Question(s):	12	**Meeting, date:**	Geneva, February 2-6, 2009
Study Group:	9 **Working Party:**	**Intended type of document** (R-C-TD):	C
Source:	USA (Proposed)		
Title:	Reference Algorithm for Computing Peak Signal to Noise Ratio (PSNR) of a Video Sequence with a Constant Delay		

Contact:	Stephen Wolf NTIA/ITS USA	Tel: +1 303-497-3771 Fax: +1 303-497-5969 Email: swolf@its.bldrdoc.gov
Contact:	Margaret Pinson NTIA USA	Tel: +1 303-497-3579 Fax: +1 303-497-5969 Email: mpinson@its.bldrdoc.gov

Introduction

Peak Signal to Noise Ratio (PSNR) has been used as a benchmark to evaluate new objective perceptual video quality metrics. For example, PSNR has been used as a benchmark for both the Multimedia (MM) and Reduced Reference Television (RRTV) test programs recently completed by the Video Quality Experts Group (VQEG). However, there is not currently an international Recommendation specifying exactly how to perform this critical measurement. Since the calculation of PSNR is highly dependent upon proper calculation of spatial alignment, temporal alignment, gain, and level offset between the processed video sequence and the original video sequence, one must also specify the method of performing these calibration procedures.

The past two validation tests (MM and RRTV) performed by VQEG utilized the exhaustive search PSNR algorithm that is the subject of this contribution. Members of VQEG agreed to use this PSNR method as a benchmark for assessing the effectiveness of perceptual video quality metrics after extensive discussions. This PSNR calculation method has the advantage of automatically determining the highest possible PSNR value for a given video sequence over the range of spatial and temporal shifts. Only one temporal shift is allowed for all frames in the entire processed video sequence (i.e., constant delay).

Proposal

We propose that the reference algorithm and code presented herein for computing PSNR be made an ITU-T Recommendation to facilitate standardized use by industry and standards organizations.

Reference Algorithm for Computing Peak Signal to Noise Ratio (PSNR) of a Video Sequence with a Constant Delay

Summary

Peak Signal to Noise Ratio (PSNR) is a useful benchmark for evaluating performance improvements of new objective perceptual video quality metrics. For example, PSNR has been used as a benchmark for both the Multimedia (MM) and Reduced Reference Television (RRTV) test programs recently completed by the Video Quality Experts Group (VQEG). Since the calculation of PSNR is highly dependent upon proper estimation of spatial alignment, temporal alignment, gain, and level offset between the processed video sequence and the original video sequence, the method of measurement for PSNR should ideally include a method for performing these calibration procedures. This PSNR calculation method in this Recommendation has the advantage of automatically determining the highest possible PSNR value for a given video sequence over the range of spatial and temporal shifts. Only one temporal shift is allowed for all frames in the entire processed video sequence (i.e., constant delay).

This Recommendation define a Full Reference (FR) algorithm for computing both the calibration and PSNR estimations for a processed video sequence. Since the PSNR algorithm only examines the Y luminance channel (as defined by ITU-R Recommendation BT.601), distortions in the C_B and C_R chrominance channels will not be detected by the algorithm of this Recommendation. The intent of this Recommendation is to define and facilitate a standardized PSNR metric for use by industry and standards organizations. Reference code and test vectors have been included to assure accurate and consistent implementation of the PSNR metric.

1 Scope

The PSNR algorithm performs an exhaustive search for the maximum Y-channel PSNR over plus or minus the horizontal and vertical spatial uncertainties (in pixels) and plus or minus the temporal uncertainty (in frames). The processed video segment is fixed and the original video segment is shifted over the search range. For each spatial-temporal (ST) shift, a linear fit between the Y-channel processed and the original pixels is performed such that the Mean Squared Error (MSE) of original - (gain*processed + offset) is minimized, hence maximizing PSNR. Thus, this calculation of PSNR yields values that are greater than or equal to commonly used PSNR implementations if the exhaustive search covered enough ST shifts.

The ST search range, Spatial Region of Interest (SROI), and Temporal Region of Interest (TROI) are input parameters to the algorithm and the user of this Recommendation is advised to carefully consider appropriate input arguments for these quantities. For instance, some video systems truncate the border picture elements replacing these pixels with black. In these cases, the SROI should be reduced so as to not include these pixels in the PSNR calculation.

Since the ST search performed by the algorithm only considers integer shifts in space and time, this method is not appropriate for video systems that contain sub-pixel shifts. Caution should also be observed when analyzing video frames that contain interlaced video frames to assure that the interlaced framing of the processed video sequence is identical to the original video sequence before applying the algorithm (i.e., the algorithm does not account for one-field or half frame timing shifts between the processed and original).

