16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

A COLOUR CORRECTION PREPROCESSING METHOD FOR MULTIVIEW VIDEO CODING

Colin Doutre and Panos Nasiopoulos

Department of Electrical and Computer Engineering, University of British Columbia
2366 Main Mall, V6T 1Z4, Vancouver, BC, Canada
email: (colind, panos)@ece.ubc.ca
web: http://www.ece.ubc.ca/~panosn

ABSTRACT

In multiview video, a number of cameras capture the same scene from different viewpoints. There can be significant variations in the colour of views captured with different cameras, which negatively affects performance when the videos are compressed with inter-view prediction. In this paper, a method is proposed for correcting the colour of multiview video sets as a preprocessing step to compression. The corrected YUV values of a pixel are expressed as a weighted sum of the original YUV values. Disparity estimation is used to find matching points between a reference view and the view being corrected. A least squares regression is performed on these sets of matching points to find the optimal weight parameters that will make the current view most closely match the reference. Experimental results show that the proposed method produces colours that closely match the reference view. Furthermore, when multiview video is compressed with H.264 using inter-view prediction, the proposed method increases compression efficiency by up to 1.0 dB compared to compressing the original uncorrected video.

1. INTRODUCTION

Multi-view video systems have received considerable attention from the research community recently [1]. Advances in display and camera technologies promise to make 3D TV practical for consumer products. To capture a 3D representation of a dynamic scene, multiple video cameras are used, which capture the scene from different viewpoints. Using Image Based Rendering (IBR) techniques [2], virtual viewpoints can be created, allowing the user to view the scene from a range of viewing positions.

Since the number of cameras can be large, and each individual video contains a large amount of data, the total amount of data captured in multi-view video systems is huge. Hence, efficient compression methods are required for practical multiview video systems to allow storage and transmission.

In traditional single view video compression, motion compensation is used to exploit correlation between frames captured at different times. In Multi-view Video Coding (MVC), techniques also exploit redundancies between views captured with different cameras to reduce the number of bits required to store the videos. This is known as disparity compensated coding. In a simple form, disparity compensation can be accomplished by using standard motion compensation techniques between frames in different views.

A challenge that arises in MVC that is not present in traditional single view video coding is the inconsistencies between cameras. It is difficult to perfectly calibrate a number of cameras, so there are always differences in brightness, colour, focus, etc. between the videos captured with different cameras. These inconsistencies reduce the correlation between views, and therefore reduce compression efficiency when one view is predicted based on another. Therefore, it is important in MVC to correct the inconsistencies between cameras, particularly in brightness and colour to improve compression performance. Variations in colour between views also negatively affects rendering of new virtual viewpoints, and will be unpleasant to users as they switch between different views.

In both image and video cameras, colour calibration is usually done by capturing a known reference such as a colour chart [3]. One way to try to calibrate an array of cameras in multiview video systems is to capture a colour chart and calibrate every camera to it [4]. However, this approach is sensitive to light conditions, and it is not always practical to capture a colour chart. In multiview imaging, it is more important that colours are consistent between different cameras, so it is more practical to choose one view as a reference and modify the colour in the other views to match the reference.

In MVC systems, brightness and colour correction can be performed either as preprocessing before compression, or incorporated into the compression process itself. Accounting for colour variations in the compression process itself has the advantage that the original data is restored during the compression process, which gives flexibility for different colour correction methods to be applied after decoding. The disadvantages are that the complexity of the compression process is increased, and it forces correction to be performed at the decoder, which further increases the complexity and cost of the decoder/display side. Performing colour correction as preprocessing also has the advantage that more complex correction algorithms can be applied, since it only has to be performed once before encoding rather than at every decoder.

A leading method for accounting for brightness variations in the compression process is the macroblock (MB) based illumination change compensation method proposed by Hur et al. in [5]. In their method, an illumination change (IC) value is calculated for each MB, which is the difference in the DC values between the MB being coded and the corresponding MB in the reference view being used for prediction. The IC for each MB is predicted from the IC values from neighboring MB's, and the difference between the actual value and predicted value is encoded in the bitstream. The motion estimation (ME) and motion compensation (MC) processes are altered to account for the IC values.

