Multiple Color Texture Map Fusion for 3D Models

Nobuyuki Bannai*, Robert B. Fisher, Alexander Agathos
School of Informatics, University of Edinburgh

Abstract

A commonly encountered problem when creating 3D models of large real scenes is unnatural color texture fusion. Due to variations in lighting and camera settings (both manual and automatic), captured color texture maps of the same 3D structures can have very different appearances. When fusing multiple texture maps to create larger models, this color variation leads to poor appearance with patchwork color tilings on homogeneous surfaces. This paper extends previous research on pairwise global color correction to multiple overlapping texture map images. The central idea is to estimate a set of blending transformations that minimize the overall global color discrepancy between the texture maps, thus spreading residual color errors, rather than letting them accumulate.

Keywords: virtual models, color consistency, texture mapping, color matching

1 Introduction

When a realistic 3D virtual model is constructed from real scenes, multiple color texture maps are often necessary. Taking images from different view angles and with different camera settings creates mismatching colors due to lighting and camera conditions. As some researchers [1, 3, 4] have shown, texturing from two different images causes color discrepancies to appear on the rendered surfaces. The discrepancies can include color as well as lightness shifts, and results in a mosaic appearance on the surfaces. To remove the color discrepancies, Agathos and Fisher [1] introduce a pairwise correction method by estimating the RGB color transformation from corresponding pixels where two texture maps overlap. The corrections are then applied to the whole texture map. That research only considered pairwise texture map corrections. Beachesne and Roy [4] introduce a relighting method by computing the lighting ratio of two textures. Their method could be modified to adjust n textures. The RGB color transformation from pixels where two textures overlap is also estimated. Baumberg [3] introduced a multi-band blending technique that blends low and high band images by generating a weighting function, but requires more images than the others, even if a target is small. If high quality color constancy processes [2] were available then we could use them. Some recent approaches to color constancy [5, 11] exploit local surface color continuity properties, which is somewhat similar to the pointwise consistency across patches used here. But most color constancy approaches require sufficient variation in the colors in the image to estimate basis functions and the color distributions must not be biased. Unfortunately, bias is the case in our examples: some of the supposedly grey surfaces in Figure 10 have slight greenish or pinkish tints resulting from the automatic camera processing (this is corrected by our algorithm). Thus, we conclude that, while promising, existing color constancy processes (1) still need further development and (2) do not usually exploit the additional information that we have here, namely multiple color samples at corresponding points.

When constructing a large 3D virtual model, especially for a building or a complex shape object, more than two different texture images are needed (*e.g.* 10 - 20 images for all sides of a building, each image with its own distortions) and multiple color image fusion is unavoidable.

*Supported by Canon Inc.

Figure 1 shows a VRML model texture mapped using 30 images. The highlighted region consists of three different images seen from right, bottom and left directions. The dark oval indicates a region where 3 color texture maps overlap. Each image has different lighting conditions and the texture mapped model has color discrepancies at the image boundaries.

The central problem with direct application of previous pairwise correction methods can be illustrated as follows. Suppose we have three texture maps A, B and C containing, amongst other texture, three views of the same surface patch. Then, we can estimate the pixel color correction matrix $T_{B\to A}$ that adjusts the pixels in B to those in A. Similarly, $T_{C\to A}$ corrects colors from C to A. This means that we can have, in theory, some patches of A, B and C all in the color space of A. Unfortunately, this does not guarantee that no color discontinuities exist between other patches mapped in B and C, *i.e.* $T_{B\to A}T_{C\to A}^{-1} \neq T_{B\to C}$. Thus applying the correction to other patches in a texture map may produce texture maps with visibly different colors on patches that ought to be the same.

When more than three patches overlap, then the problem becomes more complicated. Our solution is to choose a patch X with a good appearance, and then find transformations $T_{Y\to X}$ ($Y \neq X$) that minimize the total color error over all overlapping patches. In particular, the error metric that is minimized is the cumulative RGB differences between the pixels of all overlapping patches seen in multiple texture maps (plus some special cases for the mean values of non-overlapping patches).

This paper introduces a new approach to remove color discrepancies between multiple overlapping texture maps. By overlapping, we mean that the 3D objects that we are modelling have surfaces that are observed in more than one texture map. As each texture map image is acquired independently, camera and lighting effects cause the different views of the same (and nearby) surfaces to have different appearances. This paper exploits the fact that the views are linked by the same 3D scene surface to estimate color transformations that lead to consistent colors. To remove the color discrepancies, the pairwise global correction method [1] is extended for multiple overlapping images.

Since real data sets have perspective distortions, a common image rectification method (Homography) [13, 9] is used first to match pixels in the overlap regions correctly (Section 2.1).

Some images have substantial luminosity differences between their regions due to different light source positions and shadows. To remove the luminosity variations, we firstly use a Color Space Transformation (RGB to HSV and HSV to RGB) [6, 12], where the luminosity is adjusted while in the HSV space (Section 2.1). The colors are then adjusted pairwise between related patches to remove the major discrepancies (Section 2.2). Finally, we apply the global consistency method introduced in this paper to overlapping (Section 2.3) and non-overlapping (Section 2.4) patches. A detailed summary of the process is given in Sec. 2.5

The algorithm presented here is based on several assumptions:

1. The images will be previously sectioned into triangulated regions used for texture mapping, and corresponding planar scene surface patches will have the same large triangles with vertices at the same scene points. The color corrections will be estimated from these texture map triangles.

2. Some surface patches will appear in multiple images, and these patches should be corrected to the same final color and lightness because they came from the same physical scene surface. If the camera-based color distortion that affects these patches is corrected, then the correction will also apply to other patches in the connected images.

3. There are some planar patches in each image that allow color matching within the image and between images.

4. Some patches that should have the same color may be seen under different lighting conditions (*e.g.* direct *versus* indirect illumination).

5. There is at least one planar patch in all of the images that has good color.

Because our experiments involve color matching, it is essential to view the results in color. We have tried also to present monochrome evidence here, but the color results must be seen at URL:*http://www.ipab.informatics.ed.ac.uk/mvu/...*
NOBU/results.html.

2 Method

In this section we present a color correction method for texture map patches seen in multiple images. Assume a set of N texture map patches, *e.g.* $p_1 ... p_N$, some of which might be seen in multiple views. Construct a graph of patches, where the nodes represent texture map patches and arcs indicate they are two views of the same scene patch. Suppose a circuit in the graph exists, say A - B - C - ... Z - A, where A is the patch to register to. Then we may have a problem as the color correction of Z to A via B & C *etc.* may be different from that of Z to A directly. *I.e.*, a systematic accumulation of bias might occur. The algorithm proposed here spreads the average error over all arcs of the graph.

Before the global correction algorithm presented here is applied, several pre-processing steps are needed. In outline, the color correction process is as follows: (1) rectify and identify corresponding pixels in matched texture map regions; (2) adjust luminosity differences; (3) estimate pairwise color transform matrices and adjust patch colors initially; (4) apply the global color error minimization process. These steps are described in more detail below.

In the method presented below, the following notation is used:

i, j - image ids

r, s - region ids

p - pixel id

c - color id

x, y - pixel, region or image id

$\vec{x}_{i,r}^{c,p} = (r_r^p, g_r^p, b_r^p)$: an RGB color vector for pixel p of region r of color group c in image i

$L_{x \to y}^c$: a scalar lightness adjustment that changes the lightness of pixel, image or region x of color group c to that of pixel, image or region y.

$S_{x \to y}^c$: the initial pairwise 3x3 color correction matrix that changes the color of pixel, image or region x of color group c to that of pixel, image or region y.

$T_{x \to y}^c$: Like $S_{x \to y}^c$ except for this is a correction estimated by the global optimization algorithm

$T_{i,r \to s}^{c,d}$: Like $T_{r \to s}^c$ except for it is restricted to be a diagonal matrix and only applies to regions r and s of image i.

2.1 Image Rectification and Luminosity Adjustment

The color correction method presented below requires corresponding pixels. We can achieve this by two methods: 1) acquire the color in 3D, where each 3D point is acquired from a range sensor (or stereo) and has an attached RGB value, or 2) find corresponding pixels on planar surfaces in texture map images. The latter is the process used here, but the process below is applicable to both sensors.

To remove perspective distortions and identify corresponding pixels in the matched planar texture map regions, we estimated the homography [13, 9] that links the pixels. Here, we identified 4+ corresponding pixels by hand, but these could be found by a variety of image analysis algorithms. Let $\vec{z}$ and $\vec{Z}$ be the image coordinates of the same 3D point (in homogeneous coordinates) in the

Figure 1: A VRML model texture mapped using 30 images. The highlighted white line surrounds patches texture mapped from three different images seen from right, bottom and left directions. The dark oval surrounds a region common to all 3 source images. Many color discrepancies can be seen at the edges where planar faces meet, even though most surfaces in this view are the same color of sandstone.

original and transformed images respectively, linked by the 3x3 homography H: $\vec{Z} = H\vec{z}$. Other corresponding pixels can be found using matrix H. H is estimated using standard methods from [8].