Since the video delay is assumed to be constant for all video frames in the processed video segment, this algorithm may not be appropriate for video systems that contain variable video delays (i.e., where the video delay of individual frames may vary). The algorithm may be used in these cases if one desires the PSNR calculation to include distortions due to variable video delays.

Common applications of the PSNR algorithm include:

A reference benchmark for evaluating the effectiveness of perceptual quality metrics.

A FR Quality of Service metric (QoS) for video transmission systems.

2 References

None

3 Definitions

None

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

PSNR	Peak Signal to Noise Ratio
FR	Full Reference
MSE	Mean Squared Error
QoS	Quality of Service
SROI	Spatial Region of Interest
ST	Spatial-Temporal
TROI	Temporal Region of Interest

5 Conventions

None

6 PSNR Algorithm Description

PSNR is defined as $10*\log_{10}$ of the ratio of the peak signal energy to the MSE observed between the processed video signal and the original video signal. For the algorithm presented here, the peak signal energy is assumed to be 255^2, and the MSE summation is performed over the selected SROI and TROI of the processed video sequence. The algorithm performs a linear fit of the processed image pixels to the corresponding original image pixels for each ST shift that is examined before computing the MSE. This is equivalent to removing gain (contrast) and level offset (brightness) calibration errors in the processed video before performing the PSNR calculation.

Computation of PSNR for an (original, processed) video clip pair involves the following steps:

1. Determine the appropriate ST search range for the processed video clip. This involves estimating the *x* and *y* spatial uncertainty (in pixels, denoted here as *x_uncert* and *y_uncert*) of spatial registration errors that might be present, as well as estimating the *t* temporal uncertainty (in frames, denoted as *t_uncert*) of any temporal registration errors that might be present. Since the algorithm will perform an exhaustive search over plus or minus *x_uncert*, *y_uncert*, and *t_uncert*

shifts of the original video sequence with respect to the processed video sequence, these estimates should be as tight as possible while still including the optimal ST registration.

2. Determine the maximum SROI and TROI that can be used for the processed video clip. If the processed video clip contains truncation of border pixels (i.e., black border), the SROI of the processed video clip should be reduced to eliminate these pixels. If the processed video clip contains transition video frames at the beginning or end of the video clip (perhaps due to prior or following scene content), these frames should be eliminated from the TROI. If necessary, reduce the maximum SROI and TROI to allow for the *x_uncert*, *y_uncert*, and *t_uncert* shifts found in step 1. The final SROI and TROI of the processed video clip that is determined by this step will remain fixed for all PSNR calculations. Since the original video clip will be shifted by a maximum of plus or minus *x_uncert* and *y_uncert* pixels and plus or minus *t_uncert* frames with respect to the processed video clip, one must assure that there are valid original video pixels that align to every processed video pixel within the final SROI and TROI.

3. For each ST shift of the original sequence in step 1, (i.e., shifts in the x, y, and t directions will be denoted here as x_S, y_S, and t_S, respectively), perform a linear fit of the processed pixels to the shifted original pixels. This linear fit is performed for all pixels in the entire ST region encompassed by the processed video SROI and TROI selected in step 2. For a given ST shift, this can be expressed as finding the $Gain(x_S,y_S,t_S)$ and $Offset(x_S,y_S,t_S)$ that minimizes the MSE given by:

$$\mathrm{MSE} = \frac{1}{N}\sum_x\sum_y\sum_t \{O(x+x_s, y+y_s, t+t_s) - [Gain(x_s,y_s,t_s) * P(x,y,t) + Offset(x_s,y_s,t_s)]\}^2; (x,y) \in \mathrm{SROI}, t \in \mathrm{TROI}$$

where three dimensional matrices O and P represent the original and processed video sequences, respectively, the MSE is computed over all x, y, and t that belong to SROI and TROI, and N is the total number of pixels in the three dimensional processed video segment encompassed by SROI and TROI.

4. Compute the MSE in step 3 for all ST shifts within the spatial and temporal uncertainties defined in step 1 (i.e., $-x\_uncert \le x_S \le x\_uncert$, $-y\_uncert \le y_S \le y\_uncert$, and $-t\_uncert \le t_S \le t\_uncert$) and select the minimum MSE (i.e., MSE_{min}). This is the MSE that will maximize the PSNR, defined by

$$\mathrm{PSNR} = 10 * \log_{10}\left(\frac{255^2}{\mathrm{MSE}_{min}}\right)$$

Reference code and test vectors to implement the PSNR algorithm given above is provided in Appendix I.