A simple method for correcting colour in multiview video as a preprocessing step is the histogram matching method proposed by Fecker et al. [6]. In their method, a lookup table is calculated for each of the Y, U and V channels based on the histograms of the view being corrected and the reference view. A fundamental disadvantage of histogram based methods is that they cannot deal with occlusions between views, or situations where the views have different amounts of foreground and background. Another preprocessing method is proposed in [7], where a scaling and an offset parameter are calculated for each YUV component. For example, the modified value for each Y sample in a view being corrected is calculated with:

$$Y^{cor} = aY^{org} + b \tag{1}$$

where Y^{cor} is the corrected value, Y^{org} is the original value and a and b are the scaling and offset parameters for the Y channel. The U and V channels are corrected equivalently. The scaling and offset values for each channel are calculated based on the histograms of the reference view and the view being corrected. Thus the method has the same weaknesses as histogram methods. Furthermore, it modifies each colour channel independently, so information from the other colour channels is not used.

In this paper we propose a colour correction preprocessing method for multiview video, which is based on finding matching points between views and performing a least squares regression on those points. In our method the corrected YUV values of a pixel are calculated as a weighted linear sum of the original YUV values for the pixel plus an offset. Matching points are found using block-based disparity estimation, and these points are used for calculating the optimal weights with a least squares regression.

The rest of this paper is organized as follows. The proposed method is described in Section 2, experimental results are presented in Section 3, and conclusions are given in Section 4.

2. PROPOSED COLOUR CORRECTION PREPROCESSING METHOD

We choose one view from a multi-view video as the reference and perform colour correction on the other views to make them match the colour in the reference as closely as possible. We express the corrected YUV values for each pixel in every non-reference view as a weighted linear sum of the original YUV values for the pixel.

$$\begin{aligned} Y^{cor} &= a_{Y1}Y + a_{Y2}U + a_{Y3}V + a_{Y4} \\ U^{cor} &= a_{U1}Y + a_{U2}U + a_{U3}V + a_{U4} \\ V^{cor} &= a_{V1}Y + a_{V2}U + a_{V3}V + a_{V4} \end{aligned} \tag{2}$$

For each view being corrected, we need to find sets of coefficients, $\mathbf{a}_Y$, $\mathbf{a}_U$, and $\mathbf{a}_V$, which make the corrected YUV values in the view match the YUV values in the reference view. To do this, we use disparity estimation to find matching points in the two views, and then perform a linear regression to find the coefficients.

A translational disparity model is used, where the colour corrected value of each pixel in any view can be expressed as a translated pixel from the reference view plus an error term:

$$f^{cor}(x,y) = f^{ref}\left(x+d_x, y+d_y\right) + \varepsilon(x,y) \tag{3}$$

Here, $f^{cor}(x,y)$ is one channel (Y, U or V) of the colour corrected view, $f^{ref}(x,y)$ is the same channel of the reference view, (d_x,d_y) is the disparity between the views and $\varepsilon(x,y)$ is an error term, accounting for camera noise, inaccuracy in the disparity vector, etc. Note that this model fails in occluded regions, which have to be considered when finding matching points.

