Color Correction Preprocessing for Multiview Video Coding

Colin Doutre, *Student Member, IEEE,* and Panos Nasiopoulos, *Member, IEEE*

***Abstract*— In multiview video, a number of cameras capture the same scene from different viewpoints. There can be significant variations in the color of views captured with different cameras, which negatively affects performance when the videos are compressed with inter-view prediction. In this letter, a method is proposed for correcting the color of multiview video sets as a preprocessing step to compression. Unlike previous work, where one of the captured views is used as the color reference, we correct all views to match the average color of the set of views. Block-based disparity estimation is used to find matching points between all views in the video set, and the average color is calculated for these matching points. A least-squares regression is performed for each view to find a function that will make the view most closely match the average color. Experimental results show that when multiview video is compressed with Joint Multiview Video Model, the proposed method increases compression efficiency by up to 1.0 dB in luma peak signal-to-noise ratio (PSNR) compared to compressing the original uncorrected video.**

***Index Terms*— Color correction, disparity estimation, multiview video coding (MVC), video processing.**

I. Introduction

MULTIVIEW VIDEO systems have recently received considerable attention from the research community [1]. To capture a 3-D representation of a dynamic scene, multiple video cameras are used, which capture the scene from different viewpoints. Using image-based rendering techniques [2], virtual viewpoints can be created, allowing the user to watch the scene from a range of viewing positions.

Since a number of cameras are needed, and each individual camera captures a large amount of data, the total amount of data captured in multiview video systems is huge. Hence, efficient compression methods are required for practical multiview video systems to allow efficient storage and transmission of the data.

A challenge that arises in multiview video coding (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 typically differences in brightness, color, 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 to improve compression performance. Variations between views also negatively affect rendering of new virtual viewpoints, and will be unpleasant to users as they switch between different views.

In MVC systems, brightness and color correction can be performed either as preprocessing before compression, or incorporated into the compression process itself. Accounting for color variations in the compression process itself has the advantage that the original data is restored during the compression process, which gives flexibility for different color 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 color correction as preprocessing also has the advantage that more complex correction algorithms can be applied, since it has to be performed only 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 [3]. 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 MBs, and the difference between the actual value and predicted value is encoded in the bit stream. The motion estimation (ME) and motion compensation processes are altered to account for the IC values. This method is only applied to the luma channel of the video.

In [4], another method for correcting brightness and color during the encoding process is proposed, which uses lookup tables in RGB color space. Each frame being used for prediction is converted to RGB color space, and the red, green, and blue color channels are modified independently with three separate lookup tables. The correction lookup tables are calculated by finding matching points between pairs of views and using a dynamic programming method to find a function that will make the colors of the matching points agree. Then the frame is converted back to the YUV color space to be used for inter-view prediction. The lookup tables are sent as side information in the bit stream.

Manuscript received June 4, 2008, revised September 30, 2008 and November 19, 2008. First version published May 12, 2009; current version published September 16, 2009. This work was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). This paper was recommended by Associate Editor Y.-S. Ho.

The authors are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: colind@ece.ubc.ca; panos@ece.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2009.2022780

Fig. 1. Finding matching points between all views by choosing an anchor view and performing block matching between the anchor and all other.

A simple method for correcting color in multiview video as a preprocessing step is the histogram matching method proposed by Fecker *et al.* [5]. 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. The histogram matching method was improved in [6], with the correction being performed in the RGB color space and a time-constant correction function being used (so the same correction lookup table is used for every temporal frame in a sequence). 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

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

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.

In previous work on color correction in multiview video sets, one view is always chosen as the color reference and all other views are modified to match it on a pairwise basis [3]–[7]. The reference view is usually chosen as a view in the center of the camera arrangement [5]–[7]. The reason for choosing the center view is that it will usually have the most in common with the other views. However, the center view may not always be a good choice for the color reference. The center view may have substantially different color from the other views. For example, in the standard *Flamenco2* multiview video set, the center camera has much lower brightness and less saturated colors than the other views (Fig. 1). If it is used as the reference, all other views will have to be greatly altered to make their colors match the reference.