Appendix I

Reference Code to Calculate PSNR

(This appendix does not form an integral part of this Recommendation)

This appendix contains reference MATLAB® code and test files for calculating PSNR. Since the entire processed and original video sequences are held in memory in double precision format, a 64-bit operating system with at least 4 GB of available memory is recommended for ITU-R Recommendation BT.601 sampled video sequences that are 8-10 seconds in length. More memory will be required to process high definition video streams.

Two functions are provided in this Appendix; a function that performs the exhaustive PSNR search (psnr_search) and a function to read Big-YUV video files (read_bigyuv). See the psnr_search function documentation for a description of the Big-YUV file format and the input parameters that control the exhaustive search. This reference code assumes that the original and processed Big-YUV files contain the same number of video frames.

The psnr_search function can process all the video clips stored in a user-specified directory. When the PSNR test vectors (i.e., video clips) are stored in a directory and the psnr_search function is called with the following arguments, the exhaustive PSNR search algorithm should output the following results, with the final line printout for each test vector being the maximum PSNR that was found over the ST search range:

```
results = psnr_search ('c:\psnr\','psnr','yuv',144,176,'sroi',5,5,140,172,'spatial_uncertainty',1,1,'temporal_uncertainty',8,'verbose');

Test = psnr,   Scene = calmob,   HRC = hrc2
dy = -1, dx = -1, dt = -8, gain = 0.6593, off = 49.3691, PSNR = 15.8660
dy =  0, dx = -1, dt = -8, gain = 0.6985, off = 43.8486, PSNR = 16.2311
dy =  1, dx = -1, dt = -8, gain = 0.7379, off = 38.2988, PSNR = 16.6628
dy =  1, dx = -1, dt = -7, gain = 0.7720, off = 33.3844, PSNR = 17.1414
dy =  1, dx = -1, dt = -6, gain = 0.8090, off = 28.0591, PSNR = 17.7597
dy =  1, dx = -1, dt = -5, gain = 0.8438, off = 23.0324, PSNR = 18.4701
dy =  1, dx = -1, dt = -4, gain = 0.8682, off = 19.5165, PSNR = 19.0599
dy =  1, dx = -1, dt = -3, gain = 0.8726, off = 18.8568, PSNR = 19.1561
dy =  0, dx = -1, dt = -2, gain = 0.8913, off = 16.0535, PSNR = 19.8223
dy =  0, dx = -1, dt = -1, gain = 0.8956, off = 15.4151, PSNR = 19.9367
dy =  0, dx =  0, dt = -1, gain = 0.9493, off = 7.6633, PSNR = 22.4131
dy =  0, dx =  0, dt =  0, gain = 0.9907, off = 1.6981, PSNR = 26.6079
HRC = hrc2, psnr_ave = 26.6079

Test = psnr,   Scene = flogar,   HRC = hrc1
dy = -1, dx = -1, dt = -8, gain = 0.8117, off = 30.9903, PSNR = 15.9486
dy =  0, dx = -1, dt = -8, gain = 0.8455, off = 24.9600, PSNR = 16.4640
dy = -1, dx =  0, dt = -8, gain = 0.8433, off = 25.9047, PSNR = 16.5596
dy =  0, dx =  0, dt = -8, gain = 0.8899, off = 17.7494, PSNR = 17.5184
dy =  0, dx =  1, dt = -8, gain = 0.8994, off = 16.2670, PSNR = 17.8159
```

dy = 0, dx = 0, dt = -7, gain = 0.9014, off = 15.8865, PSNR = 17.8463
dy = 0, dx = 1, dt = -7, gain = 0.9133, off = 14.0029, PSNR = 18.2496
dy = 0, dx = 0, dt = -6, gain = 0.9156, off = 13.5618, PSNR = 18.2970
dy = 0, dx = 1, dt = -6, gain = 0.9287, off = 11.4980, PSNR = 18.7934
dy = 0, dx = 0, dt = -5, gain = 0.9334, off = 10.6651, PSNR = 18.9420
dy = 0, dx = 1, dt = -5, gain = 0.9437, off = 9.0428, PSNR = 19.4130
dy = 0, dx = 0, dt = -4, gain = 0.9537, off = 7.3531, PSNR = 19.8371
dy = 0, dx = 1, dt = -4, gain = 0.9550, off = 7.1976, PSNR = 19.9555
dy = 0, dx = 0, dt = -3, gain = 0.9730, off = 4.2002, PSNR = 20.9305
dy = 0, dx = 0, dt = -2, gain = 0.9863, off = 2.0200, PSNR = 21.9130
dy = 0, dx = 0, dt = -1, gain = 0.9909, off = 1.2905, PSNR = 22.3187
HRC = hrc1, psnr_ave = 22.3187

