NTIA Handbook 02-01

Video Quality Measurement User's Manual

Margaret Pinson
Stephen Wolf

handbook series

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

NTIA Handbook 02-01

Video Quality Measurement User's Manual

Margaret Pinson
Stephen Wolf

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

Nancy J. Victory, Assistant Secretary
for Communications and Information

February 2002

DISCLAIMER

TABLE OF CONTENTS

VIDEO QUALITY MEASUREMENT USER'S MANUAL

Margaret Pinson and Stephen Wolf [1]

The purpose of this handbook is to provide a user's manual for the video quality metric (VQM) tool. The VQM software tool performs automated batch processing of video files. Program VQM runs under the UNIX operating system and uses a control file to specify the exact video quality measurement procedures that are to be performed. All results are emailed to the user.

Program VQM compares the video sequence that has been processed by the video system under test to the original video sequence through two main steps. First, program VQM calibrates the processed video sequence to remove systematic differences between the original and processed, such as spatial and temporal shifts. Second, program VQM estimates and reports the perceived quality of the processed video using one of five video quality models. Quality estimates are reported on a scale of zero to one, where zero means that no impairment is visible and one means that the video clip has reached the maximum impairment level.

1. INTRODUCTION AND OVERVIEW

The Institute for Telecommunication Sciences (ITS) has developed an automated video quality metric (VQM) software tool that performs automated batch processing of video files that have been sampled in accordance with ITU-R Recommendation BT.601 [1], henceforth abbreviated as Rec. 601. The video quality measurement algorithms implemented by the VQM software tool are described in detail in [2]. These algorithms include calibration of the sampled video streams (e.g., gain and level offset, spatial registration, and temporal registration), as well as the calculation of video quality parameters and models of overall quality perception. The purpose of this document is to provide a user's manual for the VQM tool.

Program VQM runs under the UNIX operating system and requires one command line argument, which contains the name of a control file (e.g., file.cntl) that specifies in detail the exact video quality measurement procedures that are to be performed on the original (i.e., unimpaired reference) and processed (i.e., impaired) video files. The format for the control file is given in section 3.

To run the VQM program, type

```
vqm file.cntl
```

at the UNIX prompt.

<u>Important Note:</u> VQM creates temporary files in the current directory and the tmp directory. Insufficient disk space will prevent VQM from running successfully. When VQM finishes, it performs a cleanup by deleting all temporary files.

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

VQM will email results to the user's email address specified by the first line of the control file. When VQM completes execution and does not encounter anything abnormal, the emailed result files include one log file and one parameter time history file for each processed video file and its associated original video file. The log file provides a summary of the calibration and video quality measurement results. The time history file provides a detailed time history for each video quality parameter that was measured. When calibration results stray outside of the expected range (see section 10 of [2]), a calibration root cause analysis (RCA) file is also produced and emailed separately. Other files may also be emailed (see section 5).

1.1 Hardware and Operating System Requirements

The VQM program runs under the UNIX operating system. Binary executable programs are currently available for the following computer architectures:

1. SGI[2] – MIPS R12000 processor running SGI IRIX 6.5 or later. Binary contains optimized, 64-bit code.
2. SGI – MIPS R8000 or later processor running SGI IRIX 6.5 or later. Binary contains 32-bit code.

To determine if the SGI machine has an R8000 or R12000 MIPS processor, run the hardware inventory (hinv) command at the UNIX prompt.

3. SUN[3] – ULTRA 10 or later processor running OS 5.8 or later, with gnu GCC version 2.95.3 installed (environmental variable LD_LIBRARY_PATH must contain /usr/local/lib).
4. HP[4] – PA-RISC 1.1 processor or later running HPUX 11.0 or later, with gnu GCC version 2.95.3 installed
5. Red Hat Linux[5] – Pentium[6] III Class CPU or greater running Red Hat Linux Version 8.0 with Sendmail installed and configured to send mail.

1.2 Terms And Definitions

4:2:2 – A Y, Cb, Cr image sampling format where chrominance planes (Cb and Cr) are sampled horizontally at half of the luminance (Y) sampling rate. See Rec. 601 [1].

Big YUV - The binary file format used for storing clips that have been sampled according to Rec. 601. In the Big YUV format, all the video frames for a scene are stored in one large binary file, where each individual frame conforms to Rec. 601 sampling.

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

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

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

[2] SGI, MIPS, and IRIX are registered trademarks of Silicon Graphics, Inc.

[3] SUN and ULTRA are registered trademarks of Sun Microsystems, Inc.

[4] HP, PA-RISC, and HPUX are registered trademarks of Hewlett-Packard, Inc.

[5] Red Hat and Linux are registered trademarks of Red Hat, Inc.

[6] Registered trademark of Intel, Inc.

Frame – One complete television picture.

Gain – A multiplicative scaling factor applied by the hypothetical reference circuit (HRC) to all pixels of an individual image plane (e.g., Luminance or Chrominance). Gain is commonly known as contrast.

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

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

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

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

Offset – An additive factor applied by the system under test to all pixels of an individual image plane (e..g, Luminance or Chrominance). Offset is commonly known as brightness.

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

Output Video – Video after being processed or distorted by an HRC. Output video may also be referred to as processed video.

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

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

Rectangle Coordinates – Used to describe a rectangular shaped image sub-region that is specified by four coordinates (top, left), (bottom, right). Numbering starts from zero so that the (top, left) corner of the sampled image is (0, 0).

Reframing – The process of reordering two consecutively sampled interlaced fields of processed video into a frame of video. Reframing is necessary when the system under test does not preserve standard interlace field types (e.g., an NTSC field type one is output as an NTSC field type two and vice versa).

Root Cause Analysis (RCA) – Objective or subjective analyses used to determine the presence or absence of specific video artifacts (e.g., blurring, tiling, or dropped frames) in the processed video. Root Cause Analysis provides the user with detailed information on the likely cause of quality degradations measured by VQM. Root Cause Analysis lists a percentage for several possible impairments (e.g., jerky motion, blurring, error blocks), where 100% indicates all viewers perceive the impairment as a primary artifact, 50% indicates viewers perceive the impairment as a secondary artifact, and 0% indicates the artifact would not be perceived. Root Cause Analysis gives a more detailed view of the degradation than the single number quality estimate.

Spatial Registration – The spatial shifts of the processed video sequence with respect to the original video sequence.

Temporal Registration – The temporal shift (i.e., video delay) of the processed video sequence with respect to the original video sequence.