The method applied in Section 2.3 assumes that all surfaces to be color corrected have approximately the same illumination conditions. When two regions are expected to have the same color, but do not due to differences in illumination (Figure 2), we apply the following initial luminousity correction on individual texture maps. In this case we do not have exactly corresponding pixels, so we choose pixels from patches that should have the same color.

There are several types of color space able to deal with luminosity directly, for example the HSV color space [6, 12]. To adjust the luminous intensity differences we: (1) transform pixel colors from RGB to HSV color space; (2) get the brightness value v of the p^{th} pixel in the adjusted ($v_{s,p}$) and reference ($v_{r,p}$) regions. The regions are selected arbitrarily but should have the same color distribution; (3) calculate a brightness adjustment parameter $L_{s\to r}$ from the adjusted region to the reference region; (4) adjust the brightness of all pixels in the adjusted image with $L_{s\to r}$; (5) transform HSV to RGB color space. To calculate $L_{a\to r}$, we used:

$$L_{s\to r} = \frac{\sum_p v_{r,p} v_{s,p}}{\sum_p v_{s,p} v_{s,p}} \tag{1}$$

The pixels $v_{s,i}$ and $v_{r,i}$ do not have to correspond, but should have the same approximate color. Equation 1 is the least square estimate that minimizes the error:

$$\sum_p \|v_{r,p} - L_{s\to r} v_{s,p}\|^2 \tag{2}$$

p indexes over paired pixels.

This process does an initial coarse correction of the luminosity, but not the chromaticity.

2.2 Initial Approximate Color Correction

Before using the global color correction method, pairwise color transformations $S_{r\to s}$ are used to initially adjust each region r's color to that of the reference region s. This speeds up the convergence of the global algorithm and also helps it avoid bad local minima. We use a pairwise color correction method based on [1].

2.3 Multiple Identical Patch Image Color Correction

Before correcting color discrepancies between N partially overlapping images, for simplicity, we start with 3 images A, B and C, but the theory below applies for any arbitrary topology of overlapping regions. Suppose A & B, B & C and A & C overlap and A has a good color appearance. Then, A's color space will be the reference and B's and C's color spaces should be transformed to A's color space. We find color transformations $T_{X\to A}$ ($X \in \{A, B, C\}$) that minimize the total color error over all overlapping pixels. The goal is to spread the color matching error across all images, so we use the squared error, which penalizes large discrepancies. The error is:

$$\sum_{r,s,p(r\neq s)} f(\|T_{r\to A}\vec{x}_r^p - T_{s\to A}\vec{x}_s^p\|^2)\phi(r,s,p) \tag{3}$$

where A is the reference color space image. $\vec{x}_r^p$ is the RGB color of pixel p in image r. $\vec{x}_s^p$ is the RGB color of pixel p in image s. $\phi(r,s,p) = 1$ if pixel p is seen in both image r and image s and 0 otherwise. We assume pixel p is the image of the same scene point in both images, which is summed over all corresponding samples in all overlapping images. .

Because there is the possibility of slight mis-alignment between the color patches, we use a robust selector $f()$ to eliminate outliers in the error formulation:

$$f(x) = \begin{cases} x & (x < \tau) \\ 0 & (otherwise) \end{cases} \tag{4}$$

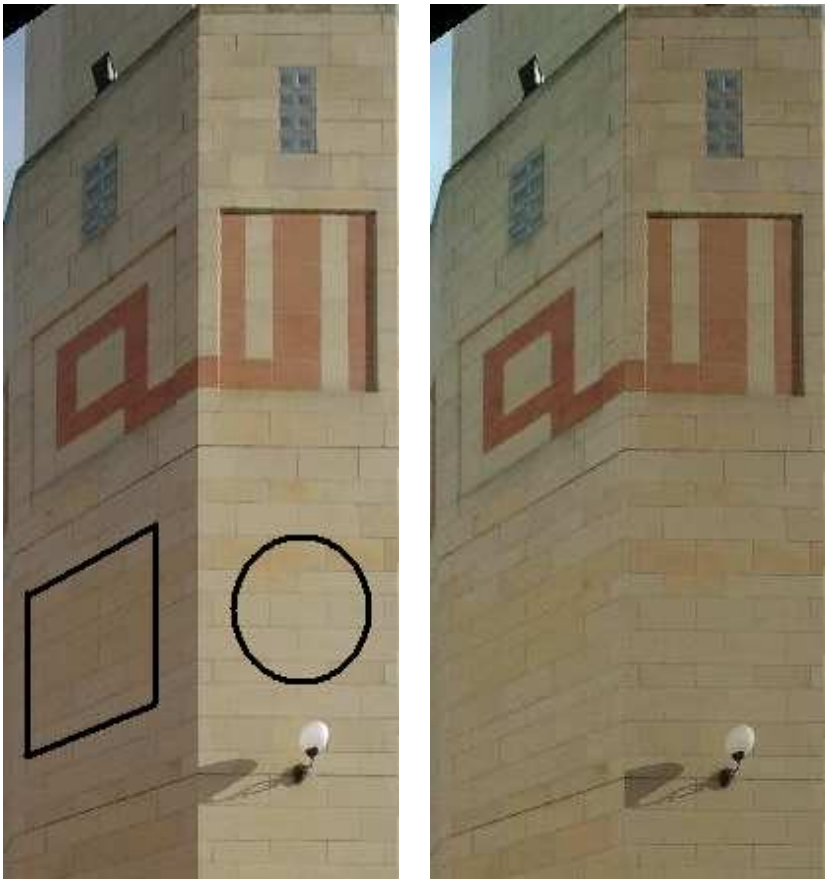

Figure 2: (left) A sample image with luminosity differences on the two surfaces. The circle and rectangle show two texture mapped regions respectively. There is a luminosity difference between the regions, though they are constructed from the same material. (right) The image after luminosity correction.

Where patches are slightly mis-aligned, *e.g.* at patch edges, overlapping pixels would tend to have quite different colors. Using these pixel correspondences in a normal least square process would bias the color transformation estimates. On the other hand, as the color spaces of the patches are already almost aligned, based on the methods discussed in Sections 2.1 and 2.2, color discrepancies between correctly overlapped pixels should be small. Hence, a threshold τ was used to separate large discrepancies (spatial mis-alignment) from small discrepancies (color discrepancies). Using the identity function in the case of small errors allows the least square process to estimate a transformation that minimizes the squared error. We did not experiment with other robust estimator functions, such as M-estimators that avoid the hard cut-off in the data selection [10].

2.4 Non-identical Patch Based Color Correction

The theory in the last section applies to two views of the same patch in two different images. There are at least two additional patch corrections possible:

1. Two different patches in the same image that are supposed to have the same color c. They could have different illumination (*e.g.* be on different sides of a building), which could also have a chromatic component due to mutual illumination. What the patches have in common is the same camera settings. As we do not have pixel-to-pixel correspondences here, we estimate a color transformation $T^{c,d}$ that converts pixel colors based only on the relation between the mean colors of the two patches. If $\vec{\mu}_X$ is the mean color of patch X and $\vec{\mu}_Y$ is the mean color of patch Y, then we want $\vec{\mu}_X = T^{c,d}_{Y\to X}\vec{\mu}_Y$. The estimated transformation assumes no crosstalk between colors and only a linear rescaling of the intensities in each color channel. So, here $T^{c,d}_{Y\to X}$ is a diagonal matrix (and hence the d superscript) with diagonal:

$$diag(T^{c,d}_{i,Y\to X}) = (\frac{\mu_{X,red}}{\mu_{Y,red}}, \frac{\mu_{X,green}}{\mu_{Y,green}}, \frac{\mu_{X,blue}}{\mu_{Y,blue}}) \tag{5}$$

2. Two unrelated patches seen in two different images, but which are supposed to have the same color. As with the previous case, we only link the patch colors by their mean colors.

2.5 Color Correction Methods Integrated

We combine all of the corrections introduced above by the following procedure. In the following, we assume that the raw texture images have already been acquired, texture patches (*e.g.* triangles) have been found and the mapping to 3D shape triangles has already occurred. Further, we assume some process exists that identifies: a) patches from different texture images that map to the same scene triangle, and b) patches that should have the same color. Let B^i be the set of all relevant texture patches in image i. Here, $T^{c,d}_{i,1\to 1} = T^c_{x\to x} = I_3$ to prevent degeneracies (I_3 is the 3x3 identity matrix).