PSNR Search Function

```
function [results] = psnr_search(clip_dir, test, varargin)
% PSNR_SEARCH
%   Estimate the Y-channel PSNR (PSNR) of all clips and HRCs (Hypothetical
%   Reference Circuits) in a video test (input argument test) where the
%   video clips are stored in the specified directory (clip_dir).  The
%   video clips must have names that conform to the naming convention
%   test_scene_hrc.yuv, with no extra '_' or '.' in the file names.  "test"
%   is the name of the test, "scene" is the name of the scene, and "hrc" is
%   the name of the HRC.  The name of the original reference clip for the
%   PSNR calculation must be "test_scene_original.yuv".
%
%   Files must be stored in "Big YUV format", which is a binary format for
%   storing ITU-R Recommendation BT.601 video sequences.  The format can
%   be used for any image size.  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: Cb1 Y1 Cr1 Y2 Cb3 Y3 Cr3 Y4…, 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. For example, Y=15 may be produced by a real system, but in
%   any case, Y, Cb, and Cr values are always between 0 and 255.  The
%   original and processed Big YUV files must have the same number of frames.
%
%   A peak signal of 255 is used for calculation of PSNR.  Double precision
%   calculations are used everywhere.  A 64-bit operating system with at
%   least 4 GB of free memory is recommended since the entire double
%   precision versions of the original and processed sequences must be held
%   in memory.
%
% SYNTAX
```

```
%   [results] = psnr_search('clip_dir', 'test', option);
%
% DESCRIPTION
%   This function will process all video clips in the user specified
%   clip_dir and test, estimate the Y-channel PSNR of each clip, and then
%   average these clip results to produce an estimate for each HRC.  The
%   algorithm performs an exhaustive search for max PSNR over plus or minus
%   the spatial_uncertainty (in pixels) and plus or minus the
%   temporal_uncertainty (in frames).  The processed video segment is fixed
%   and the original video segment is shifted over the search range.  For
%   each spatial-temporal shift, a linear fit between the processed pixels
%   and the original pixels is performed such that the mean square error of
%   [original-gain*processed+offset] is minimized (hence maximizing PSNR).
%
%   Any or all of the following optional properties may be requested (the
%   first option is required for yuv files).
%
%   'yuv',rows,cols    Specifies the number of rows and cols for the Big
%                      YUV files.
%
%   'sroi',top,left,bottom,right,   Only use the specified spatial region
%                                   of interest (sroi) for the PSNR
%                                   calculation.  This is the sroi of the
%                                   processed sequence, which remains fixed
%                                   over all spatial shifts.  By default,
%                                   sroi is the entire image reduced by the
%                                   spatial uncertainty.  If the user
%                                   inputs a sroi, allowance must be made
%                                   for the spatial search specified by
%                                   'spatial_uncertainty'.
%
%   'frames',fstart,fstop  Only use the frames from fstart to fstop
%                          (inclusive) to perform the PSNR estimate.  This
%                          specifies the temporal segment of the processed
%                          sequence, which remains fixed over all temporal
%                          shifts.  By default, the temporal segment is the
%                          entire file reduced by the temporal uncertainty.
%                          If the user inputs an fstart and fstop,
%                          allowance must be made for the temporal search
%                          specified by 'temporal_uncertainty'.
%
%
%   'spatial_uncertainty',x,y   Specifies the spatial uncertainty (plus
%                               or minus, in pixels) over which to
%                               search.  The processed remains fixed and
%                               the original is shifted.  By default,
%                               this is set to zero.
%
%   'temporal_uncertainty',t  Specifies the temporal uncertainty
%                             (plus or minus, in frames) over which
```

```
%                            to search.  The processed remains fixed
%                            and the original is shifted.  By
%                            default, this is set to zero.
%
%   'verbose'   Display output during processing.
%
%
%   The returned variable [results] is a struct that contains the following
%   information for each processed clip i:
%
%   results(i).test    The test name for the video clip.
%   results(i).scene   The scene name for the video clip.
%   results(i).hrc     The HRC name for the video clip.