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

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

VQM Model - A particular algorithm that is used by the VQM software that has been specifically optimized to achieve maximum objective to subjective correlation based upon certain optimization criteria, including the range of quality over which the model applies and the speed of computation.

2. VIDEO QUALITY MEASUREMENT STEPS

Video quality measurements encompass three distinct steps. First, the video must be converted from its original format into an electronic file format recognized by the VQM program. Second, the calibration quantities for the video are determined (i.e., spatial shifts, invalid picture area, gain and level offset, and temporal shifts). Third, the processed video is calibrated and the quality of the video sequence is estimated.

2.1 Video File Format

Currently, the VQM program supports the "Big YUV" file format, which is a convenient binary file format for storing Rec. 601 sampled video. In the Big YUV format all the frames are stored sequentially in one big binary file. The sampling is 4:2:2 and image pixels are stored sequentially by video scan line as bytes in the following order: Cb_1 Y_1 Cr_1 Y_2 Cb_3 Y_3 Cr_3 Y_4…, where Y is the luminance component, Cb is the blue chrominance component, Cr is the red chrominance component, and the subscript is the pixel number. The Y signal is quantized into 220 levels where black = 16 and white = 235, while the Cb and Cr signals are quantized into 225 levels with zero signal corresponding to 128. Occasional excursions beyond these levels may occur. The two chrominance components are sub-sampled by two horizontally. For more information, see Rec. 601 [1]. The appendix contains MATLAB[7] programs for reading and displaying 525-line Big YUV files. These programs may be used to verify that the Big YUV files are in the correct file format before executing the VQM program.

Converting video from other formats into the Big YUV format usable by VQM is outside the scope of this document. Video conversion issues can be quite complex. Care should be taken to preserve field ordering (i.e., field one remains in field one, field two remains in field two). The image's spatial scaling must be the same for the original and processed video files, because the VQM program does not support image scaling (e.g., scaling may be present if the processed video is zoomed in or out). If video is acquired via frame grabbing, make sure there are no missing or dropped video frames, as this will affect the video quality measurement.

2.2 Calibration

VQM includes an extensive calibration process that is performed before the quality metrics are calculated. Calibration involves four steps: spatial registration, valid region detection, gain/level offset calculation, and temporal registration. These steps may be calculated automatically for each original/processed video

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

clip pair using the ITS calibration algorithms.[8] Alternatively, the user may choose to enter manual values. However, if values are manually entered, the user must be very careful, because **improper calibration values can influence the video quality prediction, thereby invalidating results!** Improper calibration values usually cause the VQM program to predict a worse quality than the processed video clip actually warrants.

Many video systems spatially shift the image. For example, the image may be shifted ten pixels to the left and four lines up. Viewers normally do not care about these spatial shifts unless they are excessive. The spatial registration algorithm detects and removes spatial shifts that were introduced into the processed video by the video system under test. Each processed video clip is assumed to have one spatial shift. Spatial registration errors can adversely affect the video quality prediction.

Most video images include a black border around the edges. This area is usually part of the over scan, and thus will never be seen by the viewer. Many digital video systems increase the width of this black border (i.e., transmit fewer picture elements) to achieve greater video compression. The valid video region detection algorithm detects the valid portion of the processed video (i.e., that portion of the processed video that contains valid picture content) and discards pixels outside of that area from further consideration. Each video scene can have a different valid region. A valid video region that is too large can adversely effect the video quality prediction.

Some video systems impose a gain and level offset to the video's luminance level. The gain change is like adjusting a television's contrast knob, and the level offset change is like adjusting a television's brightness knob. Errors in specifying gain & level offset can adversely affect the video quality prediction. An automated algorithm can be used to estimate the gain and level offset of a single video clip. If the video system's gain and level offset is known, this information may be manually entered in the control file.

Most video systems have a time delay (e.g., video delay). Temporal registration is the process of time aligning a segment of processed video with the corresponding segment of original video. Exact temporal registration may be ambiguous due to certain types of impairments caused by the video system under test. Examples of these impairments include frame repetition and/or variable video delay (e.g., the delay through the system can change depending upon the scene content or network conditions). Temporal registration errors can adversely affect the video quality prediction. Two automated temporal registration algorithms can be used: sequence-based and frame-based. The reader is referred to [2] for a technical description of these algorithms and reasons why one algorithm might be preferred over the other for certain types of video systems. Temporal registration can also be manually entered.

The automated calibration routines should produce the correct calibration values a majority of the time. These automated calibration routines are recommended as defaults for that reason. However, the wise course would be to check the emailed results file for any errors or problems encountered by the calibration process, and to check the calibration values for reasonableness. For example, the spatial registration should be constant for any particular video system. Therefore, all clips run through that system should produce the same spatial registration result. When abnormal calibration values are detected, a warning report is produced by the VQM program. When calibration errors are detected, manual specification of calibration values can be used to fix problems that were encountered, and the video quality measurements can then be recomputed.

[8] In [2], robust calibration algorithms are described that utilize the calibration information from a set of video clips (rather than just one), all of which came from the same video system. Improved estimates of gain, level offset, and spatial shift can be obtained by filtering across scenes, since these calibration quantities are normally fixed for a given video system. However, the VQM program does not support these system-based calibration algorithms.

2.3 Quality Estimation

Once calibration has been completed, the VQM program estimates the quality of the video segment. Each video quality model produces one number, the estimated quality of the video sequence, as an end result. This number is on a scale of zero to nominally one, where zero means that no impairment is visible and one is the maximum impairment level observed for clips in the ITS subjective data base (see sections 8 and 9 of [2]).[9]

In general, the ITS video quality models are linear combinations of four or more parameters that measure different aspects of video quality. Each of these parameters is reported in two ways. First, the parameter has one overall value for the entire video sequence. That value is reported in the log file adjacent to the overall video quality. Second, each parameter has a time sequence of values. These time sequences of parameters are reported in a separate time history file.

The VQM program implements five models of perceived video quality, which are described in detail in [2]. The **general_model** is suitable for systems spanning a wide range of quality levels. The **developer_model** has less accuracy (i.e., poorer correlation to subjective score) than the general model but runs about five times faster. The **tv_model** is optimized for broadcast and DVD applications (e.g. MPEG-2). The **vconf_model** is optimized for video conferencing applications (e.g., H.261, H.263, MPEG-4). The **psnr_model** implements the peak signal to noise ratio measurement.

2.3.1 Considerations for Long Video Sequences

The quality models are based on subjective testing procedures that utilized scenes from 8 to 10 seconds in length. If the big YUV file contains a scene that is longer than 10 seconds, it may be parsed (i.e., broken into smaller segments) using the VQM program parsing functions. These smaller scene segments may be overlapping or not. The control file parameters that specify parsing behavior are described in list element (12) of section 3.