Define a color group c as a set of patches that have the same distribution of pixel colors. We apply this correction process to each color group independently. For each color group c do:

1. Choose a reference image, call it image $i = 1$. Number the remaining images $i = 2 \ldots N$. Let the best patch in reference image 1 be called patch $r = 1$. All patches in the other images i are numbered arbitrarily $1 \ldots N_i$.

2. In each image i, find the sets $E^c_i = \{t \mid t \in B^i, color(t) = c\}$ of all texture map patches that should have the same color. This can be done by hand or by some external process.

3. In each pair of images i, j, find the sets $F^c_{i,j}$ of all pairs of texture map patches that are two views of the same scene patch for color group c: $F^c_{i,j} = \{(r,s) \mid r \in E^c_i, s \in E^c_j, same\_in\_3D(r,s)\}$. This can be done by hand or by some external process.

4. In each pair of images i, j, find the sets $G^c_{i,j}$ of all pairs of patches that should have the same color c, but which are not two views of the same patch: $G^c_{i,j} = \{(r,s) \mid r \in E^c_i, s \in E^c_j, (r,s) \notin F^c_{i,j}\}$. This can be done by hand or by some external process.

5. For each patch r of image i, compute and apply the luminance adjustment L as calculated in Sec. 2.1, using patch 1 of image 1 as the reference patch.

6. For each image i and color c, and each pair of patches $r, s \in E_i^c, r < s$, add this to error ϵ_1:

$$\|T_{i\to 1}^c(T_{i,r\to 1}^{c,d}\vec{\mu}_{i,r}^c - T_{i,s\to 1}^{c,d}\vec{\mu}_{i,s}^c)\|^2$$

   This balances the color of two patches in the same image.

7. For each pair of images $i, j, i < j$, for each pair of unmatched regions $(r, s) \in G_{i,j}^c$ add this to error ϵ_2:

$$\|T_{i\to 1}^c T_{i,r\to 1}^{c,d}\vec{\mu}_{i,r}^c - T_{j\to 1}^c T_{j,s\to 1}^{c,d}\vec{\mu}_{j,s}^c\|^2$$

   This balances the color of two patches in different images.

8. For each pair of images $i, j, i < j$ and color c, for each pair of matched regions $(r, s) \in F_{i,j}^c$ add this to error ϵ_3:

$$\frac{1}{N_{r,s}}\sum_p f(\|T_{i\to 1}^c T_{i,r\to 1}^{c,d}\vec{x}_{i,r}^{c,p} - T_{j\to 1}^c T_{j,s\to 1}^{c,d}\vec{x}_{j,s}^{c,p}\|^2)$$

   where $N_{r,s}$ is the number of pixels p in the overlap. This balances the color of two views of the same patch in different images. The error function links the corresponding pixels together in image 1's color space.

9. Initialize $T_{i,r\to 1}^{c,d}$ for patch r of image i using eqn (5). For each $(1, r) \in G_{1,j}^c$, initialize $T_{r\to 1}^c$ using the pairwise method of [1]. For each other $T_{r\to 1}^c$, there is a chain of matched patches $(r = r_1, r_2, \ldots, r_m = 1)$ that leads back to image 1. Estimate $T_{r_i\to r_{i+1}}^c$ using [1]. Then compute: $T_{r\to 1}^c = T_{r_1\to r_2}^c T_{r_2\to r_3}^c \ldots T_{r_{m-1}\to 1}^c$

The total error is $\epsilon = \epsilon_1 + \epsilon_2 + \epsilon_3$. Minimize ϵ over all of the transformations T.

In this procedure, we assume that every image is transitively connected to image 1 via a set of patches that correspond pixel-wise. This allows the definition of the full-rank transformation matrices.

2.6 Example

We illustrate the above theory with an example. Figure 3 shows three images that each have several patches in color group 1. Patch 1 and image 1 are selected as described above. Sets of patches are: $E_1^1 = \{1, 2, 3\}$, $E_2^1 = \{1, 2\}$ and $E_3^1 = \{1, 2\}$. Sets of overlapping patches are: $F_{1,2}^1 = \{(1, 1), (3, 2)\}$, $F_{1,3}^1 = \{(3, 1)\}$ and $F_{2,3}^1 = \{(2, 2)\}$. Non-overlapping patched are: $G_{1,2}^1 = \{(1, 2), (2, 1), (2, 2), (3, 1)\}$, $G_{1,3}^1 = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 2)\}$ and $G_{2,3}^1 = \{(1, 1), (1, 2), (2, 1)\}$.

Based on the above theory, we have 2 non-identity full rank matrices ($T_{2\to 1}^1$, $T_{3\to 1}^1$) and 4 diagonal non-identity transformation matrices ($T_{1,2\to 1}^{1,d}$, $T_{1,3\to 1}^{1,d}$, $T_{2,2\to 1}^{1,d}$, $T_{3,2\to 1}^{1,d}$). Together they contain $2 \times 9 + 4 \times 3 = 30$ degrees of freedom.

These matrices are involved in 5 error functions of type ϵ_1, 12 of type ϵ_2 and 4 of type ϵ_3:

$\epsilon_1 = \|T_{1\to 1}^1(T_{1,1\to 1}^{1,d}\vec{\mu}_{1,1}^1 - T_{1,2\to 1}^{1,d}\vec{\mu}_{1,2}^1)\|^2$
$+\|T_{1\to 1}^1(T_{1,1\to 1}^{1,d}\vec{\mu}_{1,1}^1 - T_{1,3\to 1}^{1,d}\vec{\mu}_{1,3}^1)\|^2$
$+\|T_{1\to 1}^1(T_{1,2\to 1}^{1,d}\vec{\mu}_{1,2}^1 - T_{1,3\to 1}^{1,d}\vec{\mu}_{1,3}^1)\|^2$
$+\|T_{2\to 1}^1(T_{2,1\to 1}^{1,d}\vec{\mu}_{2,1}^1 - T_{2,2\to 1}^{1,d}\vec{\mu}_{2,2}^1)\|^2$
$+\|T_{3\to 1}^1(T_{3,1\to 1}^{1,d}\vec{\mu}_{3,1}^1 - T_{3,2\to 1}^{1,d}\vec{\mu}_{3,2}^1)\|^2$

$\epsilon_2 = \|T_{1\to 1}^1 T_{1,1\to 1}^{1,d}\vec{\mu}_{1,1}^1 - T_{2\to 1}^1 T_{2,2\to 1}^{1,d}\vec{\mu}_{2,2}^1\|^2 + 11$ other terms

$\epsilon_3 =$
$\frac{1}{N_{1,1}}\sum_p f(\|T_{1\to 1}^1 T_{1,1\to 1}^{1,d}\vec{x}_{1,1}^{1,p} - T_{2\to 1}^1 T_{2,1\to 1}^{1,d}\vec{x}_{2,1}^{1,p}\|^2)$

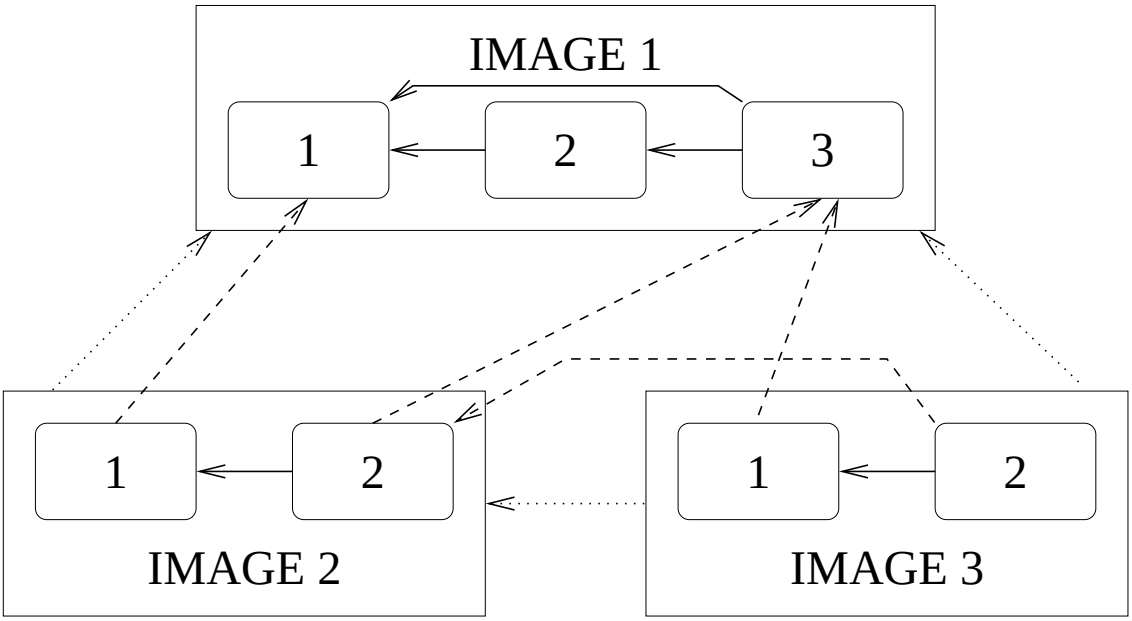

Figure 3: Example color transformation flows of the multiple patch based color correction. Solid lines imply "in-image" correction by patch mean colors and dashed lines imply pixelwise correction and dotted line implies imagewise correction., "Between image" correction by patch mean colors are omitted because there are 12 of these.