%   results(i).yshift  The y shift for max PSNR.
%   results(i).xshift  The x shift for max PSNR.
%   results(i).tshift  The time shift for max PSNR.
%   results(i).gain    The gain*processed+offset for max PSNR.
%   results(i).offset
%   results(i).psnr    The maximum PSNR observed over the search.
%
% EXAMPLES
%   These examples illustrate how to call the routine to process the VQEG
%   MM Phase I test scenes, where test scenes from each subjective
%   experiment are stored in a unique directory.  These sequences must
%   first be converted from AVI format to Big YUV format.
%
%   q01 = psnr_search('d:\q01\','q01','yuv',144,176,'sroi',5,5,140,172,...
%       'spatial_uncertainty',1,1,'temporal_uncertainty',8,'verbose');
%   c01 = psnr_search('d:\c01\','c01','yuv',288,352,'sroi',8,8,281,345,...
%       'spatial_uncertainty',1,1,'temporal_uncertainty',8,'verbose');
%   v01 = psnr_search('d:\v01\','v01','yuv',480,640,'sroi',14,14,467,627,...
%       'spatial_uncertainty',1,1,'temporal_uncertainty',8,'verbose');
%

% Define the peak signal level
peak = 255.0;

% Add extra \ in clip_dir in case user did not
clip_dir = strcat(clip_dir,'\');

% Validate input arguments and set their defaults
is_yuv = 0;
is_whole_image = 1;
is_whole_time = 1;
x_uncert = 0;
y_uncert = 0;
t_uncert = 0;
verbose = 0;
dx=1;  % dx, dy, and dt sizes to use for gain and level offset calculations
dy=1;
```

```
dt=1;
cnt=1;
while cnt <= length(varargin),
    if ~isstr(varargin{cnt}),
        error('Property value passed into psnr_search is not recognized');
    end
    if strcmpi(varargin(cnt),'yuv') == 1
        rows = varargin{cnt+1};
        cols = varargin{cnt+2};
        is_yuv = 1;
        cnt = cnt + 3;
    elseif strcmpi(varargin(cnt),'sroi') == 1
        top = varargin{cnt+1};
        left = varargin{cnt+2};
        bottom = varargin{cnt+3};
        right = varargin{cnt+4};
        is_whole_image = 0;
        cnt = cnt + 5;
    elseif strcmpi(varargin(cnt),'frames') == 1
        fstart = varargin{cnt+1};
        fstop = varargin{cnt+2};
        is_whole_time = 0;
        cnt = cnt + 3;
    elseif strcmpi(varargin(cnt), 'spatial_uncertainty') ==1
        x_uncert = varargin{cnt+1};
        y_uncert = varargin{cnt+2};
        cnt = cnt + 3;
    elseif strcmpi(varargin(cnt), 'temporal_uncertainty') ==1
        t_uncert = varargin{cnt+1};
        cnt = cnt + 2;
    elseif strcmpi(varargin(cnt),'verbose') == 1
        verbose = 1;
        cnt = cnt +1;
    else
        error('Property value passed into psnr_search not recognized');
    end
end

%  Get a directory listing
files = dir(clip_dir);  % first two files are '.' and '..'
num_files = size(files,1);

% Find the HRCs and their scenes for the specified video test
hrc_list = {};
scene_list = {};
for i=3:num_files
    this_file = files(i).name;
    und = strfind(this_file,'_'); % find underscores and period
    dot = strfind(this_file,'.');
    if(size(und,2)==2) % possible standard naming convention file found
```

```
        this_test = this_file(1:und(1)-1);  % pick off the test name
        if(strmatch(test,this_test,'exact'))  % test clip found
            this_scene = this_file(und(1)+1:und(2)-1);
            this_hrc = this_file(und(2)+1:dot(1)-1);
            % See if this HRC already exists and find its list location
            loc = strmatch(this_hrc,hrc_list,'exact');
            if(loc)  % HRC already present, add to scene list for that HRC
                if(size(strmatch(this_scene,scene_list{loc},'exact'),1)==0)
                    scene_list{loc} = [scene_list{loc} this_scene];
                end
            else  % new HRC found
                hrc_list = [hrc_list;{this_hrc}];
                this_loc = size(hrc_list,1);
                scene_list(this_loc) = {{this_scene}};
            end
        end
    end
end

scene_list = scene_list';
num_hrcs = size(hrc_list,1);

% Results struct to store results, shifts are how much the original must be
% shifted with respect to the processed
results = struct('test', {}, 'scene', {}, 'hrc', {}, 'yshift', {}, ...
    'xshift', {}, 'tshift', {}, 'gain', {}, 'offset', {}, 'psnr', {});

% Process one HRC at a time to compute average PSNR for that HRC
index = 1;  % index to store results
for i = 1:num_hrcs

    psnr_ave = 0;  % initialize the psnr average summer for this HRC
    this_hrc = hrc_list{i};
    if(strmatch('original',this_hrc,'exact')) % Don't process original
        continue;
    end
    num_scenes = size(scene_list{i},2);  % Number of scenes in this HRC

    for j = 1:num_scenes

        this_scene = scene_list{i}{j};
        results(index).test = test;
        results(index).scene = this_scene;
        results(index).hrc = this_hrc;

        % Read original and processed video files
        if (~is_yuv)  % YUV file parameters not specified
            display('Must specify Big YUV rows and cols.  Use yuv input option.');