3. CONTROL FILE OPTIONS

The control file must conform to a strict text file format. Each line of the control file starts with two words followed by a colon, where the two words identify a unique VQM program control variable. White space is inserted after the colon and the variable value is listed after this white space. This section identifies and defines all the text lines in the control file. Section 4 provides a few example control files.

Each item listed here identifies a unique line of text in the control file. These lines must appear in the order given here. The reader may want to examine the example control files given in section 4 when reading this description. White space between words may be one or more spaces or tabs. Tabs are used in the examples to enhance readability.

(1) Email Address:

All VQM results will be emailed to this address.

(2) Original File:

Name of file containing the original video.

[9] Scores greater than one (i.e., clips more impaired than those found in the ITS subjective data base) are compressed using a crushing function to limit excursions beyond one (see section 7 of [2]).

(3) Original Bytes:

Length of the original video file, in bytes.

(4) Number Processed:

The number of processed video files listed in this control file. Items **Processed File** and **Processed Bytes** are repeated in the control file for each processed file. Thus, one control file can be used to process an original file and many corresponding processed files.

(5) Processed File:

Name of file containing the processed video.

(6) Processed Bytes:

Length of the processed video file identified on the previous line, in bytes.

(7) Is Compressed:

Whether the video files are compressed (**True**) or not (**False**).

- **True** – Video files are compressed (reserved for future use).
- **False** – Video files are in the Big YUV format.

(8) Image Rows:

The number of rows (or lines) in one image frame (e.g., 486).

(9) Image Cols:

The number of columns (or pixels) in one image frame (e.g., 720).

(10) Timing Format:

The video timing of the video sequences stored in the original and processed files. Legal options are "NTSC", "PAL", "HALF_NTSC" and "THIRD_NTSC".

- **NTSC** – Video files contain 29.97 or 30 frames per second.
- **PAL** – Video files contain 25 frames per second.
- **HALF_NTSC** – Video files contain 15 frames per second.
- **THIRD_NTSC** – Video files contain 10 frames per second.

(11) Scanning Pattern:

The video scanning pattern of the video sequences stored in the original and processed files.

- **Interlace** – Each frame is split into two interlaced fields, one containing the odd-numbered lines, the other containing the even-numbered lines. Field ordering depends upon the standard indicated by the **Timing Format**: NTSC or PAL.
- **Progressive** – Each frame is a progressively-scanned video frame.

(12) Parsing Type:

Whether video files should be parsed into shorter time segments and quality measurements made over these parsed segments.

- **NONE** – Video files will not be parsed. This option is only available when **Original Bytes** and all **Processed Bytes** are identical. This option is recommended for video files approximately 5 to 15 seconds in length.

- **SPLIT** – Static splitting of video files. Control file must contain **Parsing Length** and **Parsing Frequency**, immediately following this control line. The original and processed video files will be split using the same algorithm, into segments containing **Parsing Length** frames. Segment #1 will start with the first sample in the video file. Segment #N will start **Parsing Frequency** frames after the start of segment #(N-1).
- **RUNNING** – Dynamic splitting of video files. Control file must contain **Parsing Length** and **Parsing Frequency**. Processed video files will be split according to the **SPLIT** option, described above. However, the original video file will be split into segments containing **Parsing Length** frames as follows. Segment #1 of the original video file will start with the first sample in the original video file. Next, segment #1 of the processed video file will be calibrated and the quality measured using segment #1 of the original video file. Segment #2 of the original video file will start **Parsing Frequency** frames after the start of original segment #1, adjusted by the temporal registration of processed segment #1. Segment #2 of the processed video file will then be calibrated and the quality measured using segment #2 of the original. In general, segment #N of the original video file will start **Parsing Frequency** frames after the start of original segment #(N-1), adjusted by the temporal registration of processed segment #(N-1). Segment #N of the processed video file will be calibrated and the quality measured before segment #(N+1) of the original video file is parsed. This option is most useful if the processed file is losing or gaining frames when compared to the original file. In this case, for **Temporal Calibration** choose **AUTOMATIC**, and for **Temporal Algorithm** choose **Frame**.

(12.1) Parsing Length:

When parsing, the parsing length specifies the length (in frames) of each parsed segment.

(12.2) Parsing Frequency:

When parsing, the parsing frequency specifies the number of frames between the beginning of segment N and the beginning of segment N+1. This can be used to produce overlapping or sub-sampled segments (e.g., when **Parsing Frequency** is less than **Parsing Length**). When **Parsing Frequency** is equal to **Parsing Length**, the parsed segments will abut. When **Parsing Frequency** is greater than **Parsing Length**, the parsed segments will be separated by a number of frames.

(13) Spatial Calibration:

The method used to spatially register the processed video sequence.

- **AUTOMATIC** – Spatial registration is computed automatically, using scene content (see section 4.1.5 of [2]).
- **VALUES** – Spatial registration values are specified manually. The control file must contain **Spatial F1Vertical** and **Spatial F2Vertical** (or **Spatial Vertical**), and **Spatial Horizontal** control lines immediately following this control line.

(13.1) Spatial F1Vertical:

Only used when **Spatial Calibration** is **VALUES** and **Scanning Pattern** is **Interlace**. Used to manually specify the vertical shift of field one in field lines. Value must be an integer. A positive vertical shift is associated with a processed field that has been moved down by that number of field lines. A negative vertical shift is associated with a processed field that has been moved up by that number of field lines.

In most cases, **Spatial F1Vertical** should be equal to **Spatial F2Vertical**. If the video system under test reframes video (e.g., moves NTSC field one into NTSC field two, and NTSC field two into the next NTSC field one), then **Spatial F1Vertical** should be equal to **Spatial F2Vertical** minus one**.**

(13.2) Spatial F2Vertical:

Only used when **Spatial Calibration** is **VALUES** and **Scanning Pattern** is **Interlace**. Used to manually specify the vertical shift of field two in field lines. Value must be an integer. A positive vertical shift is associated with a processed field that has been moved down by that number of field lines.

In most cases, **Spatial F2Vertical** should be equal to **Spatial F1Vertical**. If the video system under test reframes video (e.g., moves NTSC field one into NTSC field two, and NTSC field two into the next NTSC field one), then **Spatial F2Vertical** should be equal to **Spatial F1Vertical** plus one**.**

(13.3) Spatial Vertical:

Only used when **Spatial Calibration** is **VALUES** and **Scanning Pattern** is **Progressive**. Used to manually specify the vertical shift in frame lines. Value must be an integer. A positive vertical shift is associated with a processed frame that has been moved down by that number of frame lines.

