum valid region possible. This section presents a modified version of the valid region algorithm in section 5.1, where the modifications seek to preserve as much of the valid region as possible.

5.2.1 Modified Core Valid Region Algorithm

This section describes the modified 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 luminance image of a single video frame. When measuring the valid region of a *processed* video sequence, any spatial scaling and shift imposed by the video system must have been removed from the luminance image before applying the core algorithm (see section 3).

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.

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 contain 92% of the image horizontally and 92% of the image vertically. A border of 4% of the available pixels and lines is placed outside

the current valid region at each image edge. Thus, the final valid region estimate will lie between this 4% point and the entire image.

The modified core algorithm examines the area of video between the edge of the picture 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" and the mean will be represented by "M_J".

2. Take the mean of the next column of pixels, "M_{J+1}".

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 steep ramp[7] up from black border to valid picture ($M_J + 20 < M_{J+1}$). If either of these conditions is 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, the algorithm is the same but the directions are reversed (e.g., J is decremented instead of incremented). 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.

5.2.2 Applying the Modified Core Valid Region Algorithm to a Video Sequence

Original Video

The modified core algorithm is first applied to the original sequence of images. The core algorithm is run on the first image in the video sequence and a uniform sub-sampling of the images that follow. For example, the core algorithm could examine image numbers 0, 15, 30, 45, and so forth. When all selected 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 edge of the image are

[7] The difference in the slope of the ramp between section 5.1and 5.2 is intentional. Design differences between these types of video codecs influence this constant.

considered to contain either black or a transitional ramp up from black. The resulting valid region is returned as the original valid region.

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 spatial scaling and shift correction of the processed video frames. The modified core algorithm is then run on the first image in the processed video sequence, and a uniform sub-sampling of the images that follow. The resulting valid region is returned as the processed valid region.

6 COMBINING ALGORITHMS AND APPLYING CORRECTIONS

To fully calibrate a processed video sequence, we suggest performing the calibration in the following order. The calibration information from earlier steps is used in subsequent steps. First, estimate the temporal registration of the uncalibrated processed video sequence using the algorithm in section 2 and apply the appropriate temporal correction. Second, estimate the spatial scaling and shift using the algorithm in section 3. Third, estimate the luminance gain and level offset using the algorithm in section 4. Fourth, estimate the valid region using the algorithms in section 5. Fifth, if the spatial registration values are non-zero, optionally compensate for the spatial registration and reapply the temporal registration algorithm in section 2 using the calibrated processed video clip to obtain a slightly improved temporal registration estimate. This fifth step is more important for interlaced video because the new temporal registration will be field-accurate.

If spatial scaling, spatial shift, luminance gain, and level offset estimates are available for other processed video sequences that have passed through the same video system (i.e., all video sequences can be considered to have the same calibration numbers, except for temporal registration), then calibration results can be filtered across scenes to achieve increased accuracy. We have found that median filtering across scenes produces robust estimates for spatial scaling, spatial shift, luminance gain, and level offset. In this case, the fourth and fifth steps given above can result in improved accuracy by median filtering the values for spatial scaling, spatial shift, luminance gain, and level offset.

Taken together, these four algorithms accurately estimate video calibration values in an RR measurement environment. The temporal registration algorithm requires a data transmission bit-rate of 1 to 6 kb/s (depending upon frame rate and feature quantization accuracy). The spatial registration and luminance gain/offset algorithms jointly require a data transmission bit-rate of 4 to 30 kb/s (depending upon image size and feature quantization accuracy). The valid region algorithm requires an insignificant transition bit-rate (i.e., four image coordinates).

7 REFERENCES

[1] ITU-T Recommendation J.143, "User requirements for objective perceptual video quality measurements in digital cable television," Recommendations of the ITU, Telecommunication Standardization Sector.

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

[3] S. Wolf and M. Pinson, "Video quality measurement techniques," NTIA Report 02-392, Jun. 2002. Available at <www.its.bldrdoc.gov/n3/video/documents.htm>.