We use block based disparity estimation on the luma channel to find matching points between the view being corrected and the reference view. Since there may be substantial variations in the luma levels between views, we use the Mean-Removed Sum of Absolute Differences (MRSAD) as the matching criterion. The MRSAD has been shown to be more accurate than the standard SAD for disparity estimation on multi-view video [5]. For an *NxN* block of pixels located at position (x_0,y_0) the MRSAD is defined as:

$$\begin{aligned} MRSAD(i,j) &= \sum_{x=x_0}^{x_0+N}\sum_{y=y_0}^{y_0+N}\left|\left(Y^{org}(x,y)-m^{org}\right)-\left(Y^{ref}(x-i,y-j)-m^{ref}\right)\right| \\ m^{org} &= \frac{1}{N^2}\sum_{x=x_0}^{x_0+N}\sum_{y=y_0}^{y_0+N}Y^{org}(x,y) \\ m^{ref} &= \frac{1}{N^2}\sum_{x=x_0}^{x_0+N}\sum_{y=y_0}^{y_0+N}Y^{ref}(x,y) \end{aligned} \tag{4}$$

Note that m^{org} and m^{ref} are the mean values of the pixels in each block, and *(i,j)* is the displacement being frames.

The view being corrected is divided into blocks of 8x8 pixels, and for each block a disparity vector, (d_x,d_y), is calculated by minimizing the MRSAD over a search range, $d_x \in [s_{x1}, s_{x2}]$, $d_y \in [s_{y1}, s_{y2}]$:

$$\left(d_x,d_y\right) = \underset{\substack{s_{x1}\le i\le s_{x2}\\ s_{y2}\le j\le s_{y2}}}{\arg\min}\; MRSAD(i,j) \tag{5}$$

The search range should be selected to fall around the epipolar line. The disparity vector calculated with equation (5) may not correspond to true disparity, because of occlusion between the views, noise, or other factors. Therefore, an additional test is used to decide whether the disparity estimation has found matching points for the current block. The MRSAD is compared to the mean-removed sum of absolute values (MRSAV) of the block:

$$MRSAV = \sum_{x=x_0}^{x_0+N} \sum_{y=y_0}^{y_0+N} \left| \left(Y^{org}(x,y) - m^{org} \right) \right| \tag{6}$$

The blocks are considered to be a match if $MRSAD(d_x, d_y) < 0.5 MRSAV$. The threshold 0.5 was determined experimentally. If the condition is true, the pixels from each block are added to vectors of matching points, $\mathbf{Y}^{\mathbf{org}}$, $\mathbf{U}^{\mathbf{org}}$, $\mathbf{V}^{\mathbf{org}}$, $\mathbf{Y}^{\mathbf{ref}}$, $\mathbf{U}^{\mathbf{ref}}$, $\mathbf{V}^{\mathbf{ref}}$.

Using the disparity model given by equation (3) the vectors of matching points can be expressed as:

$$\mathbf{Y}^{\mathbf{ref}} = \mathbf{Y}^{\mathbf{cor}} + \boldsymbol{\varepsilon}$$
$$\begin{bmatrix} Y_1^{ref} \\ Y_2^{ref} \\ Y_3^{ref} \\ \vdots \\ Y_N^{ref} \end{bmatrix} = \begin{bmatrix} Y_1^{cor} \\ Y_2^{cor} \\ Y_3^{cor} \\ \vdots \\ Y_N^{cor} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_N \end{bmatrix} \tag{7}$$

Substituting in equation (2), this can be expressed in terms of the original YUV values from the frame being corrected:

$$\begin{bmatrix} Y_1^{ref} \\ Y_2^{ref} \\ Y_3^{ref} \\ \vdots \\ Y_N^{ref} \end{bmatrix} = \begin{bmatrix} Y_1^{org} & U_1^{org} & V_1^{org} & 1 \\ Y_2^{org} & U_2^{org} & V_2^{org} & 1 \\ Y_3^{org} & U_3^{org} & V_3^{org} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ Y_N^{org} & U_N^{org} & V_N^{org} & 1 \end{bmatrix} \begin{bmatrix} a_{Y1} \\ a_{Y2} \\ a_{Y3} \\ a_{Y4} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_N \end{bmatrix} \tag{8}$$

Letting $\Psi = \left[\mathbf{Y}^{\mathbf{org}} \quad \mathbf{U}^{\mathbf{org}} \quad \mathbf{V}^{\mathbf{org}} \quad \mathbf{1} \right]$, (8) can be written in matrix notation as:

$$\mathbf{Y}^{\mathbf{ref}} = \Psi \mathbf{a}_{\mathbf{Y}} + \boldsymbol{\varepsilon} \tag{9}$$