            return
        else  % YUV file
```

```
% Re-generate the original and processed YUV file name
orig = strcat(clip_dir, test,'_', this_scene, '_', 'original', '.yuv');
proc = strcat(clip_dir, test,'_', this_scene, '_', this_hrc, '.yuv');
% Set/Validate the ROI
if (is_whole_image) % make ROI whole image less uncertainty
    top = 1+y_uncert;
    left = 1+x_uncert;
    bottom = rows-y_uncert;
    right = cols-x_uncert;
elseif (top<1 || left<1 || bottom>rows || right>cols)
    display('Requested SROI too large for image size.');
    return;
end
% Find the total frames of the input original file
[fid, message] = fopen(orig, 'r');
if fid == -1
    fprintf(message);
    error('Cannot open this clip''s bigyuv file, %s', orig);
    return
end
% Find last frame.
fseek(fid,0, 'eof');
tframes = ftell(fid) / (2 * rows * cols);
fclose(fid);
% Find the total frames of the processed file
[fid, message] = fopen(proc, 'r');
if fid == -1
    fprintf(message);
    error('Cannot open this clip''s bigyuv file, %s', proc);
    return
end
% Find last frame.
fseek(fid,0, 'eof');
tframes_proc = ftell(fid) / (2 * rows * cols);
fclose(fid);
% Validate that orig and proc have the same number of frames
if (tframes ~= tframes_proc)
    display('The orig & proc files must have the same length.');
    return
end
% Set/Validate the time segment to use
if (is_whole_time) % use whole time segment less uncertainty
    fstart= 1+t_uncert;
    fstop = tframes-t_uncert;
elseif (fstart<1 || fstop>tframes)
    display('Requested Temporal segment too large for file size.');
    return
end
 % Validate the spatial uncertainty search bounds
if (left-x_uncert < 1 || right+x_uncert > cols)
```

```
        display('Spatial x-uncertainty too large for SROI.');
        return;
    end
    if (top-y_uncert < 1 || bottom+y_uncert > rows)
        display('Spatial y-uncertainty too large for SROI.');
        return;
    end
    % Validate the temporal uncertainty search bounds
    if(fstart-t_uncert < 1 || fstop+t_uncert > tframes)
        display('Temporal uncertainty too large for fstart or fstop.');
        return;
    end
    % Read in video and clear color planes to free up memory
    [y_orig,cb,cr] = read_bigyuv(orig,'frames',fstart-t_uncert,...
        fstop+t_uncert,'size',rows,cols,'sroi',top-y_uncert,...
        left-x_uncert,bottom+y_uncert,right+x_uncert);
    clear cb cr;
    [y_proc,cb,cr] = read_bigyuv(proc,'frames',fstart,fstop,...
        'size',rows,cols,'sroi',top,left,bottom,right);
    clear cb cr;
end

% Convert images to double precision
y_orig = double(y_orig);
y_proc = double(y_proc);

[nrows, ncols, nsamps] = size(y_proc);
% Reshape y_proc for the PSNR calculation:  this stays fixed
y_proc = reshape(y_proc,nrows*ncols*nsamps,1); % make column vector

% Compute PSNR for each spatial-temporal shift
best_psnr = -inf;
best_xshift = 0;
best_yshift = 0;
best_tshift = 0;
best_gain = 1;
best_offset = 0;
if(verbose)
    fprintf('\nTest = %s,   Scene = %s,   HRC = %s\n',test, this_scene, this_hrc);
end

for k = -t_uncert:t_uncert
    for m = -x_uncert:x_uncert
        for n = -y_uncert:y_uncert

            % Perform gain and level offset calculation
            this_fit = polyfit(y_proc,reshape(y_orig(1+n+y_uncert:n+y_uncert+nrows,...
                1+m+x_uncert:m+x_uncert+ncols,1+k+t_uncert:k+t_uncert+nsamps),...
                nrows*ncols*nsamps,1),1);
```

```
                    % Calculate the PSNR
                    this_psnr = 10*(log10(peak*peak)-log10(sum(((this_fit(1)*y_proc+this_fit(2))-...
                     reshape(y_orig(1+n+y_uncert:n+y_uncert+nrows,1+m+x_uncert:m+x_uncert+ncols,...
                     1+k+t_uncert:k+t_uncert+nsamps),nrows*ncols*nsamps,1)).^2)/(nrows*ncols*nsamps)));

                    if(this_psnr > best_psnr)
                        best_psnr = this_psnr;
                        best_yshift = n;
                        best_xshift = m;
                        best_tshift = k;
                        best_gain = this_fit(1);
                        best_offset = this_fit(2);
                        if(verbose)
                            fprintf('dy =%3i, dx =%3i, dt =%3i, gain = %5.4f, off = %5.4f, PSNR = %5.4f\n',...
                            best_yshift,best_xshift,best_tshift,best_gain,best_offset,best_psnr);
                        end
                    end

                end
            end
        end

        results(index).yshift = best_yshift;
        results(index).xshift = best_xshift;
        results(index).tshift = best_tshift;
        results(index).gain = best_gain;
        results(index).offset = best_offset;
        results(index).psnr = best_psnr;
        psnr_ave = psnr_ave+best_psnr;
        index = index+1;

    end

    % Compute average PSNR for this HRC
    psnr_ave = psnr_ave/(num_scenes);
    if(verbose)
        fprintf('HRC = %s, psnr_ave = %5.4f\n',this_hrc, psnr_ave);
    end

end
```