(13.4) Spatial Horizontal:

Only used when **Spatial Calibration** is **VALUES**. Used to manually specify the horizontal shift of the fields (**Interlace**) or frames (**Progressive**) in pixels. Value must be an integer. A positive horizontal shift is associated with a processed field or frame that has been moved to the right by that number of pixels.

(14) Valid Calibration:

The method used to determine the valid region of the processed video sequence. The valid region step of calibration locates the rectangular area of the processed video sequence that contains valid picture content. The border of black pixels around the edge of the image is marked "invalid".

- **AUTOMATIC** – Valid region is computed automatically, using scene content (see section 4.2.2 of [2]).
- **VALUES** – Valid region is specified manually. The control file must contain **Valid Top, Valid Left, Valid Bottom,** and **Valid Right** control lines immediately following this control line.

(14.1) Valid Top:

Only used when **Valid Calibration** is **VALUES**. Specify the first or top-most line of the image frame that contains valid video. This must be an even number, where 0 is the top-most line of the image.

(14.2) Valid Left:

Only used when **Valid Calibration** is **VALUES**. Specify the first or left-most pixel of the image frame that contains valid video. This must be an even number, where 0 is the left-most pixel of the image.

(14.3) Valid Bottom:

Only used when **Valid Calibration** is **VALUES**. Specify the line past the last line of the image frame that contains valid video. This must be an even number, where 0 is the top line of the image.

(14.4) Valid Right:

Only used when **Valid Calibration** is **VALUES**. Specify the pixel past the last or right-most pixel of the image frame that contains valid video. This must be an even number, where 0 is the left-most pixel of the image.

(15) Gain Calibration:

The method used for computing the gain and level offset of the processed video. Gain and level offset are computed separately for the luminance and chrominance image planes (i.e., Y, Cb, and Cr). For the Y component, the gain and level offset calculations assume that Y_processed = Y_gain*Y_original + Y_offset. Similar equations apply for the Cb and Cr components.

- **AUTOMATIC** – Gain and level offset are computed automatically, using scene content (see section 4.3.3 of [2]). Gain and level offset estimates are reported for all three video components (Y, Cb, Cr) but only applied to the Y video component.
- **VALUES** – Luminance gain and level offset values are specified manually. The control file must contain **Luminance Gain** and **Luminance Offset** control lines immediately following this control line. When **Gain Calibration** is **VALUES**, gain and level offset for the chrominance image planes are not specified.

(15.1) Luminance Gain:

Only used when **Gain Calibration** is **VALUES**. Specifies the gain that has been applied to the Y component of the processed video. This is a real number, which must be between 0.5 and 2.0. Large system gains may produce processed video that has amplitude clipping.

(15.2) Luminance Offset:

Only used when **Offset Calibration** is **VALUES**. Specifies the level offset (in Rec. 601 quantization levels, where an offset of 1 is one quantization level) that has been applied to the Y component of the processed video. This is a real number that can be either positive or negative. Large system offsets may produce processed video that has amplitude clipping.

(16) Temporal Calibration:

The method used to determine the temporal registration of the processed video sequence.

- **AUTOMATIC** – Temporal registration is computed automatically using the **Temporal Algorithm** specified in item (16.2). The control file must also contain (16.2) **Temporal Algorithm** and (16.3) **Temporal InvalidUncertainty** control lines. When parsing with the **RUNNING** option, **AUTOMATIC** must be chosen for **Temporal Calibration**.
- **VALUES** – Temporal registration is specified manually. The control file must contain the **Temporal Shift** control line immediately following this control line.

(16.1) Temporal Shift:

Only used when **Temporal Calibration** is **VALUES**. When **Scanning Pattern** is **Progressive,** the temporal shift is specified in frames. When **Scanning Pattern** is **Interlace,** the temporal shift is specified in fields. For interlaced video, an even number indicates a frame shift and an odd number indicates that the video system under test has reframed the video (e.g., moved NTSC field one into NTSC field two, and NTSC field two into the next NTSC field one). Temporal shift should only be odd when the spatial registration values also indicate reframing. See comments under **Spatial F1Vertical** and **Spatial F2Vertical**. If reframing is required, the processed video sequence is reframed, never the original video sequence.

Independent of the **Scanning Pattern**, positive temporal shifts indicate that images must be removed from the beginning of the processed video segment. Negative temporal shifts indicate that images must be removed from the beginning of the original video sequence.

(16.2) Temporal Algorithm:

Only used when **Temporal Calibration** is **AUTOMATIC.** Specifies whether to use the sequence-based or frame-based temporal registration algorithm (sections 4.4.1 and 4.4.2, respectively, of [2]).

- **Sequence** – Automated temporal registration algorithm compares a sequence of values computed for the original video clip, to a sequence of values computed from the processed video clip. This algorithm is well suited for television video systems (e.g., systems with little or no dynamic time warping & dropped frames).
- **Frame** – Automated temporal registration algorithm attempts to align each processed video image (e.g., frames for progressive video, fields for interlace video) with a matching original video image. The most commonly observed video delay is selected for the delay of the segment. This algorithm is well suited for videoconferencing systems or when the range of video quality being measured is unknown.

(16.3) Temporal InvalidUncertainty:

Only used when **Temporal Calibration** is **AUTOMATIC.** Specifies whether the VQM program retains the maximum possible number of frames after determining temporal registration, or discards a number of frames equal to the **Alignment Uncertainty**, at the beginning and end of the processed video segment.

- **False** – Discard only those frames that must be discarded due to the temporal alignment (i.e., those original frames for which there are no processed frames, and vice versa).
- **True** – Discard a number of frames equal to the **Alignment Uncertainty**, at the beginning and end of the processed video segment. This option is useful when used in combination with parsing, because the user knows, *a priori*, the portion of each segment for which the video quality metric will be computed.

(17) Alignment Uncertainty:

Specifies the alignment uncertainty (in frames) to be used by all calibration steps that require an alignment uncertainty. The alignment uncertainty indicates the uncertainty of the alignment of the individual segments after parsing. The processed segment must be temporally registered within plus or minus the alignment uncertainty with respect to the original segment in order for the temporal registration algorithm to work properly. For a **RUNNING** parse, the alignment uncertainty is the greatest number of frames by which the temporal registration of segment #N may be greater than or less than the temporal registration of segment #(N-1).