The parameter vector $\mathbf{a}_{\mathbf{Y}}$ which minimizes the energy of the error vector $\boldsymbol{\varepsilon}$ can be found with a standard least squares estimator:

$$\mathbf{a}_{\mathbf{Y}} = \left(\Psi^{\mathrm{T}} \Psi \right)^{-1} \Psi^{\mathrm{T}} \mathbf{Y}^{\mathbf{ref}} \tag{10}$$

Similarly, the coefficients for generating the corrected U and V values can be found with:

$$\begin{aligned} \mathbf{a}_{\mathbf{U}} &= \left(\Psi^{\mathrm{T}} \Psi \right)^{-1} \Psi^{\mathrm{T}} \mathbf{U}^{\mathbf{ref}} \\ \mathbf{a}_{\mathbf{V}} &= \left(\Psi^{\mathrm{T}} \Psi \right)^{-1} \Psi^{\mathrm{T}} \mathbf{V}^{\mathbf{ref}} \end{aligned} \tag{11}$$

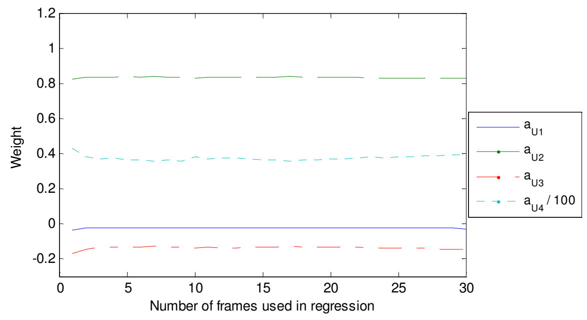

Figure 1: Least-squares weights for the U component of one view of the Flamenco2 multiview video

After the weight vectors have been calculated with equations (10) and (11), the view can be colour corrected to match the reference with equation (2).

Here, we find a single disparity vector for 8x8 pixel blocks and consider every pixel within the block to be a match to the reference frame. In stereo matching research, it is more common to estimate disparity on a pixel by pixel basis, with a block surrounding the current pixel being used in the matching process [8]. Calculating disparity and finding matches on a pixel by pixel basis may result in more accurate matches, and hence less outliers in the regression, but would greatly increase the number of computations needed. It would also make it more difficult to reuse the disparity vectors calculated during later compression of the multiview video.

The majority of video content is stored in YUV 4:2:0 format, where the chroma (UV) channels are downsampled by a factor of two in the vertical and horizontal directions relative to the luma. Equations (8) through (11) require a Y, U and V sample for every matching point between frames. Therefore, we downsample the Y channel by two in both directions for the purpose of finding the correction parameters. To apply equation (2) during the correction step, we upsample the U and V channels in order to calculate the corrected Y channel. That is when applying equation (1), the corrected Y channel is a function of the original Y channel and upsampled original U and V channels, and the corrected UV channels are functions of the original UV channels and the downsampled original Y channel.

For videos of reasonable length, it is not practical (or necessary) to use every frame during the regression for finding the weight parameters. Even if the video is only a few hundred frames long, millions of matching points could be found. Performing the least squares regression on vectors of this size would be computationally expensive and unnecessary since the parameters will converge with fewer points. This is illustrated in Figure 1, which shows the weights of the parameter vector $\mathbf{a}_{\mathbf{U}}$ (as defined in equation (2)) for one view of the Flamenco2 video obtained with different number of frames used during the disparity estimation and regression. We can see the parameters converge with only a few frames. Similar convergence results were obtained for other views and test videos, and the other parameter vectors.

