I-077

Color Correction of Multiview Camera System Using Matched Feature Points

Mehrdad Panahpour Tehrani† Akio Ishikawa† Shigeyuki Sakazawa† Atsushi Koike†

1. Introduction

The quality of generated viewpoints in Free viewpoint TV (FTV) [1] can be improved if the multiview images are color corrected. Color correction can be done either by adjusting cameras' parameter or image processing. Yamamoto *et al*. [2] proposed a color correction for multiview images using Scale Invariant Feature Transform (SIFT) [3], which finds color correspondences. In [2], at first a reference camera is manually chosen and other cameras called target cameras, which will be corrected to have similar color with the reference camera. This method cannot perform well if the target and reference cameras are far, because it cannot find correct correspondences. This method picks correspondences only from one frame; therefore it cannot handle occlusion parts well.

In this paper, we propose an advanced approach based on [2] for color correction of multiview images that overcomes the abovementioned problems.

2. Color Correction Algorithm

At first, we choose one reference camera from multiple cameras. The nearest camera called target camera and will be corrected to have similar color with the reference camera. Then, the color corrected camera (i.e. target camera in previous step) is chosen as reference camera, and the next viewpoint is called target camera, which will be corrected. This procedure is done until we reach the first reference camera. It is the first iteration. Fig. 1 shows how we move on camera array during iterations. This method simply can be extended for any camera configuration based on finding the nearest camera.

For each RGB channel, total intensities change (i.e. difference between reference intensity (b_n) and correct intensity (c_n) for target image, obtained by the algorithm) is calculated for all pairs in all iterations. During iterations, if the ratio of changes (i.e. current change divided by previous change) in average for all pairs is less than a threshold, the result of that iteration is determined n as final result.

For each pair of target and reference cameras, we detect "*K*" geometrically matched feature point locations using modified SIFT and save them in set "$\mathbf{P_1}$". In other word, we eliminate errors caused by wrong correspondences (i.e. non-geometrically matched). Then set "$\mathbf{Q_1}$" for matched colors' intensities is generated using $\mathbf{P_1}$ for each RGB channels, independently. To compensate the color changes in edge areas, matched colors of Gaussian filtered ("*G*-1" times) of original pair in the same locations (taken from $\mathbf{P_1}$) are also collected and added to $\mathbf{Q_1}$. The same procedure (generating $\mathbf{P_i}$, Gaussian filtering, and generating $\mathbf{Q_i}$) is applied to several frames (or time instances – "*F*" frames, with "*S*" frames' interval) of the camera pair, in order to compensate the occlusion in the first frame. Then, set "$\mathbf{Q} = \sum \mathbf{Q_i}$" is generated. Finally, we calculate a lookup table (LUT) for each RGB channel that indicates a non-linear transformation. The target camera (in the first frame) is corrected by using this table.

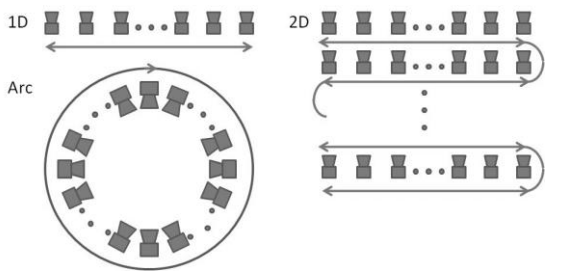

Fig.1: Iteration procedure for different camera arrays

LUT or "f_c" is generated by energy "$E(C)$" minimization of matched colors between target and reference images, using dynamic programming. Each color correspondences in $\mathbf{Q}$ is represented as a 2D Gaussian distribution N(σ_f , μ=(a_i' , a_i)). "σ_f" is constant and (a_i' , a_i) are correspondence intensities in $\mathbf{Q}$. Following equations shows how "f_c" or LUT is obtained.