$$+\frac{1}{N_{3,2}}\sum_p f(\|T^1_{1\to1}T^{1,d}_{1,3\to1}\vec{x}^{1,p}_{1,3} - T^1_{2\to1}T^{1,d}_{2,2\to1}\vec{x}^{1,p}_{2,2}\|^2)$$
$$+\frac{1}{N_{3,1}}\sum_p f(\|T^1_{1\to1}T^{1,d}_{1,3\to1}\vec{x}^{1,p}_{1,3} - T^1_{3\to1}T^{1,d}_{3,1\to1}\vec{x}^{1,p}_{3,1}\|^2)$$
$$+\frac{1}{N_{2,2}}\sum_p f(\|T^1_{2\to1}T^{1,d}_{2,2\to1}\vec{x}^{1,p}_{2,2} - T^1_{3\to1}T^{1,d}_{3,2\to1}\vec{x}^{1,p}_{3,2}\|^2)$$

The final error to be minimized is $\epsilon = \epsilon_1 + \epsilon_2 + \epsilon_3$ over the 30 parameters of the T matrices.

2.7 Color Space Error Minimization

To estimate the color transformations that minimize the total color error E over all overlapping pixels and paired patches, the *MATLAB* minimization function *fminsearch* was used (this is a unconstrained nonlinear optimizer suitable for multidimensional spaces). The entries of the transformation matrices T were unfolded to form the state vector used by *MATLAB*. The initial values of T were initialized as in step 9 of Section 2.5.

3 Results and Discussion

In the experiments below, we evaluate the color correction on both real images with known, introduced color distortions, and real images with unknown color distortions. In the evaluations, we look at measures of the color discrepancies before correction, after the initial pairwise corrections introduced in [1], and the full multiple image correction presented here.

3.1 Experiments with Artificial Data Sets

Before the method was applied to real data sets, it was tested with artificially perturbed real data sets. The data sets were created by dividing a test image into 3 fragments, top (A), bottom left (B) and bottom right (C), with *Microsoft Photo Editor*. For example, see Figures 4 and 6. These images only show the overlapping regions. The images in the top and bottom rows in both figures show the test set before and after color correction respectively. The first data set (Figure 4) just had its color balance, brightness and contrast changed and the other (Figure 6) had some Gaussian noise added. The leftmost and middle images are slightly transformed versions of the rightmost image. We applied the multiple image color correction defined as in Section 2.5 to the set of images. Before using the multiple image color correction method, we found initial color transformations with the pairwise color correction method [1].

Figure 5 shows RGB color difference histograms before and after color correction. The computed differences are the pixelwise differences in value between the uncorrected and corrected images, over the 3 RGB color channels. What the histograms generally show is: (1) there are substantial differences between the intensities in all 3 color channels before correction (dotted lines have the peak far from zero) which are removed by the pairwise (dashed lines) and multiple image algorithms (solid lines), (2) the previous pairwise process and the new multiple image process work comparably on pairs of images (top and middle rows - the histograms for the corrected data all have peaks near zero) and (3) the new multiple image algorithm produces better results when multiple images are considered (bottom row).

The top row of Figure 5 shows the RGB color difference histograms between the original undistorted image (top right in Figure 4) and (a) the corrupted image 1 (top left in Figure 4) with the dotted line here, (b) the corrected image using the pairwise algorithm [1] (image not shown) with the dashed line here and (c) the corrected image using the new algorithm (bottom left in Figure 4) with the solid line here. These histograms show that both algorithms effectively remove the errors (peaks of corrected images are near zero). The middle row of Figure 5 shows similar RGB color difference histograms between the original undistorted image (top right in Figure 4) and the corrupted image 2 (top middle in Figure 4). The conclusions are the same as for the top row.

The bottom row of Figure 5 shows the RGB color difference histograms between corrupted images 1 and 2 (dotted line: the original noisy images), (dashed line: the corrections using the pairwise algorithm) and (solid line: the corrections using the new algorithm). The dotted line shows that there are substantial differences between the corrupted images, the dashed line shows that the these can be removed fairly well by the pairwise algorithm, but not completely as the peak is offset from zero (recall that these 2 images were not pairwise corrected, rather they were individually corrected by comparison to the original undistorted image). The solid line shows that the multiple image correction process (which includes all three images from the top row of Figure 4) has produced better results as its peak is nearer to zero. The color differences between images 1 and 2 after the multiple image color correction are smaller than after the pairwise color correction and the color differences between the reference image and image 1, image 2 are almost the same after optimization. Thus, it is clear that the multiple patch optimization process improves the overall global color balance without greatly worsening the pairwise adjustment.

Figure 7 shows similar results to Figure 5, except here the images had Gaussian noise added before corrections. Note that the scale of these graphs is quite different from those of Figure 5, because of the large amount of Gaussian noise. The change in scale masks the improvements in the histogram peak positions, but the same improvements occur. The allow us to make the same conclusions as the noise-free figures, and also conclude that the addition of noise does not make a substantial difference to the correction process.

Table 1 shows the accumulated color error between images A (reference image), B (image 1) and C (image 2) before and after color corrections. The error accumulation ΔUV between images U and V (where $U, V \in \{A, B, C\}$) is the pixelwise difference in colors, as given by:

$$\Delta UV = \Delta UV_{red} + \Delta UV_{green} + \Delta UV_{blue}$$

where

$$\Delta UV_{red} = \sum_p \|\vec{x}_U^{red,p} - \vec{x}_V^{red,p}\|$$

$$\Delta UV_{green} = \sum_p \|\vec{x}_U^{green,p} - \vec{x}_V^{green,p}\|$$

$$\Delta UV_{blue} = \sum_p \|\vec{x}_U^{blue,p} - \vec{x}_V^{blue,p}\|$$

ΔUV_{red}, ΔUV_{green} and ΔUV_{blue} are R, G and B error accumulation between images U and V. $\vec{x}_U^{red,p}$, $\vec{x}_U^{green,p}$, $\vec{x}_U^{blue,p}$, $\vec{x}_V^{red,p}$, $\vec{x}_V^{green,p}$ and $\vec{x}_V^{blue,p}$ are the R, G and B values of pixel p in images U and V respectively. The table shows the pixelwise accumulated color differences between each pair of images, for both algorithms (and before correction) and the details for the individual color channels. We can observe: (1) there is initially a lot of difference in pixel values between images ("Before" values in the rightmost column), (2) the pairwise corrections makes a big improvement, but still leaves a lot of error in ΔBC, and (3) the multiple image correction slightly degrades ΔAB and ΔAC while making a big improvement in ΔBC.

Table 2 shows the same error accumulations for the case where noise was added. Here, there is still a substantial amount of noise remaining, but the multiple image correction slightly degrades ΔAC while making a bigger improvement to ΔBC (and also slightly improves ΔAB).

We show below the color transformation matrices that were estimated from the previous pairwise image algorithm ($S_{B\to A}$ and $S_{C\to A}$) and the color transformation matrices that were estimated from new multiple image ($T_{B\to A}$ and $T_{C\to A}$) color corrections for each original and noise corrupted data set. While the actual values in the matrices are not too important, they help us to make several points: (1) there is a significant change in the chromatic balance between images, not simply a rescaling of each individual color channel (as there are significant off-diagonal values) and (2) a small amount of noise does not greatly affect the process: the two sets of matrices are roughly similar (mean Frobenius norm of 0.29 as contrasted with mean Frobenius norm 1.73 for two randomly generated 3x3 matrices with entries from [-1,1]).