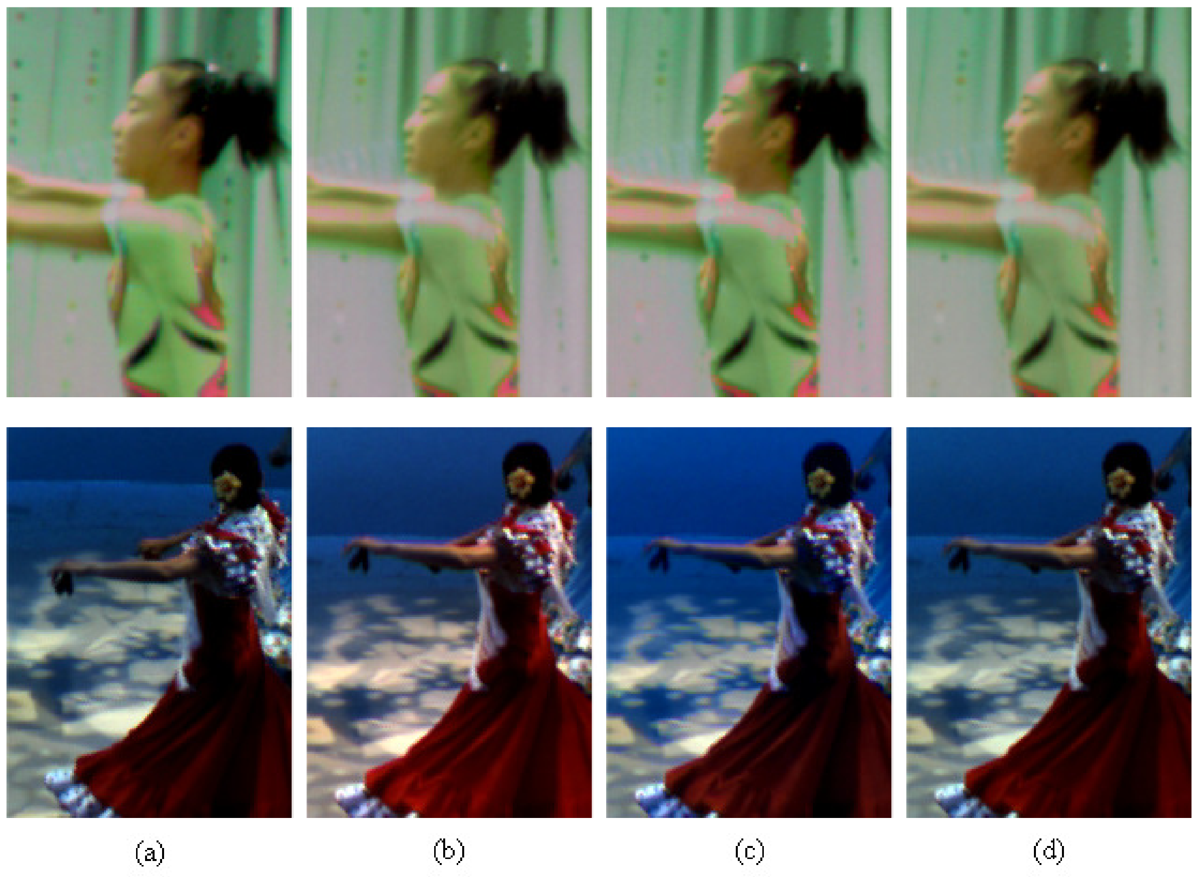

Figure 2: Subjective comparison of different colour matching algorithms on the Rena (top) and Flamenco2 (bottom) test sequences. (a) Reference view (b) Original version of view being corrected (c) Corrected with Histogram Matching (d) Corrected with proposed algorithm

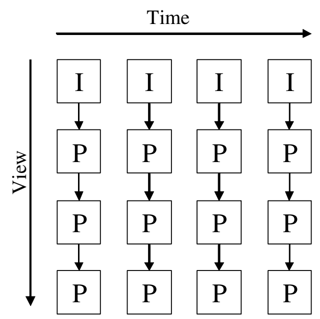

Figure 3: Prediction structure used in coding experiments

3. RESULTS

Results are presented for two standard multiview video test sequences Rena (16 views) and Flamenco2 (5 views). For each video, a view near the centre of the camera array was chosen as the reference, and the other views were corrected to match it. 60 temporal frames were processed for each video, with every tenth frame being used in the regression analysis for calculating the parameter vectors $\mathbf{a}_Y$, $\mathbf{a}_U$ and $\mathbf{a}_V$. One set of parameters was used to correct every frame in a view (i.e. different parameters were not applied at different times).