In this letter, we propose a new method, where instead we find the average color of the video set and modify all views to match the average. The advantage of using the average color as the reference is that the minimum amount of modification will be needed across the set of views in order to make them consistent. A set of matching points between all views in the video set is found with block-based disparity estimation. The average Y, U, and V value for each set of matching points is calculated. A least-squares regression is preformed for each view to find the coefficients of a polynomial that will make the corrected YUV values of the view most closely match the average YUV values. Experimental results show that applying the proposed color correction greatly increases the compression efficiency when multiview video is compressed with inter-view prediction.

The rest of this letter is organized as follows. The proposed method is described in Section II. Experimental results are presented in Section III, including subjective color comparisons and tests on how the proposed method improves inter-view prediction in multiview video coding. Conclusions are given in Section IV.

II. Color Correction Preprocessing Method

Although modifying each view to match the average color of all views is a relatively intuitive concept, in practice defining and calculating the average color is not very straightforward. We propose a point-based method for finding the average color. We find matching points between all views using a correlation method, and then calculate the average Y, U, and V values of these matching points. We use a least-squares regression to find a polynomial function to map the initial YUV values in each view to match these average YUV values. These steps are described in detail in the following sections.

A. Point Matching Between Views

Point matching in stereo and multiview camera arrays is a classical problem that has been extensively studied. An overview that evaluates several methods is presented in [8]. There are two main approaches to point matching: block-based methods and feature-based methods. Block-based methods divide one image into small blocks and attempt to find matching blocks in another image based on some criteria, for example the sum of squared differences. Feature-based methods, such as the scale-invariant feature transform [9], involve extracting keypoints from each image and looking for matches between keypoints. Block-based methods produce far more matching points between images, but the matches are less reliable.

In order to find a large number of matching points, we use block-based disparity estimation on the luma channel to find matching points between all views. One view in the center of the camera arrangement is chosen as the anchor. This view is divided into blocks of size 8×8 pixels, and matching blocks for every block in the anchor are found in all other views (Fig. 1). In video coding, block matching is usually done using the sum of absolute differences or sum of square differences as the cost function. However, these cost functions are sensitive to the brightness level of the two blocks, and have been shown to have very poor performance when there are brightness variations between views [10]. In multiview

(a) (b)

Fig. 2. Color correction of the *Flamenco2* multiview video set: (a) original data and (b) color-corrected with the proposed method.

video sets, there may be substantial brightness variations, so a more robust matching criterion is needed. Therefore, we use the normalized cross correlation (NCC) defined for an $N \times N$ block located at position (x_0, y_0) in the anchor frame as shown in (2) at the bottom of the page, where $Y_{anc}(x, y)$ is a frame of the anchor view, $Y_{\text{view}}(x, y)$ is the view in which a matching block is being found, and m_{view} and m_{anc} are the mean values of the blocks in the two frames

$$m_{\text{view}} = \frac{1}{N^2} \sum_{x=x_0}^{x_0+N-1} \sum_{y=y_0}^{y_0+N-1} Y_{\text{view}}(x, y)$$
$$m_{anc} = \frac{1}{N^2} \sum_{x=x_0}^{x_0+N-1} \sum_{y=y_0}^{y_0+N-1} Y_{anc}(x, y). \tag{3}$$

Note that the mean of the block is subtracted and the energy of the block is normalized in the NCC calculation. This provides robustness to changes in brightness between the views. The disparity is estimated by choosing the vector (i^*, j^*) that results in the maximum NCC over a search range $i \in [s_{x1}, s_{x2}]$, $j \in [s_{y1}, s_{y2}]$

$$(i^*, j^*) = \underset{\substack{s_{x1} \le i \le s_{x2} \\ s_{y2} \le j \le s_{y2}}}{\arg\max}\ NCC(i, j). \tag{4}$$

In this letter, we have used a rectangular search range and a full search (where every displacement vector within the search range is evaluated). The search range was chosen separately for each video set based on the range of disparities observed in the sequence. Note that the computational cost of disparity estimation could be reduced by using a fast disparity estimation algorithm such as those proposed in [11] and [12].