Estimated for data sets without noise:

$$S_{B\to A} = \begin{pmatrix} 0.7355 & 0.3454 & -0.1235 \\ 0.2401 & 0.7099 & 0.0908 \\ -0.0957 & 0.1060 & 0.9123 \end{pmatrix}$$

$$S_{C\to A} = \begin{pmatrix} 0.7453 & 0.3568 & -0.1410 \\ 0.2098 & 0.6845 & 0.1408 \\ -0.0695 & 0.1151 & 1.0186 \end{pmatrix}$$

$$T_{B\to A} = \begin{pmatrix} 0.6596 & 0.4628 & -0.1692 \\ 0.3219 & 0.5848 & 0.1371 \\ -0.1301 & 0.1607 & 0.8878 \end{pmatrix}$$

$$T_{C\to A} = \begin{pmatrix} 0.6316 & 0.4916 & -0.1881 \\ 0.2865 & 0.5933 & 0.1727 \\ -0.0866 & 0.1372 & 1.0058 \end{pmatrix}$$

Estimated for data sets with noise:

$$S_{B\to A} = \begin{pmatrix} 0.6377 & 0.4625 & -0.0859 \\ 0.3537 & 0.5295 & 0.1528 \\ -0.0455 & 0.0475 & 0.7715 \end{pmatrix}$$

$$S_{C\to A} = \begin{pmatrix} 0.7164 & 0.3591 & -0.0747 \\ 0.3717 & 0.4955 & 0.1956 \\ -0.0052 & 0.0327 & 1.0189 \end{pmatrix}$$

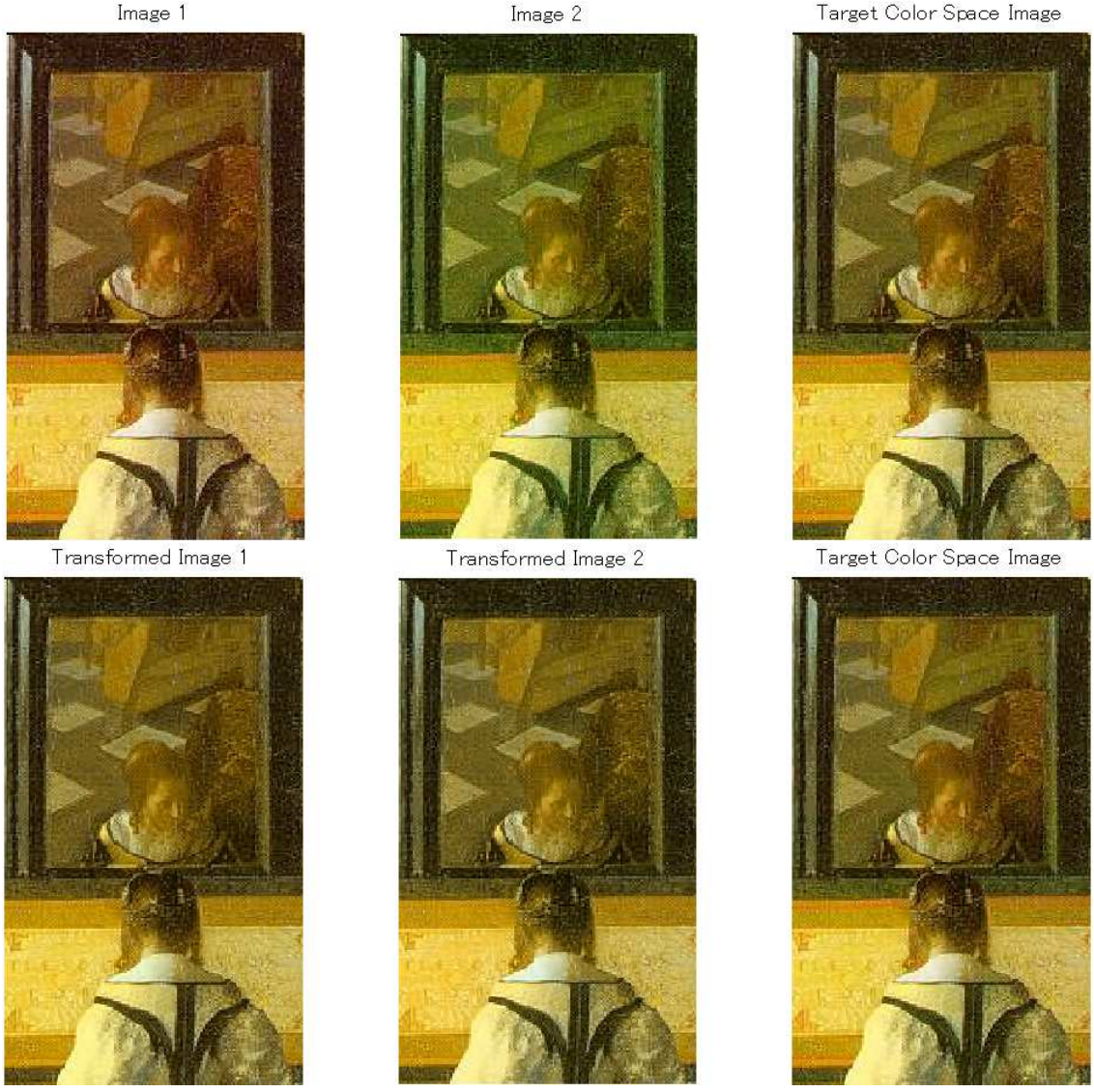

Figure 4: The top row shows the test image 1 (left), test image 2 (middle) and the reference image (right) before color correction. The bottom row shows the result images after color correction. No noise is added to the test images.

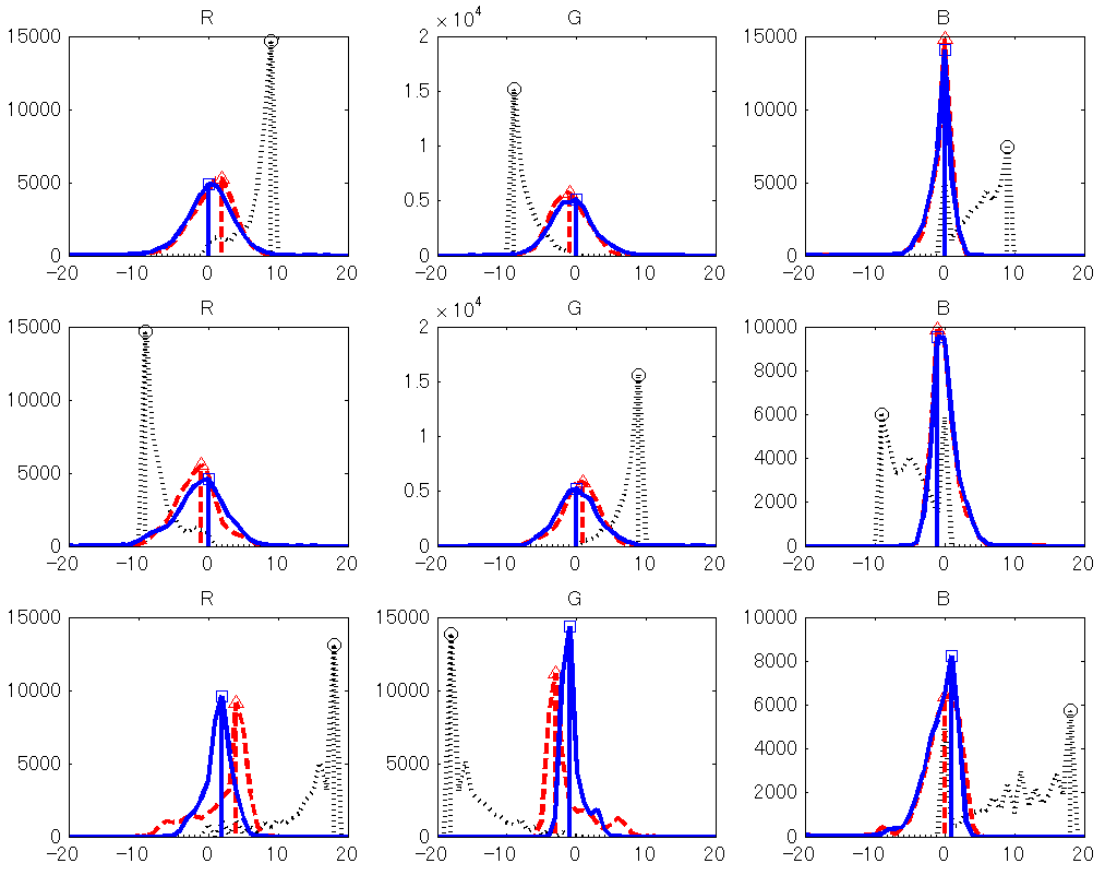

Figure 5: RGB color difference histograms of the artificial data sets without noise before and after the color corrections. The top row graphs show RGB color difference histograms between the reference image and image 1. The middle row shows those between the reference image and image 2. The bottom row shows those between images 1 and 2. The dotted and solid lines in the histograms show RGB color differences before and after the color correction. The dashed lines show results after only pairwise color correction. Refer to main text for discussion of results.