An example corrected frame from each video is shown in Figure 2. The proposed method is compared against the histogram matching method in [6]. It can be seen that the proposed method produces colours closer to those of the reference view, particularly in the forehead region in Rena and the floor in Flamenco2.

To show the effect the proposed method has on video compression performance when inter-view prediction is used, the test videos were compressed with the H.264 reference software (JM). Disparity compensated prediction was used with an IPPP coding structure in the view direction (Fig. 3). We compare our method with compressing the original (uncorrected) video and with the histogram matching (HM) method in [6].

Peak Signal to Noise Ratio (PSNR) vs. bit rate curves for both the luma and chroma channels of the test videos are shown in Figure 4. We observe that the gain in luma PSNR for our method over compressing the original uncorrected video is about 1.0 dB for Rena and 0.8 dB for Flamenco2. In the chroma channels, the proposed method increases PSNR by about 1 to1.2 dB for both Rena and Flamenco2 compared to compressing the original uncorrected video. The proposed method gives better performance than the HM method, particularly in the chroma channels. For the Rena video set, the proposed method in-

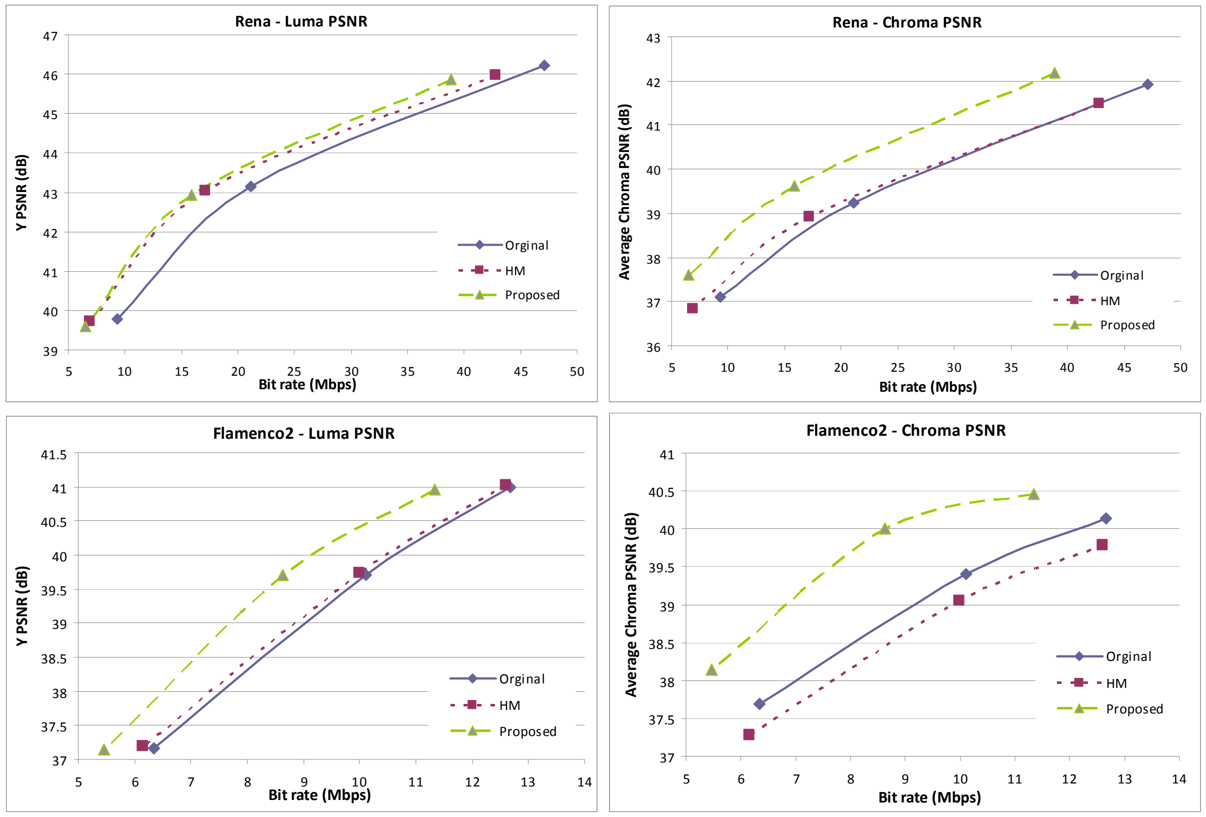

Figure 4: Rate-distortion curves for the luma (Y) and chroma (UV) channels of the Rena and Flamenco2 test sequences with different colour correction preprocessing

creases PSNR by about 0.1 dB in the luma and 1.0 dB in chroma compared to HM. For the Flamenco2 video, the gain in PSNR over HM is about 0.8 dB in the luma channel and 1.5 dB in the chroma channels. Note that HM actually decreases the quality in the chroma channels for the Flamenco2 video compared to compressing the original uncorrected video, whereas the proposed method always shows better performance than compressing the original video.

4. CONCLUSION