$$c_n = f(b_n) \qquad B = \{0,1,2,...,n(=b_n),...,255\} \qquad C = \{c_0, c_1, ..., c_n, ..., c_{255}\}$$

$$f_c = \underset{C}{argmin}\, E(C)$$

$$E(C) = E_1(C) + \lambda E_2(C)$$

$$E_1(C) = \sum_{n=1}^{255} J(C,n)$$

$$J(\mathrm{C,n}) = \sum_{i=1}^{KGF} \frac{-\alpha_i}{2\pi\sigma_f^2 KGF} \exp\left(-\frac{(b_n - a_i)^2 + (c_n - a_i')^2}{2\sigma_f^2} \right), \quad \alpha_j = \frac{256 - |a_i - a_i'|}{256}$$

$$E_2(C) = \sum_{n=1}^{255} |c_n - c_{n-1} - 1|$$

where "b_n" is reference value in set "*B*" and "c_n" is correct value for target image in set "*C*". "*KGF*" is length of $\mathbf{Q}$. "λ" is non-constant and depends of image texture (i.e. obtained experimentally). "*J*(C,n)" is distribution of $\mathbf{Q}$ for all color correspondences, which is linearly weighed. The weighting "α_i" also helps to reduce the effect of wrong color correspondences. "$E_1(C)$" means the target image should be corrected by using $\mathbf{Q}$. "$E_2(C)$" forces LUT to increase step-by-step. Note that "$E_1(C)$" and "$E_2(C)$" are calculated through dynamic programming.

In this approach, RGB channels are corrected independently and non-linearly because of channel-independent lookup tables. Instead of using a color pattern board, this approach detects geometrically matched points and corresponding colors. Therefore it is not necessary to capture color pattern board, possible to handle occlusions, and suitable for multiview images.

There are following differences between our method and [2]:

1. Color correction is always applied to the nearest camera.
2. Using several time instances.
3. Weighting for distribution function of correspondences, *J*(C,n).
4. Suppressing non-geometrically matched feature points.
5. Iteration based until color correction with optimal result.

3. Experiment

To evaluate the proposed method, we applied our method to several multiview sequences. Here, we show only the experimental result for "*crowd*", and "*object*". The constant parameters for all sequences are obtained experimentally and shown in Table 1.

Note that λ is also constant (λ=0.0001) according to [2]. However, λ must be adaptive to each image texture.

† Visual Communication Laboratory, KDDI R&D LABS Inc.

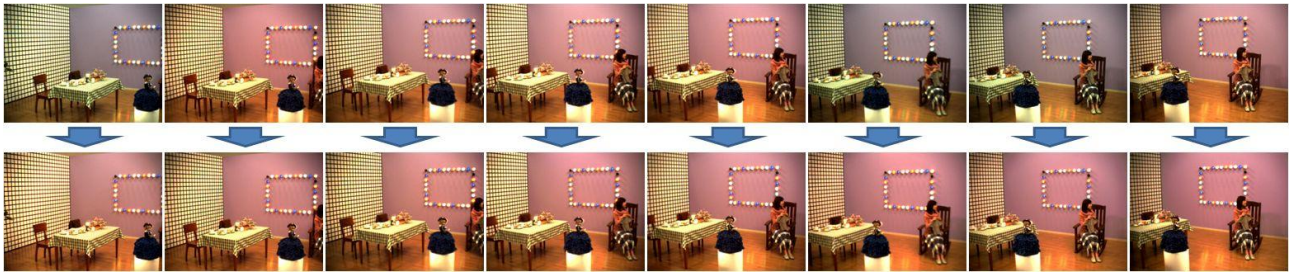

Fig.2: Result of color correction using condition (b) for "*object*" (Iteration started from the most left camera)

Fig.3: Result of color correction using condition (b) for "*crowd*" (Iteration started from the most left camera)