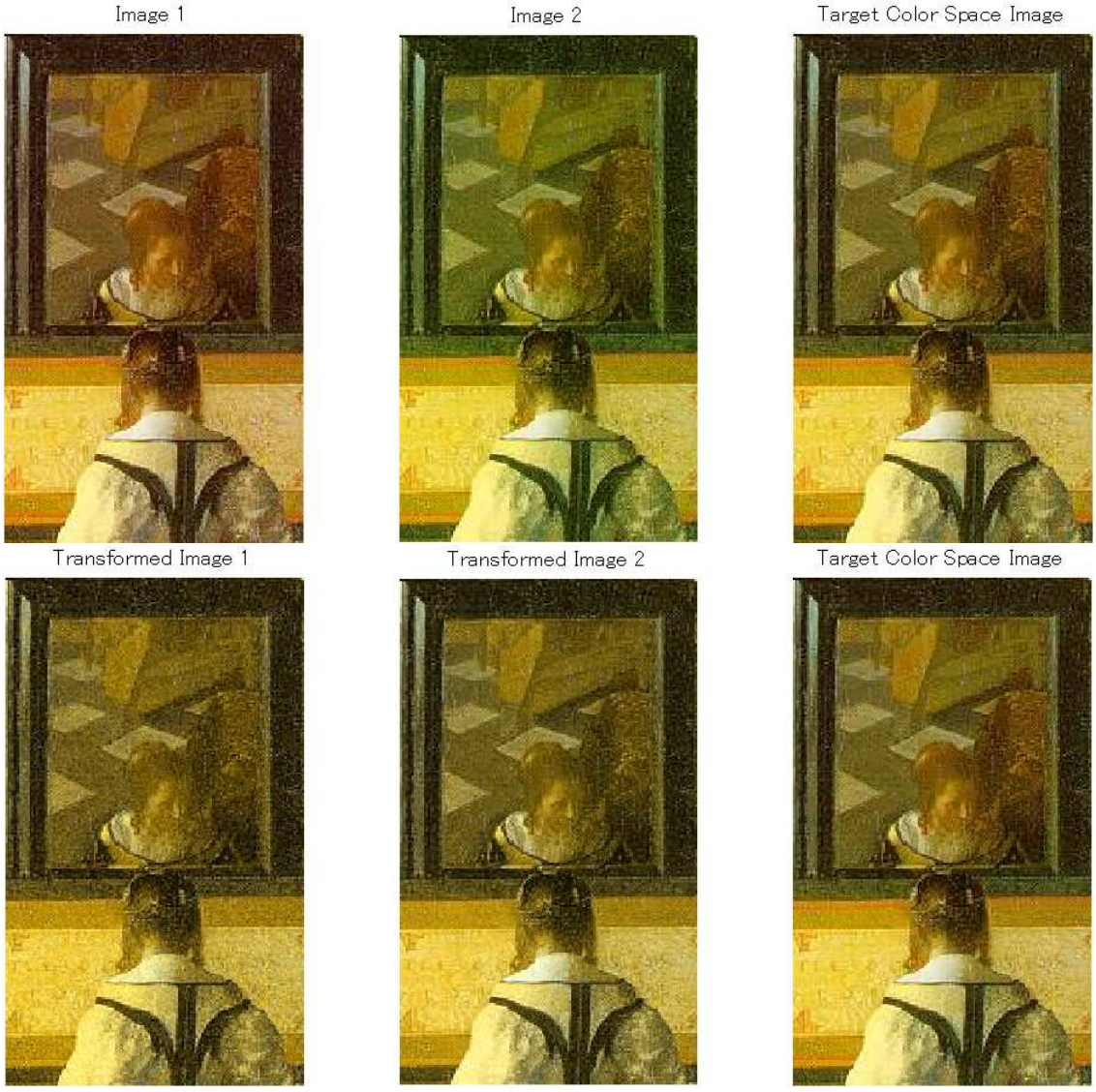

Figure 6: The top row shows test image 1 (left), test image 2 (middle) and the reference image (right) before color correction. The bottom row shows the result images after color correction. Gaussian noise is added to the test images.

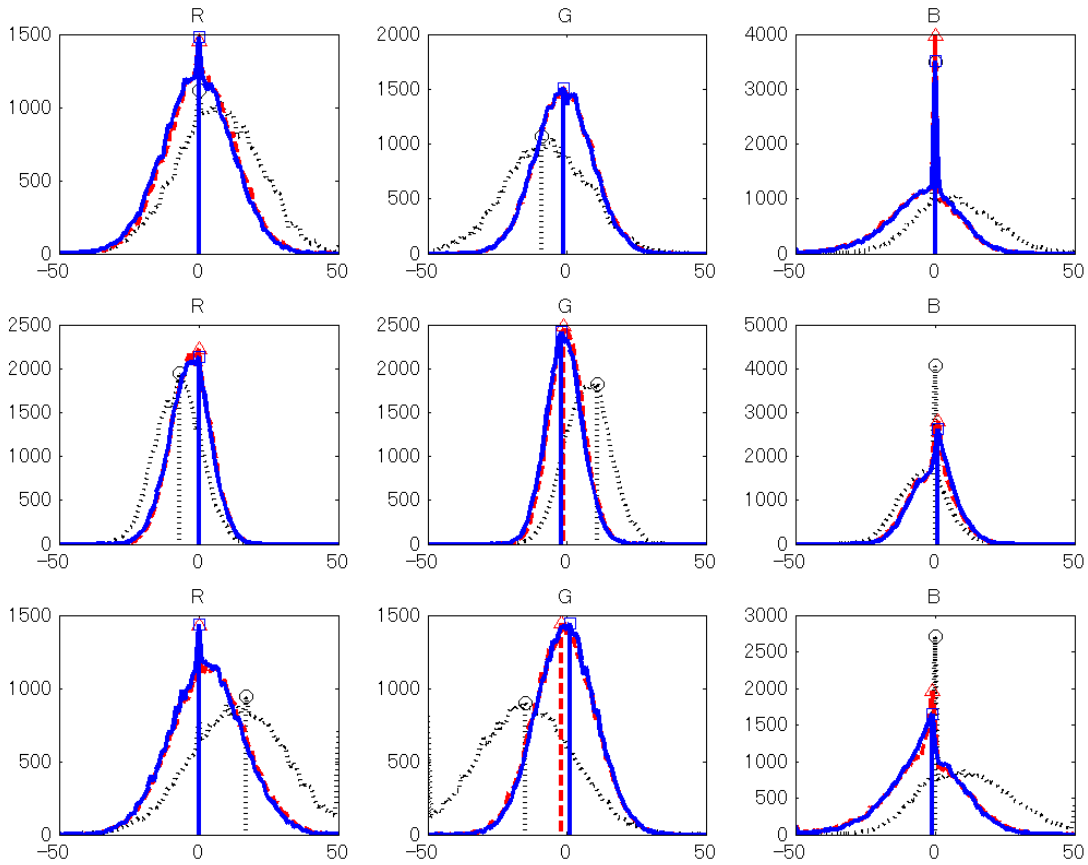

Figure 7: RGB color difference histograms of the artificial data sets with noise before and after the color corrections. The top row graphs show RGB color difference histograms between the reference image and image 1. The middle row shows those between the reference image and image 2. The bottom row shows those between images 1 and 2. The dotted and solid lines in the histograms show RGB color differences before and after the color correction. The dashed lines show results after only pairwise color correction. Refer to main text for discussion of results.

Table 1: Error accumulations with an artificial data set without noise.

Between images A and B	ΔAB_{red}	ΔAB_{green}	ΔAB_{blue}	ΔAB
Before color correction	267725	279374	199646	746745
After pairwise color correction	94876	85674	40225	220775
After multiple color correction	99702	92157	44322	236181
Between images A and C	ΔAC_{red}	ΔAC_{green}	ΔAC_{blue}	ΔAC
Before color correction	267854	281863	184799	734516
After pairwise color correction	103834	82042	48917	234793
After multiple color correction	112159	89853	49257	251269
Between images B and C	ΔBC_{red}	ΔBC_{green}	ΔBC_{blue}	ΔBC
Before color correction	535579	561237	384445	1481261
After pairwise color correction	144080	113924	75192	333196
After multiple color correction	76053	53918	70185	200156

Table 2: Error accumulations with an artificial data set with noise.