(18) Calibration Frequency:

Specifies the frequency (in frames) at which images in the video streams are examined during calibration, by all steps that look at every **Calibration Frequency**-th image. For instance, if **Calibration Frequency** is 15 frames, the gain and level offset will be estimated using processed frame 0, frame 15, frame 30, frame 45, and so forth (i.e., every 15th frame).

(19) Video Model:

The video quality model (section 7 of [2]) used to predict the overall perceptual impression of video quality. Valid options are:

- **general_model** A general model suitable for systems spanning a wide range of quality levels.
- **developer_model** Less accurate than the general model but runs about 5 times faster. This model is suitable for systems spanning a wide range of quality levels.

- **tv_model** A model optimized for TV broadcast and DVD applications, such as MPEG-2 video.
- **vconf_model** A model optimized for video conferencing applications, such as H.261, H.263, and MPEG4.
- **psnr_model** A model based upon peak signal to noise ratio, mapped to a 0 to 1 scale, where 0 indicates imperceptible impairment, and 1 indicates maximum impairment.

4. CONTROL FILE EXAMPLES

This section gives several examples of control files for typical user applications of the VQM tool.

4.1 Entire File Unparsed, Automatic Calibration

This control file takes as input a pair of Big YUV formatted files. Each file contains NTSC (525-line) video frames, sampled at the standard Rec. 601 rate of 486 rows by 720 columns. Each video file contains 301 frames (10 seconds plus one frame).

The entire processed video file and original video file will be compared for processing purposes (i.e., each video file contains one video sequence). For the unparsed option, the original and processed video files must contain exactly the same number of images. Calibration will be done automatically using the standard ITS calibration algorithms. These calibration algorithms presume that the proper time alignment of the pair of video sequences is within plus or minus 30 frames (i.e., one second) of the initial time alignment, which presumes that the first frame in each file aligns. That is, the N^{th} frame of the processed video sequence aligns somewhere between original frame (N-30) and (N+30).

After the original and processed video sequences have been calibrated, video quality measurements will be computed on the largest common sequence of time-aligned video (i.e., those processed frames for which there are aligned original frames). Since the time alignment uncertainty is 30 frames, the final time aligned processed segment will contain somewhere between 271 and 301 frames. Because the video clip is assumed to contain one constant and unchanging delay, the sequence-based temporal registration is chosen. After calibration, video quality will be measured using the general model. Results will be emailed to jsmith@its.bldrdoc.gov.

```
Email Address:          jsmith@its.bldrdoc.gov
Original File:          orig.yuv
Original Bytes          210651840
Number Processed:       1
Processed File:         proc.yuv
Processed Bytes:        210651840
Is Compressed:          False
Image Rows:             486
Image Cols:             720
Timing Format:          NTSC
Scanning Pattern:       Interlace
Parsing Type:           NONE
Spatial Calibration:    AUTOMATIC
Valid Calibration:      AUTOMATIC
Gain Calibration:       AUTOMATIC
```

Temporal Calibration:	AUTOMATIC
Temporal Algorithm:	Sequence
Temporal InvalidUncertainty:	False
Alignment Uncertainty:	30
Calibration Frequency:	15
Video Model:	general_model

4.2 Running Parsing, Abutted Segments, Automatic Calibration

This control file takes a pair of five-minute, Big YUV video files. Since the video quality models were not intended to operate on such long video sequences, these video files will be parsed into 30 second segments (900 frames), and then each segment will be processed separately. The files will be parsed with a running parse and temporally registered with the frame-based algorithm, because the delay is expected to change over the course of the file. From any one segment to another, the temporal alignment is expected to change by no more than 10 seconds, and so an alignment uncertainty of 300 frames (10 seconds) is used. This is quite a large uncertainty and would normally be much larger than is required.

The operator wants to be able to concatenate parameter time histories from each time segment to form a long time history for the entire processed video file. So, the user has requested the **Temporal InvalidUncertainty** option, which means that the first 10 seconds and last 10 seconds of processed video segment will be considered "invalid" after the temporal registration process and hence no quality parameters will be extracted from these frames. Quality parameters will be extracted from the middle 10 seconds of the processed video segment. To be able to abut the parameter time histories from successive processed time segments, the following rule *must* be followed: the parsing frequency must be equal to the parsing length minus twice the alignment uncertainty.

The parsing length was chosen to be 30 seconds (900 frames). Subtracting twice the alignment uncertainty from that number means a parsing frequency of 10 seconds (300 frames) must be used. The processed file will be parsed into 30-second segments, each overlapping by 20 seconds. Video quality measurements will use the middle 10 seconds of each processed segment, and the 10 second segment out of the 30-second original segment that provides the best alignment. Therefore, when considered as a whole, the quality predictions will cover the entire file, continuously, with two exceptions: (1) the first 10 seconds of the processed file (i.e., the alignment uncertainty) will never be processed, and (2) some video at the end of the processed file, something greater than or equal to the alignment uncertainty (10 seconds), will also be left unprocessed.

All calibration will be done automatically. After calibration, video quality will be measured using the general model. Results will be emailed to jsmith@its.bldrdoc.gov.

Email Address:	jsmith@its.bldrdoc.gov
Original File:	original.yuv
Original Bytes	6298560000
Number Processed:	1
Processed File:	processed.yuv
Processed Bytes:	6298560000
Is Compressed:	False
Image Rows:	486
Image Cols:	720
Timing Format:	NTSC
Scanning Pattern:	Interlace
Parsing Type:	RUNNING

Parsing Length:	900
Parsing Frequency:	300
Spatial Calibration:	AUTOMATIC
Valid Calibration:	AUTOMATIC
Gain Calibration:	AUTOMATIC
Temporal Calibration:	AUTOMATIC
Temporal Algorithm:	Frame
Temporal InvalidUncertainty:	True
Alignment Uncertainty:	300
Calibration Frequency:	15
Video Model:	general_model

4.3 SIF, Split Parsed, Abutted Segments, Automatic Calibration

This control file lists one original video sequence and three processed video sequences. All files are in the Big YUV format and contain 900 frames. Because the SIF sampling structure was used, the Big YUV files contain progressive images that are sampled at 352 pixels by 240 lines. These are SIF images, displayed at 30 frames per second, and so the "NTSC" timing option is selected. The files will be parsed with a static parse, because the delay is not expected to increase or decrease steadily over the course of the file. From any one segment to any other, the temporal alignment is expected to change by no more than plus or minus 5 seconds, and so an alignment uncertainty of 180 frames (6 seconds) is used. The extra second was added for safety.