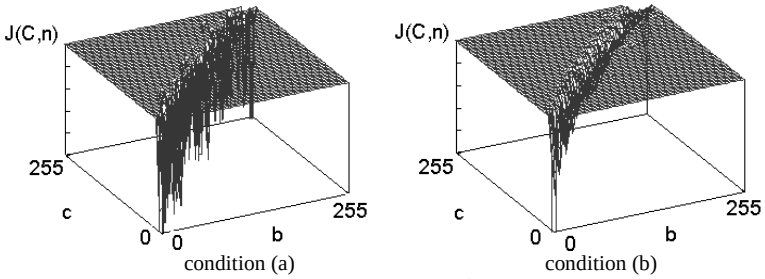

Fig.4: J(C,n) (R channel, "*crowd*") for two conditions in Table 2

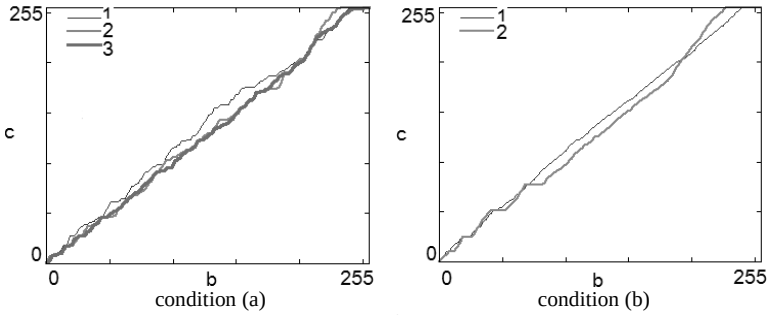

Fig.5: LUT (R channel, "*crowd*") for two conditions in Table 2 in last iteration (Numbers in graphs indicate total number of iterations)

Experimentally, we found out that for images with high texture λ=0.0001 is suitable and algorithm is stable, however when the image has many flat areas or has low texture λ=0.001 performs better. We adjusted λ experimentally for "*crowd*", and "*object*" equal to 0.0001, 0.0003, respectively.

Experiments were done for two conditions as shown in Table 2. To evaluate our method (i.e. condition (b)), we perform iteration based color correction of [2] (i.e. condition (a)). Table 2 shows each condition and the number of iterations that the color correction algorithm stopped based on the constraint explained in section 2. Regarding the speed of color correction algorithm, Table 2 shows the efficiency of the proposed method, since the number of iteration is reduced to two for all sequences, when we apply several time instances, weighting the distribution function and suppressing the non-geometrically matched feature points. Note that the number of iterations depends on the threshold. In this experiment, we set the threshold equal to 0.2. However, iterations in the case of "*object*" stop with average change ratio about 0.1.

Fig. 2, and Fig. 3 show the result of color correction for "*crowd*" (3 cameras), "*objects*" (8 cameras) using condition (b), respectively. As it can be seen, all viewpoints are color-corrected uniformly, with the dominant color pattern in multiview images; even color correction algorithm was started from the viewpoint without dominant color pattern (the most left camera in Fig 2 and Fig.3). In other word, we can start color correction from any viewpoint and still obtain the dominant color pattern, eventually. It guarantees the optimal performance of our proposed method. However, in [2] the first viewpoint decides the dominant color.

Table 1: Parameters

K=300	F=30	S=20		σ_f = 0.5
G=4	σ_0 = 0	σ_1 = 0.5	σ_2 = 1.0	σ_3 = 2.5

Table 2: Conditions and Iterations for "*crowd*", "*object*", and "*race*"

	Iteration		
Condition	*Crowd*	*Object*	*Race*
(a) F=1	3	2	3
(b) F=30 + weighting + suppress	2	2	2

Fig. 4 shows distribution function J(C,n) (i.e. distribution of correspondences' intensities). Obviously it became narrower and smoother for condition (b). This reduces the noise caused by wrong correspondences, and makes the result of dynamic programming more robust. Hence, the result of LUT for two conditions in Table 2 applied to "*crowd*" became smoother, and algorithm can converge and reach to optimal result in two iterations, as shown in Fig. 5.

4. Conclusion

This paper proposed a novel method for color correction of multiview camera system. This method can eases geometry compensation of multiview images, matting (i.e. separating background and foreground) and also makes the audiences feel comfortable when cameras are switched or when free viewpoint is generated in a 3D display system.

In the future, we will focus on adaptively choosing constant parameters for the proposed algorithm based on multiview camera images' characteristic, and cameras' configuration.

References

[1] P. Na Bangchang, M. Panahpour Tehrani, T. Fujii, M. Tanimoto, "Realtime System of Free Viewpoint Television", Journal of ITE, Vol. 59, No. 8, pp. 63-70, (2005)

[2] K. Yamamoto, T. Yendo, T. Fujii, M. Tanimoto and D. Suter: "Color Correction for Multi-camera System by using Correspondences," SIGGRAPH2006 Poster (2006).

[3] D. G. Lowe "Distinctive Image Features from Scale-Invariant Keypoints," Intl. J. of Computer Vision, vol.60, no.2, pp.91-110 (2004).