Between images A and B	ΔAB_{red}	ΔAB_{green}	ΔAB_{blue}	ΔAB
Before color correction	491877	502577	430804	1425258
After pairwise color correction	363001	307692	378626	1049319
After multiple color correction	367116	305717	369160	1041993
Between images A and C	ΔAC_{red}	ΔAC_{green}	ΔAC_{blue}	ΔAC
Before color correction	335462	335332	265927	936721
After pairwise color correction	227648	186988	227497	642133
After multiple color correction	235630	194703	226418	656751
Between images B and C	ΔBC_{red}	ΔBC_{green}	ΔBC_{blue}	ΔBC
Before color correction	683675	693771	560021	1937467
After pairwise color correction	406481	324966	414531	1145978
After multiple color correction	392052	318100	398122	1108274

$$T_{B\to A} = \begin{pmatrix} 0.5889 & 0.5082 & -0.0661 \\ 0.3791 & 0.4842 & 0.1738 \\ -0.0429 & 0.0708 & 0.7280 \end{pmatrix}$$

$$T_{C\to A} = \begin{pmatrix} 0.6598 & 0.4072 & -0.0491 \\ 0.4053 & 0.4437 & 0.2334 \\ -0.0008 & 0.0501 & 0.9620 \end{pmatrix}$$

3.2 Experiments with Real Data

To show color correction performances with real data sets, we show a VRML model textured with three original images taken from left (image X), front (image Y) and right (image Z) directions in Figure 8. The black rectangles in the left image show color discrepancies at image boundaries. To remove the color discrepancies between the images, the patch based color correction method presented in Section 2.5 was applied. Initial color transformations were found by using [1] between images X and Y and between Y and Z independently. As shown in Figure 8 (bottom), the color correction method effectively removes color discrepancies.

It takes about 3 hours to compute all color transformation matrices for the patch based color correction with 500 pixels from each overlapping patch with a laptop computer (*Intel Pentium III*

1 GHz CPU, 512MB RAM, 40GB HDD) running *MATLAB* under *Microsoft Windows XP Home Edition*.

We show another VRML model textured with eight original images for all sides of a small tower in Figure 10. Image numbering starts from the top leftmost image in a clockwise direction. Each image has three patches. Patches $p1$, $p2$ and $p3$ are in image 1, patches $p4$, $p5$ and $p6$ are in image 2, patches $p7$, $p8$ and $p9$ are in image 3, *etc.* The front face of the tower (the top leftmost image in Figures 10 and 11, *i.e.* patch $p2$ in Figure 9) is the reference color space and the other patch color spaces were transformed to the front face's ($p2$'s) color by using the multiple patch based color correction defined in Sec. 2.5. Figure 9 shows patch relations between the images. The patches connected with arrows in Figure 9 overlap between the images. The arrow directions show color transformation paths to remove color discrepancies between the images. As seen in Figure 11 (result), the color discrepancies at image boundaries were almost removed.

It takes about 7 hours to compute all color transformation matrices for the patch based color correction with 50 pixels in each overlapping patch on the same computer environment as before.

The result is not perfect because the image rectification may be slightly inaccurate and weatherings on the object surfaces in the images mislead the algorithm. Moreover, the subsampled pixels which were chosen to estimate color transformation matrices and the number of them may not be always be the best (*i.e.* we may have selected some particularly noisy pixels). However, the color discrepancies are almost removed across all images.

Figure 12 shows another VRML model textured with 30 original and color corrected images. The color correction method was successful and a good appearance model was constructed.

3.3 Summary and Conclusions

We summarize and conclude this work as follows:

- A color correction method for multiple view textures having overlapping regions was proposed and some results were shown.
- The proposed method works as supported by both the visual evidence and histogram analysis.
- To obtain a good result, good image registration and high resolution images are necessary.
- Highlights, shadows and weatherings are still problems when removing color discrepancies correctly. Shadows may be removed by using Finlayson's technique [7].
- The homography based registration method is useful for planar regions, which makes the method particularly suitable for many architectural scenes. Further work on shading correction is needed for general curved surfaces. However, color correction applied to a planar patch of a texture map can often be useful on nearby curved or finely triangulated surfaces.
- The current Matlab implementation is slow. Recoding into `C++` should allow $50 - 100$ times improvement in speed, allowing either increased numbers of pixels to be used or faster optimization.
- Currently the process only applies to each color group independently. Since multiple texture map patches are acquired in a single image, some correction matrices could be shared and optimized simultaneously. This would allow more consistent colors between different color groups.

Acknowledgements

Many thanks to *Canon Inc.* and *Edinburgh Central Mosque* for support on this project.

Figure 8: A VRML model textured with three original images (left) and color corrected images (right). The black rectangles in the left image show color discrepancies at image boundaries. Color discrepancies between images were almost removed after the color correction.

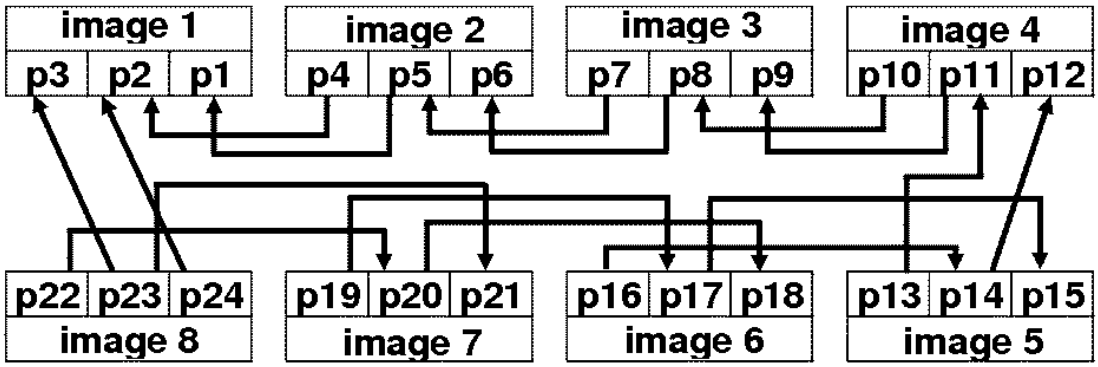

Figure 9: Corresponding patch relations between the images. Figure 10 shows the input images and some of the patches. Arrows show directions of color transformations between patches with overlap.

References

[1] A.Agathos and R.B.Fisher. *Colour Texture Fusion of Multiple Range Images.* Proc. 4th Int. Conf. on 3-D Digital Imaging and Modeling, Banff, pp 139-146, 2003.

[2] K.Barnard, G.Finlayson and B.Funt. *Colour Constancy for Scenes with Varying Illumination.* A survey, Computer Vision and Image Understanding, Vol. 65(2), pp 311-321, 1997.

[3] A.Baumberg. *Blending images for texturing 3D models.* Proc. Conf. on British Machine Vision Association, Cardiff, pp 404-413, 2002.

[4] E.Beauchesne and S.Roy. *Automatic Relighting of Overlapping Textures of a 3D Model.* Proc. Conf. on Computer Vision and Pattern Recognition, Madison, Vol. 2, pp 139-146, 2003.

[5] D. H. Brainard and W. T. Freeman. *Bayesian Color Constancy.* J. Optical Society of America, A, 14(7), pp. 1393-1411, July, 1997.

[6] D.Cardani. *Adventure in HSV Space. Retrieved: February 2004, available from: http://www.buena.com/articles/hsvspace.pdf.* BUENA SOFTWARE INC., 2001.

[7] G.D.Finlayson, M.S.Drew and C.Lu. *Intrinsic Images by Entropy Minimization.* Proc. 8th European Conf. on Computer Vision, Prague, pp 582-595, 2004.

[8] R. Hartley, A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[9] P.Heckbert. *Projective Mappings for Image Warping. Master's thesis, University of California, Berkeley, 1989. Retrieved: December 2003, available from: http://www-2.cs.cmu.edu/ ph/869/www/notes/proj/proj.pdf.* School of Computer Science, Carnegie Mellon University, 2001.

[10] William J.J. Rey. *Introduction to Robust and Quasi-Robust Statistical Methods.* Springer, Berlin, Heidelberg, 1983.

[11] B. Singh, W. T. Freeman, and D. H. Brainard. *Exploiting spatial and spectral image regularities for color constancy.* 3rd Intl. Workshop on Statistical and Computational Theories of Vision (associated with Intl. Conf. on Computer Vision), Nice, France, October, 2003.