The disparity vector calculated with (4) 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 NCC ranges from $[-1, 1]$ and has a value of 1 when the blocks are scaled versions of each other (after mean removal). If the NCC value between the block in the anchor view and the matching blocks in other views is above 0.7 for all views, the blocks are considered to be valid matches across all views, and the pixels in the block are added to vectors of matching points $\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_M}, \mathbf{U_1}, \mathbf{U_2}, \ldots, \mathbf{U_M}, \mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_M}$. The subscript indicates the view number, with M being the number of views in the video set. The threshold 0.7 was determined experimentally to provide good results for typical multiview content.

With the vectors of matching points, the average YUV values across all views can easily be calculated for these points

$$\mathbf{Y^{avg}} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{Y_i} \quad \mathbf{U^{avg}} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{U_i} \quad \mathbf{V^{avg}} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{V_i}. \tag{5}$$

B. Color Correction Function

A transformation has to be found for each view to map the view's captured YUV values to match the average YUV values as closely as possible. We use three functions to calculate the corrected Y, U, and V values for each frame, which are designed to minimize the error between the corrected YUV values for the view and the average YUV values

$$\mathbf{Y^{avg}} = \mathbf{Y_i^{cor}} + \boldsymbol{\varepsilon}. \tag{6}$$

In color correction research, polynomials are often used to find least-squares fits to nonlinear functions. For a third-order polynomial, the corrected Y, U, or V value for each pixel is

$$NCC(i, j) = \frac{\sum_{x=x_0}^{x_0+N-1} \sum_{y=y_0}^{y_0+N-1} (Y_{anc}(x, y) - m_{anc})(Y_{\text{view}}(x - i, y - j) - m_{\text{view}})}{\sqrt{\sum_{x=x_0}^{x_0+N-1} \sum_{y=y_0}^{y_0+N-1} (Y_{anc}(x, y) - m_{anc})^2 \cdot \sum_{x=x_0}^{x_0+N-1} \sum_{y=y_0}^{y_0+N-1} (Y_{\text{view}}(x - i, y - j) - m_{\text{view}})^2}} \tag{2}$$

calculated based on the captured YUV values of the pixel as

$$\begin{aligned} Y^{\text{cor}} &= a_{Y1}Y + a_{Y2}U + a_{Y3}V + a_{Y4}Y^2 + a_{Y5}U^2 \\ &\quad + a_{Y6}V^2 + a_{Y7}YU + a_{Y8}YV + a_{Y9}UV \\ &\quad + a_{Y10}Y^3 + a_{Y11}U^3 + a_{Y12}V^3 + a_{Y13}YUV \\ &\quad + a_{Y14}Y^2U + a_{Y15}Y^2V + a_{Y16}YU^2 + a_{Y17}YV^2 \\ &\quad + a_{Y18}U^2V + a_{Y19}UV^2 + a_{Y20}. \end{aligned} \tag{7}$$

The correction function for each view is controlled by the weighting vector $\mathbf{a_Y} = [a_{Y1},\ a_{Y2}, \ldots, a_{Y20}]$. The corrected U and V values are calculated equivalently with different weight vectors $\mathbf{a_U}$ and $\mathbf{a_V}$. A third-order polynomial is shown in (7) but any order is possible. Through experimentation, we have found that increasing the order above three results in minimal gain and considerable increase in complexity.