The operator also wants to be able to concatenate parameter time histories for each segment to form a time history for the entire processed video file. So, the user has requested the **Temporal InvalidUncertainty** option with the parsing frequency equal to the parsing length minus twice the alignment uncertainty. The parsing length was chosen to be 27 seconds (810 frames). That number minus twice the alignment uncertainty means a parsing frequency of 15 seconds (450 frames) must be used. The processed file will be parsed into 27-second segments, each overlapping by 12 seconds. Video quality measurements will use the middle 15 seconds (450 frames) of each processed segment, and the 15 second segment of the 27-second original segment that provides the best alignment. Therefore, when considered as a whole, the quality predictions will cover the entire file, continuously, with two exceptions: (1) the first 6 seconds of the processed file (i.e., the alignment uncertainty) will never be processed, and (2) some video at the end of the processed file, something greater than or equal to the alignment uncertainty (6 seconds), will also be left unprocessed.

All calibration will be done automatically. Due to the possibility of dynamically changing delay, the frame-based temporal registration algorithm was chosen. After calibration, video quality will be measured using the general model. Results will be emailed to jsmith@its.bldrdoc.gov.

Email Address:	jsmith@its.bldrdoc.gov
Original File:	original.yuv
Original Bytes	152064000
Number Processed:	3
Processed File:	processed_1.yuv
Processed Bytes:	152064000
Processed File:	processed_2.yuv
Processed Bytes:	152064000
Processed File:	processed_3.yuv
Processed Bytes:	152064000

Is Compressed:	False
Image Rows:	240
Image Cols:	352
Timing Format:	NTSC
Scanning Pattern:	Progressive
Parsing Type:	SPLIT
Parsing Length:	810
Parsing Frequency:	450
Spatial Calibration:	AUTOMATIC
Valid Calibration:	AUTOMATIC
Gain Calibration:	AUTOMATIC
Temporal Calibration:	AUTOMATIC
Temporal Algorithm:	Frame
Temporal InvalidUncertainty:	True
Alignment Uncertainty:	180
Calibration Frequency:	15
Video Model:	general_model

4.4 SIF, Split Parsed, 15 Frames Per Second, Maximal Content Segments, Values Calibration

This control file lists one original video sequence and two processed video sequences. All video files are in the Big YUV format and contain 600 frames. The files will be parsed with a static parse, because the delay is not expected to increase or decrease steadily over the course of the file. From any one segment to any other, the temporal alignment is expected to change no more than 5 seconds, and so an alignment uncertainty of 180 frames (6 seconds) is used. The extra second was added for safety. In this case, the video files contain progressive images that are 352 pixels by 240 lines. These are SIF images, to be displayed at 15 frames per second, and so the "HALF_NTSC" timing option is selected.

Let us presume that an examination of results generated from another control file has led to the conclusion that the automatic calibration failed for two of the processed clips. Therefore, manual entry of calibration will now be requested for the spatial, valid, and gain calibration and these processed clips will be rerun. The spatial registration results from the earlier control file indicate that both fields of the processed image have been shifted down one field line, and to the left three pixels. The valid video region was (2, 8), (238, 350) where the bottom-right rectangle coordinate (238, 350) is excluded, and the top-left rectangle coordinate (2, 8) is included. A gain (1.04) and offset (10.0) will be removed from each processed pixel, such that the modified pixel value will be the processed pixel value minus the level offset, all divided by the gain. No temporal registration will be required, since the files all aligned perfectly under the assumption that the first frames in each file align, so the temporal shift will be set to zero.

After calibration, video quality will be measured using the developer model.

Email Address:	jsmith@its.bldrdoc.gov
Original File:	original.yuv
Original Bytes	101376000
Number Processed:	2
Processed File:	processed_1.yuv
Processed Bytes:	101376000
Processed File:	processed_2.yuv
Processed Bytes:	101376000

Is Compressed:	False
Image Rows:	240
Image Cols:	352
Timing Format:	HALF_NTSC
Scanning Pattern:	Progressive
Parsing Type:	SPLIT
Parsing Length:	810
Parsing Frequency:	450
Spatial Calibration:	VALUES
Spatial F1Vertical:	1
Spatial F2Vertical:	1
Spatial Horizontal:	-3
Valid Calibration:	VALUES
Valid Top:	2
Valid Left:	8
Valid Bottom:	238
Valid Right	350
Gain Calibration:	VALUES
Luminance Gain:	1.04
Luminance Offset:	10.0
Temporal Calibration:	VALUES
Temporal Shift:	0
Alignment Uncertainty:	180
Calibration Frequency:	15
Video Model:	developer_model

5. VQM PROGRAM OUTPUT

The VQM program is intended to be run in the background, like a batch job. The VQM program does not inform the user by way of standard out on progress, time remaining, calibration values, or model predictions. The VQM program does not produce any permanent output files in the directory in which it was run.

The output of the tool depends upon what happened during calibration and processing. This section describes the types of files that are emailed to the **Email Address** listed in the control file. These emails are sent throughout the program's run time, and thus serve to keep the user updated on the status of the program. The emailed files, viewed as a whole, serve to describe what happened during the video quality measurement process as well as listing calibration values, video quality model values, time histories of parameters, and errors that were encountered during processing. The log files contain text that is designed for easy reading.

The VQM program's run time cannot easily be predicted. Run time depends primarily upon the length and number of video files, image size, the scene content, the calibration options selected, and the model selected. On the development platform (an R12000 SGI Octane) given a pair of 10-second video files containing 720 pixels by 486 lines, the VQM program can run automatic calibration and the general model in approximately 20 minutes.

5.1 Begin File

The user will receive a begin file with a subject title that begins with the word "Begin". This file lets the user know that processing has begun and summarizes the processing requests given in the control file.

5.2 Log File

The user will receive one or more log files with a subject title that begins with the words “Log of”. These individual log files detail the processing request, calibration results, video quality parameters, and overall video model values. If an error occurs during processing, the associated error message will also be included in the log file.

If the video sequences were not parsed, then the user will receive separate log files for each processed video clip in the control file. If the video sequences were parsed, the user will receive a separate log file for each time segment. For example, if a five-minute video sequence is parsed into ten segments, the user will receive separate emails for each of those ten segments. In this case, the email subject titles will also include the segment number, where #1 is the first segment. Whenever a processed video file is parsed, the user also receives an extra log file that delineates the parsing process. This extra log file provides the information for determining the time (i.e., frame) relationships between the parsed segments and the original and processed video files.