Read Big-YUV Function

```
function [y,cb,cr] = read_bigyuv(file_name, varargin);
% READ_BIGYUV
%   Read images from bigyuv-file.
%
% SYNTAX
%   [y] = read_bigyuv(file_name);
%   [y,cb,cr] = read_bigyuv(...);
```

```
%   [...] = read_bigyuv(...,'PropertyName',PropertyValue,...);
%
% DESCRIPTION
%   Read in images from bigyuv file named 'file_name'.
%
%   The luminance plane is returned in 'Y'; the color planes are
%   returned in 'cb' and 'cr' if requested.  The Cb and Cr color planes
%   will be up-sampled by 2 horizontally.
%
%   The following optional properties may be requested:
%
%   'sroi',top,left,bottom,right,
%                       Spatial region of interest to be returned.  By
%                       default, the entirety of each image is returned.
%                       Inclusive coordinates (top,left),(bottom,right)
%                       start numbering with row/line number 1.
%   'size',row,col,    Size of images (row,col).  By default, row=486,
%                      col=720.
%   'frames',start,stop,    Specify the first and last frames, inclusive,
%                           to be read ('start' and 'stop').  By default,
%                           the first frame is read.
%   '128'    Subtract 128 from all Cb and Cr values.  By default, Cb and
%            Cr values are left in the [0..255] range.
%   'interp'    Interpolate Cb and Cr values.  By default, color planes
%               are pixel replicated.  Note: Interpolation is slow.
%

% read values from clip_struct that can be over written by variable argument
% list.
is_whole_image = 1;
is_sub128 = 0;
is_interp = 0;

num_rows = 486;
num_cols = 720;

start = 1;
stop = 1;

% parse varargin list (property values)
cnt = 1;
while cnt <= nargin - 1,
    if ~isstr(varargin{cnt}),
        error('Property value passed into read_bigyuv not recognized');
    end
    if strcmp(lower(varargin(cnt)),'sroi') == 1,
        sroi.top = varargin{cnt+1};
        sroi.left = varargin{cnt+2};
        sroi.bottom = varargin{cnt+3};
        sroi.right = varargin{cnt+4};
```

```
        is_whole_image = 0;
        cnt = cnt + 5;
    elseif strcmp(lower(varargin(cnt)),'size') == 1,
        num_rows = varargin{cnt+1};
        num_cols = varargin{cnt+2};
        cnt = cnt + 3;
    elseif strcmp(lower(varargin(cnt)),'frames') == 1,
        start = varargin{cnt+1};
        stop = varargin{cnt+2};
        cnt = cnt + 3;
    elseif strcmp(lower(varargin(cnt)),'128') == 1,
        is_sub128 = 1;
        cnt = cnt + 1;
    elseif strcmp(lower(varargin(cnt)),'interp') == 1,
        is_interp = 1;
        cnt = cnt + 1;
    else
        error('Property value passed into read_bigyuv not recognized');
    end
end

if mod(num_cols,2) ~= 0,
    fprintf('Error: number of columns must be even.\nFile format stores 4 bytes for 2 pixels\n');
    error('Invalid specification for argument "num_cols" in read_bigyuv');
end

% Open image file
% [test_struct.path{1} clip_struct.file_name{1}]
[fid, message] = fopen(file_name, 'r');
if fid == -1
    fprintf(message);
    error('read_bigyuv cannot open this clip''s bigyuv file, %s', file_name);
end

% Find last frame.
fseek(fid,0, 'eof');
total = ftell(fid) / (2 * num_rows * num_cols);
if stop > total,
    error('Requested a frame past the end of the file.  Only %d frames available', total);
end
if stop < 0,
    error('Range of frames invalid');
end
if start > stop | stop < 1,