The optimal weight vectors that will make the corrected values match the average values can be calculated with a least squares regression. Define Ψ as follows, with multiplication of vectors being performed elementwise:

$$\begin{aligned} \Psi = \Big[&\mathbf{Y_i}\ \mathbf{U_i}\ \mathbf{V_i}\ \mathbf{Y_i^2}\ \mathbf{U_i^2}\ \mathbf{V_i^2}\ \mathbf{Y_iU_i}\ \mathbf{Y_iV_i}\ \mathbf{U_iV_i} \\ &\mathbf{Y_i^3}\ \mathbf{U_i^3}\ \mathbf{V_i^3}\ \mathbf{Y_iU_iV_i}\ \mathbf{Y_i^2U_i}\ \mathbf{Y_i^2V_i} \\ &\mathbf{Y_iU_i^2}\ \mathbf{Y_iV_i^2}\ \mathbf{U_i^2V_i}\ \mathbf{U_iV_i^2}\ \mathbf{1}\Big]. \end{aligned} \tag{8}$$

Here, $\mathbf{Y_i}$, $\mathbf{U_i}$, and $\mathbf{V_i}$, are the vectors of pixels of view 'i' found in the block matching process. The color-corrected Y values can be calculated as

$$\mathbf{Y}^{\text{cor}} = \Psi \mathbf{a_Y}. \tag{9}$$

Substituting this into (6) gives

$$\mathbf{Y}^{\mathbf{avg}} = \Psi \mathbf{a_Y} + \boldsymbol{\varepsilon}. \tag{10}$$

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

$$\mathbf{a_Y} = (\Psi^{\mathrm{T}}\Psi)^{-1}\Psi^{\mathrm{T}}\mathbf{Y}^{\mathbf{avg}}. \tag{11}$$

Equivalent regressions are performed to calculate the correction vectors for the U and V channels. After the weight vectors have been calculated with (11), each pixel in the view can be color-corrected with (7).

The majority of video content is stored in YUV 4:2:0 formats, where the chroma (UV) channels are downsampled by a factor of two in the vertical and horizontal directions relative to the luma. Equations (7) 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. This yields a quarter-size video in YUV 4:4:4 format. To apply (7) during the correction step, we upsample the U and V channels in order to calculate the corrected Y channel. That is, when applying (7), the corrected Y channel is a function of the original Y channel and upsampled U and V channels, and the corrected U and V channels are functions of the original U and V channels and the downsampled original Y channel.

In practice, the block-matching process is not perfect, so there will be a number of bad matches that will cause outliers in the regression. But as long as the ratio of good matches to bad matches is relatively high, the bad matches will not have a significant effect on the estimated correction parameters. If there were a high number of outliers, robust regression methods could be used [13]; however, we have found they are not necessary for the multiview videos used here.

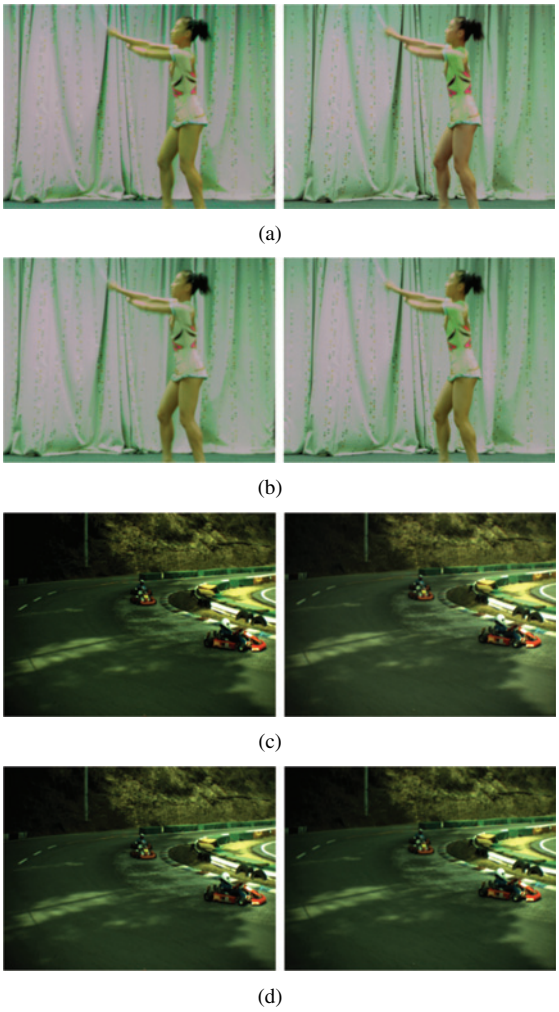

Fig. 3. Sample views from the *Rena* and *Race* test video sets: (a), (c) frames from two views before correction and (b), (d) after proposed color correction.