5.3 Time History File

The user will receive a time history file with a subject title that begins with the words “Time History of”. For each segment processed, the operator is emailed a time history of all the individual parameters that are included in the **Video Model**. When parsing has been selected, care must be taken if the user seeks to piece together a time history for the entire, unparsed, video sequence. See the control file examples for advice. These parameter time histories are detailed supplemental information for advanced analysis.

5.4 Fatal Control File

If the VQM program cannot execute because of errors encountered in the control file, the user will receive a file with the subject title “Fatal Control File Error”. This file will provide specific details about the error encountered in reading or implementing the control file commands.

5.5 Calibration Warnings File

Calibration problems cause a separate calibration analysis report to be emailed with the subject title that begins with the words “Calibration Warnings”. This report will, for an individual clip, list calibration algorithm errors and warn of suspicious calibration values.

6. INTERPRETING RESULTS

6.1 Calibration

The automated calibration algorithms in the VQM program are quite robust. However, there is always the possibility that these algorithms will fail for some scenes or video systems. When testing a video system, it is wise to compare calibration results for all video segments that are processed. The calibration values should match expectations. The two types of calibration results that require extra explanation at this time are spatial registration and temporal registration.

6.1.1 Spatial Registration

For interlace systems, spatial registration will be reported as three numbers: horizontal shift, field one vertical shift, and field two vertical shift. The horizontal shift is in pixels and vertical shifts are in field lines – not frame lines, as might be expected. Either the vertical shifts should be identical to each other, or the field two vertical shift should be equal to field one vertical shift plus one. If the latter is true, then the video system under test has re-framed the video (i.e., original field one became processed field two, and original field two became the next processed field one). Any other combination of numbers is indicative of a problem with the processed video itself. For instance, the two fields may be spatially mis-aligned and the video picture will look like it bounces up and down.

For progressive systems, spatial registration will be reported as two numbers: horizontal shift and vertical shift. The horizontal shift is in pixels and the vertical shift is in frame lines.

As was mentioned earlier, under normal circumstances, any given video system will have one spatial registration. However, the horizontal shift may alternate between two consecutive integers (e.g., 9 and 10) if the true horizontal shift is between the two numbers. The VQM program only aligns to the nearest pixel.

Spatial registration results that change significantly from clip to clip are probably indicative of a problem. If the problem is inherent to the video system being tested, then the video may look like it wanders around the screen. This is very unlikely, because a viewer would surely object. Another video system impairment that may cause wandering spatial registration results occurs if the processed video has been spatially scaled. Spatial re-scaling is currently not supported by the VQM program. If some segments are very impaired, a false spatial registration may result, or the spatial registration routine may fail. In the case of failure, a zero spatial shift is assumed.

6.1.2 Temporal Registration

For interlace systems, the temporal shift corresponds to the shift in *fields* between the processed segment and the original segment. If spatial registration does not indicate reframing, only even numbered temporal shifts are considered. If spatial registration indicates reframing, only odd numbered temporal shifts are considered. In this second case, the processed video will always be reframed, never the original video. When interpreting results for parsed video files, remember that the temporal shift is referenced to the current segment, namely after parsing. For progressive systems, the temporal shift corresponds to the shift in *frames* between the processed segment and the original segment.

Temporal registration is reported as an integer, which may be either positive, negative, or zero. A positive temporal shift corresponds to discarding that number of images (fields for interlace, frames for progressive) from the beginning of the processed video file. A negative temporal shift corresponds to discarding that number of images (fields for interlace, frames for progressive) from the beginning of the original video file. Likewise, a positive temporal shift also means that the same number of images must be discarded from the end of the original video file, and a negative temporal shift also means that the same number of images must be discarded from the end of the processed video file.

For interlaced video systems that have reframed the video (i.e., temporal shifts of an odd number of fields), there is one more complication that must be remembered when using temporal shifts to figure out which video fields were processed. Extra fields are first discarded from the processed video file (one at the beginning and one at the end). Then, one extra original video frame is discarded from either the beginning (if the temporal shift is negative) or the end (if the temporal shift is positive). This is done because the processed video is reframed, not the original video.

There may be some *a priori* expectations about how much dynamic time warping the video system under test produces. In this case, one should examine the time shifts to be sure they are reasonable.

The automatic temporal registration algorithms will be more accurate with longer clips. If the specified alignment uncertainty is too small, the automatic temporal registration algorithms will not be able to determine the correct temporal shift. When a **RUNNING** parse is selected, these types of errors can compound, and as a result there will be little chance of finding the correct temporal alignment for the later time segments. One indication of this problem will be much lower video quality than what is expected. The solution is to rerun with a larger **Alignment Uncertainty**.

6.2 Video Model

The accuracy of the video model for a given video system will increase when results from multiple video clips are averaged (see section 9 of [2]). By averaging over all video clips that were sent through a particular video system, an improved estimate for the quality of that video system can be obtained. When comparing two or more video systems, the fairest comparison is to pass the same group of scenes through both video systems and then compare these results.

7. REFERENCES

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

[2] S. Wolf and M. Pinson, “Video quality measurement techniques,” NTIA Report 02-392, June 2002.

APPENDIX: MATLAB PROGRAMS FOR READING AND DISPLAYING BIG YUV FILES

To run the enclosed routines, store the functions “disp_bigyuv” and “roi_yuv” as separate m files. Function “disp_bigyuv” displays a sequence of stored YUV images in a big YUV file. Function “roi_yuv” supports and is required by “disp_bigyuv”.

The following are some examples of how disp_bigyuv can be called from the MATLAB prompt:

disp_bigyuv (‘c:\bigyuv_file.yuv’, ‘y’, 1, 1) to display the first Y image with 486 rows and 720 columns.

disp_bigyuv (‘c:\bigyuv_file.yuv’, ‘cb’, 1, 10, 288, 352) to display the first 10 Cb images, where each image has 288 rows and 352 columns.

disp_bigyuv (‘c:\bigyuv_file.yuv’, ‘cr’, 10, 100, 486, 720, 4, 9, 480, 704) to display a 480 row by 704 pixel region of interest out of a total possible size of 486 rows and 720 columns, beginning at line 4 and pixel 9. A total of 100 Cr images in the Big YUV file will be displayed starting at image number 10 in the Big YUV file.