    error('Range of frames invalid, or no images exist in this bigyuv file');
end

% find range of frames requested.
prev_tslice_frames = start - 1;
tslice_frames = stop - start + 1;
```

```
number = start;

% go to requested location
if isnan(start),
    error('first frame of this clip is undefined (NaN).');
end
offset = prev_tslice_frames * num_rows * num_cols * 2; %pixels each image
status = fseek(fid, offset, 'bof');

if status == -1,
    fclose(fid);
    error('read_bigyuv cannot seek requested image location');
end

% initialize memory to hold return images.
y = zeros(num_rows,num_cols,tslice_frames, 'single');

if (nargout == 3),
    cb = y;
    cr = y;
end

% loop through & read in the time-slice of images
this_try = 1;
for cnt = 1:tslice_frames,
    where = ftell(fid);
    [hold_fread,count] = fread(fid, [2*num_cols,num_rows], 'uint8=>uint8');
    if count ~= 2*num_cols*num_rows,
        % try one more time.
        fprintf('Warning: read_bigyuv could not read requested time-slice; re-trying\n');
        %pause(5);
        if where == -1,
            fprintf('Could not determine current location.  Re-try failed.\n');
            error('read_bigyuv could not read entirety of requested image time-slice');
            fclose(fid);
        end
        fseek(fid, where, 'bof');
        [hold_fread,count] = fread(fid, [2*num_cols,num_rows], 'uint8=>uint8');
        if count ~= 2*num_cols*num_rows,
            fclose(fid);
            hold = sprintf('read_bigyuv failed for %s\nCould not read time-slice', file_name);
            error(hold);
        end
    end

    % pick off the Y plane (luminance)
    temp = reshape(hold_fread', num_rows, 2, num_cols);
    uncalib = squeeze(temp(:,2,:));
    y(:,:,cnt) = single(uncalib);
```

```
    % If color image planes are requested, pick those off and perform
    % pixel replication to upsample horizontally by 2.
    if nargout == 3,
        temp = reshape(hold_fread,4,num_rows*num_cols/2);

        color = reshape(temp(1,:),num_cols/2,num_rows)';
        color2 = [color ; color];
        uncalib = reshape(color2,num_rows,num_cols);
        cb(:,:,cnt) = single(uncalib);
        if is_sub128,
            cb(:,:,cnt) = cb(:,:,cnt) - 128;
        end

        color = reshape(temp(3,:),num_cols/2,num_rows)';
        color2 = [color ; color];
        uncalib = reshape(color2,num_rows,num_cols);
        cr(:,:,cnt) = single(uncalib);
        if is_sub128,
            cr(:,:,cnt) = cr(:,:,cnt) - 128;
        end

        % Interpolate, if requested
        if is_interp == 1,
            for i=2:2:num_cols-2,
                cb(:,i,cnt) = (cb(:,i-1,cnt) + cb(:,i+1,cnt))/2;
                cr(:,i,cnt) = (cr(:,i-1,cnt) + cr(:,i+1,cnt))/2;
            end
        end
    end
end

fclose(fid);

if ~is_whole_image,
    y = y(sroi.top:sroi.bottom, sroi.left:sroi.right, :);
    if nargout == 3,
        cb = cb(sroi.top:sroi.bottom, sroi.left:sroi.right, :);
        cr = cr(sroi.top:sroi.bottom, sroi.left:sroi.right, :);
    end
end
```

Bibliography

VQEG Final Report of MM Phase I Validation Test (2008), "*Final report from the Video Quality Experts Group on the validation of objective models of multimedia quality assessment, phase I*", Video Quality Experts Group (VQEG), http://www.its.bldrdoc.gov/vqeg/projects/multimedia, ITU-T Study Group 9 TD923, Study Period 2005-2008.

ITU-R Recommendation BT.601-6 (01/07), "*Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios*".