III. Results

Results are presented for four standard multiview video test sequences: *Rena* (16 views), *Flamenco2* (5 views), *Race* (8 views), and *Ballroom* (8 views). All videos have resolution 640 × 480. Sixty 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 $\mathbf{a_V}$. The same set of parameters was used to correct every temporal frame in a view.

To show subjectively how the proposed method alters multiview video, the first frame from each view of the *Flamenco2* video set before and after color correction is shown in Fig. 2. Before correction, there are visible differences in brightness and color between the different views, particularly the center view which is darker than the rest and has less saturated colors. After correction, the views look very consistent in

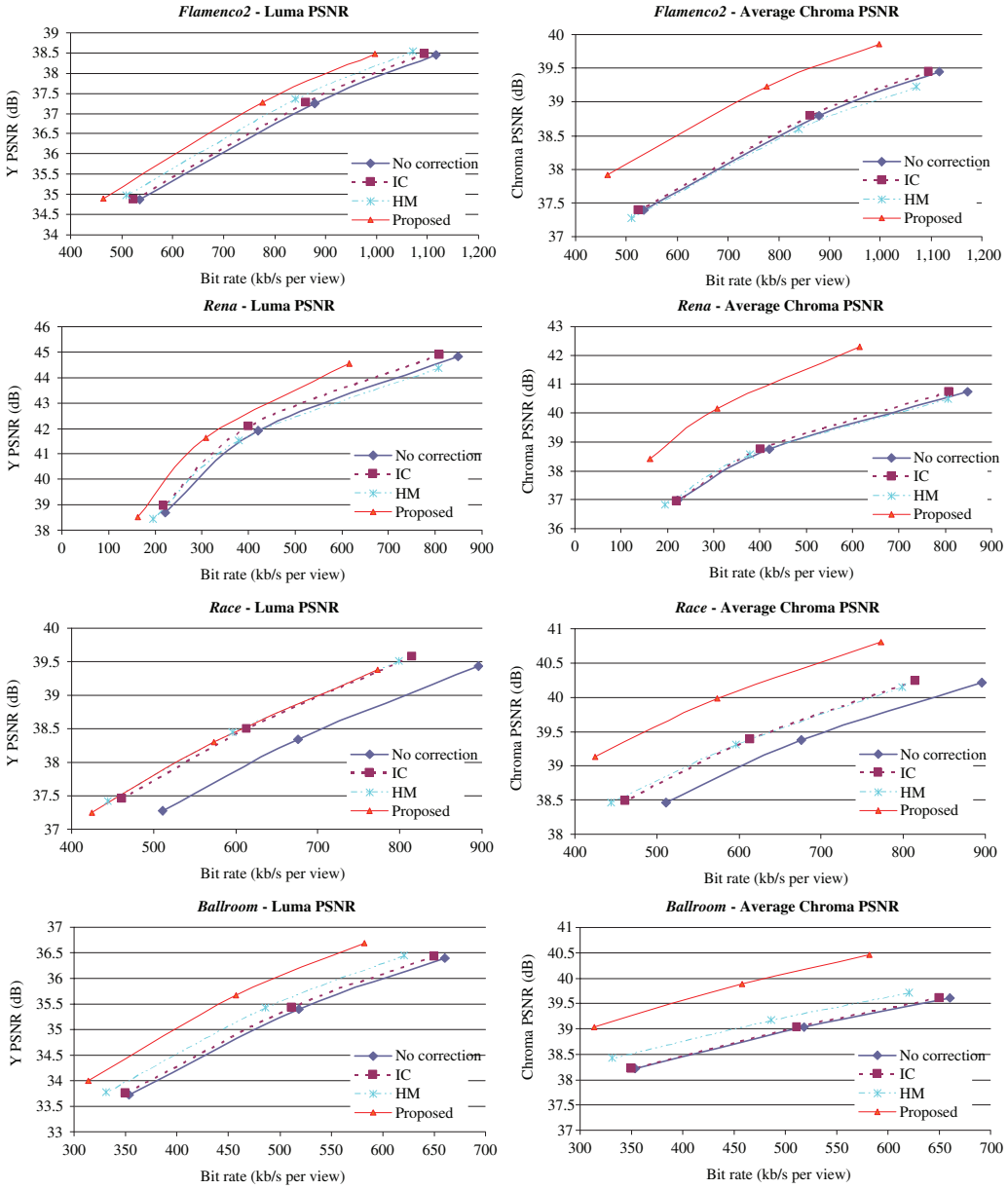

Fig. 4. Rate-distortion performance obtained with the proposed color correction preprocessing.

brightness and color, and all the views have color which looks like the average color before correction. Since the other video sets have many views (8 or 16), it is not practical to show all of the views at once due to limited space. For this reason, we select to show only two views from the *Rena* and *Race* video sets for subjective comparison (Fig. 3). These views have noticeably different color before correction but look consistent after our proposed color-correction algorithm is applied.

A. *Effect on Compression Performance*

When measuring PSNR, the compressed video is compared to a reference which is assumed to have perfect quality. Usually the original video is used as the reference. However, when color correction is being performed, a fundamental assumption is that the original data is not perfect and needs to be modified. Therefore, when measuring the PSNR of video where both color correction and compression have been applied, we use the color-corrected version of the video as the reference. The PSNR reported is a measure of the amount of distortion introduced in the compression process. Note that when different color correction schemes are compared, we use different references for measuring the PSNR, because there is no ground truth video that has perfect color.

In most video coding papers, only the luma PSNR is reported since the human visual system is more sensitive to luma than chroma. However, here we are modifying the color of the video, so the chroma quality is also relevant. Hence we provide two rate-distortion curves for each video to show the luma and chroma PSNR. The chroma PSNR is the average PSNR of the U and V channels.