A1 FUNCTION DISP_BIGYUV.M

```
function disp_bigyuv(infile, img, beg_image, num_images, varargin)
%  DISP_BIGYUV(INFILE, IMG, BEG_IMAGE, NUM_IMAGES, varargin)
%
%  Displays a sequence of stored yuv images in bigyuv file INFILE, where
%  BEG_IMAGE is the integer number of the first image in the bigyuv file to display, and
%  NUM_IMAGES is the total number of images to display.  IMG specifies which image
%  to display:  y, cb, cr, or ycbcr (color image).  For the cb and cr images, pixels are
%  replicated by two horizontally so they are the same size as the y image.
%
%  If varargin is present, it must have length = 2 or 6.  If the length is 2, then the program
%  expects to receive the image size (num_rows and num_cols, in that order).  If the length is 6,
%  the program expects to also receive the region of interest (ROI) defined
%  by the upper left corner (ROW_START, COL_START) and the size (ROW_SIZE, COL_SIZE), in
%  that order.  The default image size is 486 rows and 720 columns and the default ROI is the
%  whole image.
%

%  Assign defaults
num_rows = 486;
num_cols = 720;
row_start = 1;
col_start = 1;
row_size = num_rows;
col_size = num_cols;

if (length(varargin) == 2);
   num_rows = varargin{1};
   num_cols = varargin{2};
   row_size = num_rows;
   col_size = num_cols;
end

if (length(varargin) == 6);
   num_rows = varargin{1};
   num_cols = varargin{2};
   row_start = varargin{3};
   col_start = varargin{4};
   row_size = varargin{5};
   col_size = varargin{6};
end

fid = fopen(infile, 'r');
```

```
%  Skip to the first image
for j = 1:beg_image-1
   y = fread(fid, [2*num_cols,num_rows], 'uint8');
end

%  Could use fseek to skip to correct point in file instead
% if (beg_image > 1)
%     if (fseek(fid, 2*num_cols*num_rows*(beg_image-1),-1) == -1)
%         disp('Error positioning file pointer')
%         return
%     end
% end

for j = beg_image:beg_image+num_images-1

   %  Read the image, pixel replicate Cb and Cr by factor of two horizontally
    y = fread(fid, [2*num_cols,num_rows], 'uint8');
    y = reshape(y', num_rows, 2, num_cols);
    cb = y(:, 1, :);
    y = y(:, 2, :);
    y = squeeze(y);
    cb = squeeze(cb);
    cr=cb;
    for i=1:2:num_cols
    cb(:,i+1) = cb(:,i);
    cr(:,i) = cr(:,i+1);
    end

   %  Select the region of interest to display
   [y,cb,cr] = roi_yuv(y, cb, cr, row_start, col_start, row_size, col_size);
   [new_num_rows,new_num_cols] = size(y);

   %  display the roi
   if (j==beg_image)
      figure('Units', 'pixels', 'Position', [100 100 new_num_cols new_num_rows]);
      colormap(gray(256));
      set(gca, 'Position', [0 0 1 1]);

      switch img
         case {'y','Y'}
            h = image(y, 'EraseMode', 'none');
         case {'cb','CB','Cb'}
            h = image(cb, 'EraseMode', 'none');
         case {'cr','CR','Cr'}
            h = image(cr, 'EraseMode', 'none');
        case {'YCbCr','ycbcr','YCBCR'}
            % convert to RGB computer
            y2 = (y-16)*1.164;
            cb2 = (cb-128)*2.009;
            cr2 = (cr-128)*1.589;
            red = cr2+y2;
            green = y2-0.194*cb2-0.509*cr2;
            blue = cb2+y2;
            red = round(red);
            green = round(green);
            blue = round(blue);
            red(find(red<0)) = 0.0;
            green(find(green<0)) = 0.0;
            blue(find(blue<0)) = 0.0;
            red(find(red>255)) = 255.0;
            green(find(green>255)) = 255.0;
            blue(find(blue>255)) = 255.0;
            rgb = cat(3,red/255,green/255,blue/255);
            h = image(rgb, 'EraseMode', 'none');
         otherwise
            disp('unknown image type');
      end
   else
      switch img
         case {'y','Y'}
            set(h, 'CData', y)
            drawnow
         case {'cb','CB','Cb'}
            set(h, 'CData', cb)
            drawnow
         case {'cr','CR','Cr'}
            set(h, 'CData', cr)
            drawnow
         case {'YCbCr','ycbcr','YCBCR'}
         % convert to RGB computer, same algorithm as ycbcr_to_rgb
            y2 = (y-16)*1.164;
            cb2 = (cb-128)*2.009;
            cr2 = (cr-128)*1.589;
            red = cr2+y2;
            green = y2-0.194*cb2-0.509*cr2;
            blue = cb2+y2;
            red = round(red);
            green = round(green);
            blue = round(blue);
            red(find(red<0)) = 0.0;
            green(find(green<0)) = 0.0;
            blue(find(blue<0)) = 0.0;
            red(find(red>255)) = 255.0;
            green(find(green>255)) = 255.0;
            blue(find(blue>255)) = 255.0;
            rgb = cat(3,red/255,green/255,blue/255);
            set(h, 'CData', rgb)
            drawnow
      end
   end
end
fclose(fid);
```

A2 FUNCTION ROI_YUV.M

```
function [varargout] = roi_yuv(y_in, cb_in, cr_in, row_start, col_start, row_size, col_size)
%  [varargout] = ROI_YUV(Y_IN, Cb_IN, Cr_IN, ROW_START, COL_START, ROW_SIZE, COL_SIZE)
%
%  Given an input SDI (Rec. ITU-R 601) Y_IN/Cb_IN/Cr_IN image, select a region of interest
%  (ROI) where the upper left corner of the ROI is (ROW_START, COL_START) and the ROI size
%  is (ROW_SIZE, COL_SIZE).  ROW_START and COL_START should be odd.
%
%  Y_IN, Cb_IN, and Cr_IN must all be the same size.
%
%  If one output argument is requested, returns [Y_OUT] image.
%
%  If two output arguments are requested, returns [Cb_OUT, Cr_OUT] images.
%
%  If three output arguments are requested, returns [Y_OUT, Cb_OUT, Cr_OUT] images.
%

num_rows_in = size(y_in, 1);
num_cols_in = size(y_in, 2);
y = zeros(row_size, col_size);
cb = zeros(row_size, col_size);
cr = zeros(row_size, col_size);

y = y_in(row_start:row_start+row_size-1,col_start:col_start+col_size-1);
cb = cb_in(row_start:row_start+row_size-1,col_start:col_start+col_size-1);
cr = cr_in(row_start:row_start+row_size-1,col_start:col_start+col_size-1);

%  Want y
if (nargout == 1)
   varargout{1} = y;
end

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

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