In this paper, a method is proposed for correcting colour in multiview video sequences as a preprocessing step to compression. One view is chosen as the reference, and all other views are corrected to match it. The corrected YUV values are expressed as a weighted linear sum of the original YUV values and an offset. Disparity Estimation is used to find matching points between the view being corrected and the reference, and a least squares regression is performed on the set of matching points to find the optimal weight values for correction. Experiment results show that the proposed method produces videos with colours that closely match the reference view, and that it increases compression efficiency by up to 1.0 dB when multiview video is compressed using H.264 with inter-view prediction. Future work will explore using more complicated non-linear functions to map sets of original YUV to corrected YUV values to determine if further gains can be achieved.

REFERENCES

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, C. Zhang, “Multiview Imaging and 3DTV,” IEEE Signal Processing Magazine, vol. 24, no. 6, pp. 10-21, Nov. 2007.

[2] S.C. Chan, H.Y. Shum; K.T. Ng, “Image-Based Rendering and Synthesis,” IEEE Signal Processing Magazine, vol. 24, no. 6, pp. 22-33, Nov. 2007.

[3] C.S. McCamy, H. Marcus, and Davidson, J.G., "A Color Rendition Chart," Journal of Applied Photographic Engineering, vol. 11, no. 3 pp. 95-99. 1976.

[4] A. Ilie and G. Welch, “Ensuring color consistency across multiple cameras,” in Proc. IEEE International Conference on Computer Vision (ICCV), Oct. 2005, pp. 1268–1275.

[5] J.H. Hur, S. Cho, and Y.L. Lee, “Adaptive Local Illumination Change Compensation Method for H.264-Based Multiview Video Coding,” IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 11, pp. 1496–1505, Nov. 2007.

[6] U. Fecker, M. Barkowsky, and A. Kaup, “Improving the prediction efficiency for multi-view video coding using histogram matching,” in Proc. Picture Coding Symposium 2006, pp. 2–16, Apr. 2006.

[7] Y. Chen, C. Cai, and J. Liu, “YUV Correction for Multi-View Video Compression,” Proc. Int. Conf. Pattern Recognition 2006, pp. 734-737, Aug. 2006.

[8] D. Scharstein and R. Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," International Journal of Computer Vision, vol. 47, pp. 7-42, 2002.