In order to evaluate how the proposed method effects multiview video encoding performance, experiments were run with the joint multiview video model reference software (JMVM 8.0) [14]. A temporal GOP size of 8 was used, and the QPs used are those used in the JVT common test conditions [15].

We compare our method with 1) compressing the original video; 2) the illumination compensation (IC) method adopted in JMVM; and 3) the histogram-matching (HM) method presented in [6]. Fig. 4 shows the PSNR versus bit rate curves for both the luma and chroma channels of the test videos. The proposed method results in luma PSNR gains ranging from about 0.5 to 1 dB over compressing the original data without IC. The gains compared to using IC range from 0 to 0.6 dB. Greater PSNR gains are seen in the chroma channels. The proposed method gives chroma PSNR gains ranging from about 0.7 to 2.1 dB over compressing the original videos. For all videos the proposed method gives similar or better performance than the HM method.

B. Computational Complexity

The proposed method consists of three main steps: 1) finding matching points through disparity estimation; 2) performing the least squares regressions; and 3) calculating the corrected YUV values. The complexity of the first two steps is dependent on the number of frames used to calculate the correction parameters, which can be a small subset of the total temporal frames in the video (six frames were used in our tests). The last step must be performed on every frame in the video set, so its computational complexity depends on the temporal length of the video.

Disparity estimation is used to find matching points between all views. The computational complexity of this process is heavily dependent on the search range and search method used. The search range required varies from video to video, as it depends on the scene and camera geometry. In our experiments, we have used a rectangular search window, and a full search (where the cost function (2) is evaluated at every possible disparity within the window). The speed of the search process could be considerably increased by using fast disparity estimation methods such as those in [11], [12]. If the videos are well rectified, disparity estimation can be reduced to a 1-D search [16], greatly reducing the number of search points. Note that the disparity vectors calculated in this step could be reused in later compression of the video (or serve as a starting point for a smaller refinement search), so the additional complexity of this step to a complete system may be minor.