[4] M. Pinson and S. Wolf, "Video scaling estimation technique," NTIA Technical Memorandum TM-05-417, Jan. 2005. Available at <www.its.bldrdoc.gov/n3/video/documents.htm>.

FORM **NTIA-29** U.S. DEPARTMENT OF COMMERCE
(4-80) NAT'L. TELECOMMUNICATIONS AND INFORMATION ADMINISTRATION

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION NO. TR-06-433	2. Government Accession No.	3. Recipient's Accession No.
4. TITLE AND SUBTITLE Reduced Reference Video Calibration Algorithms		5. Publication Date October 2005
		6. Performing Organization Code
7. AUTHOR(S) Margaret H. Pinson and Stephen Wolf		9. Project/Task/Work Unit No. 3141000-300
8. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Telecommunication Sciences National Telecommunications & Information Administration U.S. Department of Commerce 325 Broadway Boulder, CO 80305		10. Contract/Grant No.
11. Sponsoring Organization Name and Address National Telecommunications & Information Administration Herbert C. Hoover Building 14th & Constitution Ave., NW Washington, DC 20230		12. Type of Report and Period Covered
14. SUPPLEMENTARY NOTES		

15. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)

This document describes four Reduced Reference (RR) video calibration algorithms of low computational complexity. RR methods are useful for performing end-to-end in-service video quality measurements since these methods utilize a low bandwidth network connection between the original (source) and processed (destination) ends. The first RR video calibration algorithm computes temporal registration of the processed video stream with respect to the original video stream (i.e., video delay estimation). The second algorithm jointly calculates spatial scaling and spatial shift. The third algorithm calculates luminance gain level offset of the processed video stream with respect to the original video stream. The fourth algorithm estimates the valid video region of the original or processed video stream (i.e., the portion of the video image that contains actual picture content). All the algorithms utilize the luminance (Y) image plane of the video signal.

16. Key Words (Alphabetical order, separated by semicolons)

calibration; delay; gain; offset; spatial scaling; spatial shift; temporal shift; video

17. AVAILABILITY STATEMENT	18. Security Class. (This report)	20. Number of pages
☐ UNLIMITED.	Unclassified	34
	19. Security Class. (This page) Unclassified	21. Price:

NTIA FORMAL PUBLICATION SERIES

NTIA MONOGRAPH (MG)

A scholarly, professionally oriented publication dealing with state-of-the-art research or an authoritative treatment of a broad area. Expected to have long-lasting value.

NTIA SPECIAL PUBLICATION (SP)

Conference proceedings, bibliographies, selected speeches, course and instructional materials, directories, and major studies mandated by Congress.

NTIA REPORT (TR)

Important contributions to existing knowledge of less breadth than a monograph, such as results of completed projects and major activities. Subsets of this series include:

NTIA RESTRICTED REPORT (RR)

Contributions that are limited in distribution because of national security classification or Departmental constraints.

NTIA CONTRACTOR REPORT (CR)

Information generated under an NTIA contract or grant, written by the contractor, and considered an important contribution to existing knowledge.

JOINT NTIA/OTHER-AGENCY REPORT (JR)

This report receives both local NTIA and other agency review. Both agencies' logos and report series numbering appear on the cover.

NTIA SOFTWARE & DATA PRODUCTS (SD)

Software such as programs, test data, and sound/video files. This series can be used to transfer technology to U.S. industry.

NTIA HANDBOOK (HB)

Information pertaining to technical procedures, reference and data guides, and formal user's manuals that are expected to be pertinent for a long time.

NTIA TECHNICAL MEMORANDUM (TM)

Technical information typically of less breadth than an NTIA Report. The series includes data, preliminary project results, and information for a specific, limited audience.

For information about NTIA publications, contact the NTIA/ITS Technical Publications Office at 325 Broadway, Boulder, CO, 80305 Tel. (303) 497-3572 or e-mail info@its.bldrdoc.gov.

This report is for sale by the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161,Tel. (800) 553-6847.