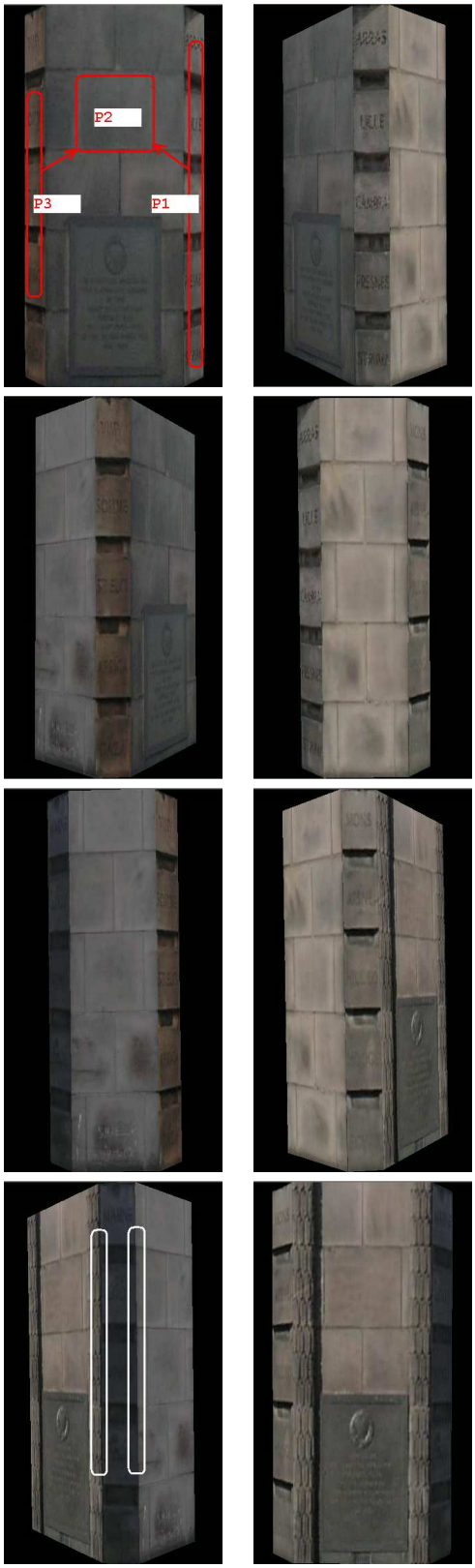

Figure 10: A VRML model textured with eight real images before the color correction. Image 1 is the top leftmost image. The image numbering starts from the top leftmost in a clockwise direction. There are some obvious color discrepancies at image boundaries (*e.g.* white rectangles in image 6).

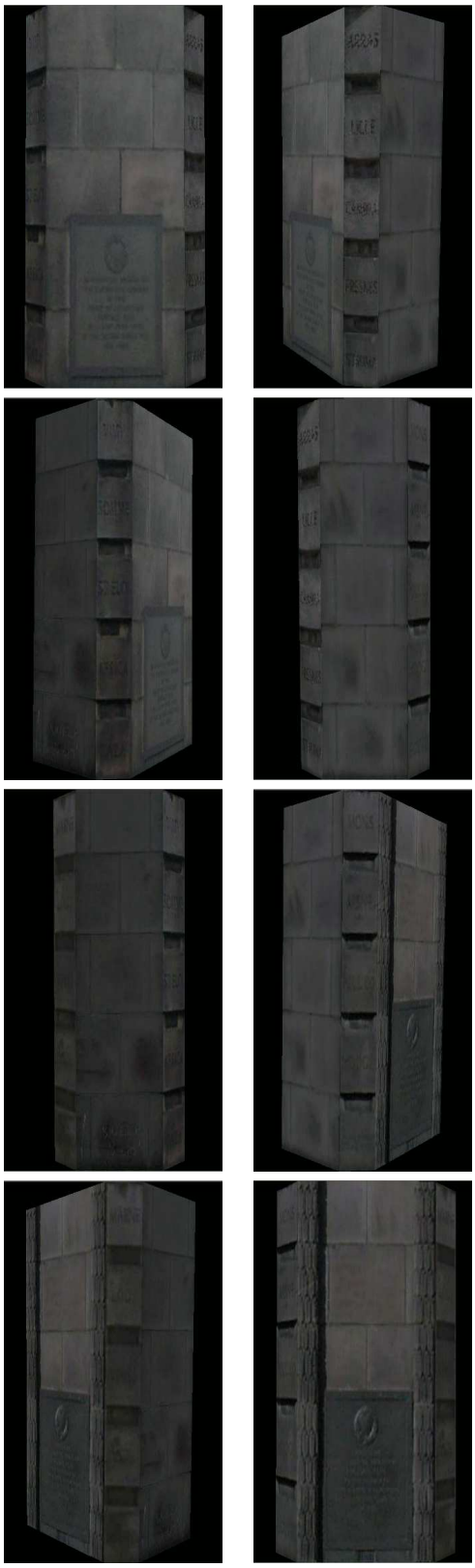

Figure 11: A VRML model textured with eight real images after the patch based color correction. Image 1 is the top leftmost image. The image numbering starts from the top leftmost in a clockwise direction. Color discrepancies at image boundaries were almost removed.

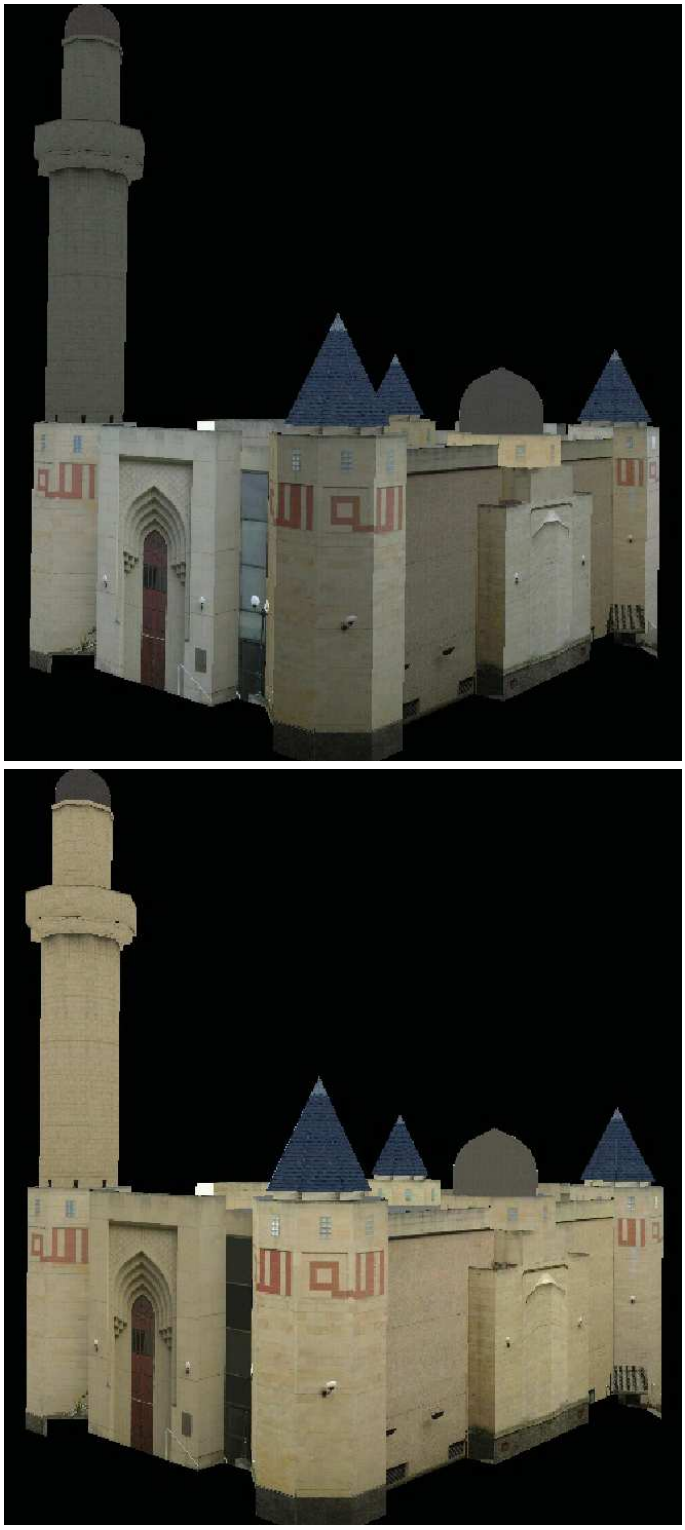

Figure 12: A VRML model textured with 30 original images (top) and color corrected images (bottom). The appearance of all corrected stone surfaces is much more consistent than the uncorrected model.

[12] S.Sural, G.Qian and S.Pramanik. *Segmentation and Histogram Generation Using the HSV Color Space for Image Retrieval*. Proc. Int. Conf. on Image Processing, Vol. 2, pp 589-592, 2002.

[13] I.Zoghlami, O.Faugeras and R.Deriche. *Using geometric corners to build a 2D mosaic from a set of images*. Proc. Conf. on Computer Vision and Pattern Recognition, pp 420-425, 1997.