Three least-squares regressions must be performed per view to calculate the correction vectors with (11). There are many very efficient numerical methods for performing least-squares regressions [17]. In our experiments we have used the MATLAB "\" operator which calculates the least-squares solution based on QR factorization [18]. On an Intel Core2 Duo E4400 2.0-GHz system, performing the least-squares regressions in MATLAB took 1.5 to 3.1 s per view, depending on how many matching points were found in disparity estimation.

To calculate the final corrected YUV values with (7), 45 multiplications and 19 additions are needed per sample (Y, U, or V). In our implementation, written in C code and compiled with Microsoft Visual Studio, this takes about 15 ms per 640×480 pixel frame. The complexity could be reduced by using a lower order polynomial, or removing some terms from (7), at the expense of slightly less accurate correction.

IV. Conclusion

In this letter, a method has been proposed for correcting color in multiview video sets to the average color of the set of original views. The color correction is done as a preprocessing step to compression. Block-based disparity estimation is used to find matching points between all views. A least-squares regression is performed on the set of matching points to find the optimal parameters for a polynomial that will make the captured values from each view match the average color values. Experimental results show that the proposed method produces video sets that are highly consistent in color. Applying the proposed method increases compression efficiency by up to 1.0 dB in luma PSNR when multiview video is compressed with JMVM.

References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview Imaging and 3-D TV," *IEEE Signal Process. Mag.*, vol. 24, no. 6, pp. 10–21, Nov. 2007.

[2] S. C. Chan, H. Y. Shum, and K. T. Ng, "Image-based rendering and synthesis," *IEEE Signal Process. Mag.*, vol. 24, no. 6, pp. 22–33, Nov. 2007.

[3] 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.

[4] K. Yamamoto, M. Kitahara, H. Kimata, T. Yendo, T. Fujii, M. Tanimoto, S. Shimizu, K. Kamikura, and Y. Yashima, "Multiview video coding using view interpolation and color correction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1436–1449, Nov. 2007.

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

[6] U. Fecker, M. Barkowsky, and A. Kaup, "Histogram-based pre-filtering for luminance and chrominance compensation of multi-view video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 9, pp. 1258–1267, Sep. 2008.

[7] Y. Chen, C. Cai, and J. Liu, "YUV correction for multi-view video compression," in *Proc. Int. Conf. Pattern Recognition*, Aug. 2006, pp. 734–737.

[8] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, no. 1–3, pp. 7–42, Apr.–Jun. 2002.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[10] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, Minneapolis, MN, Jun. 2007, pp. 1–8.

[11] J. Lu, H. Cai, J. G. Lou, and J. Li, "An epipolar geometry-based fast disparity estimation algorithm for multiview image and video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 6, pp. 737–750, Jun. 2007.

[12] Y. Kim, J. Kim, and K. Sohn, "Fast disparity and motion estimation for multi-view video coding," *IEEE Trans. Consumer Electron.*, vol. 53, no. 2, pp. 712–719, May 2007.

[13] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. 1st ed. New York: Wiley, 1987.

[14] A. Vetro, P. Pandit, H. Kimata, A. Smolic, and Y. K. Wang, *Joint Multiview Video Model (JMVM) 8.0*, ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/SG16, Doc. JVT-AA207, Apr. 2008.

[15] Y. Su, A. Vetro, and A. Smolic, *Common Test Conditions for Multiview Video Coding*, ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/SG16, Doc. JVT-T207, Jul. 2006.

[16] D. V. Papadimitriou and T. J. Dennis, "Epipolar line estimation and rectification for stereo image pairs," *IEEE Trans. Image Process.*, vol. 5, no. 4, pp. 672–676, Apr. 1996.

[17] A. Björck, *Numerical Methods for Least Squares Problems*. 1st ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1996.

[18] C. Moler, *Numerical Computing with MATLAB*. 1st ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2004, pp. 146–150.