Fast and Stable Color Balancing for Images and Augmented Reality

Thomas Oskam [1,2] Alexander Hornung [1] Robert W. Sumner [1] Markus Gross [1,2]

[1] Disney Research Zurich [2] ETH Zurich

***Abstract*—This paper addresses the problem of globally balancing colors between images. The input to our algorithm is a sparse set of desired color correspondences between a source and a target image. The global color space transformation problem is then solved by computing a smooth vector field in CIE Lab color space that maps the gamut of the source to that of the target. We employ normalized radial basis functions for which we compute optimized shape parameters based on the input images, allowing for more faithful and flexible color matching compared to existing RBF-, regression- or histogram-based techniques. Furthermore, we show how the basic per-image matching can be efficiently and robustly extended to the temporal domain using RANSAC-based correspondence classification. Besides interactive color balancing for images, these properties render our method extremely useful for automatic, consistent embedding of synthetic graphics in video, as required by applications such as augmented reality.**

I. Introduction

Pablo Picasso highlighted the importance of color in painting with the famous quote, *"They'll sell you thousands of greens. Veronese green and emerald green and cadmium green and any sort of green you like; but that particular green, never."* Today, the same struggle to capture the perfect nuance of color lives on in digital photography and image-based applications. The colors that are ultimately recorded by both consumer-grade and high-end digital cameras depend on a plethora of factors and are influenced by complex hardware and software processing algorithms, making the precise color quality of captured images difficult to predict and control. As a result, one generally has to rely on post-processing algorithms to fix color-related issues.

This problem is generally known as color balancing or color grading, and plays a central role in all areas involving capturing, processing and displaying image data. In digital photography and filmmaking, specifically trained artists work hard to achieve consistent colors for images captured with different cameras, often under varying conditions and even for different scenes. With today's tools this process requires considerable, cost-intensive manual efforts. Similar issues arise in augmented reality applications, where computer generated graphics must be embedded seamlessly and in real-time into a video stream. Here, achieving consistent colors is critical but highly challenging, since white-balance, shutter time, gamma correction, and other factors may continually change during acquisition and cannot be perfectly controlled or addressed through calibration alone.

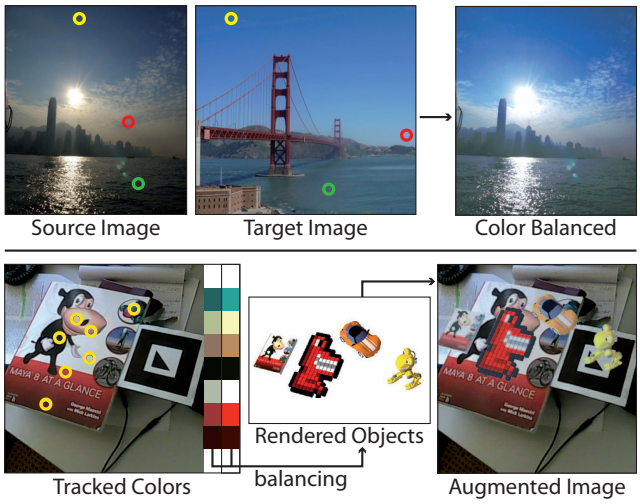

Figure 1. Two applications of our color balancing algorithm. Top: an underexposed image is balanced using only three user selected correspondences to a target image. Bottom: our extension for temporally stable color balancing enables seamless compositing in augmented reality applications by using known colors in the scene as constraints.

Therefore, an efficient correction and adaptation of colors that is agnostic to the underlying capture hardware is highly desirable in such AR-related applications. However, aside for a few preliminary efforts [1], [2] practically applicable solutions to this problem are yet to be developed.

This paper contributes a real-time color balancing algorithm that is able to globally adapt the colors of a source image to match the look of a target image, requiring only a sparse set of color correspondences. Fundamentally, our approach is based on radial basis function (RBF) interpolation. However, we show that, for an optimal RBF-based color matching, it is crucial to optimize for the shape of the employed basis functions. As a second main contribution, we then extend the per-image computation to be temporally consistent, enabling the application of the basic balancing mechanism in video-based applications such as augmented reality. A key component to the success of this system is a reliable RANSAC-based classification algorithm that selects a temporally stable set of correspondences used for the basic color balancing. We present a fast GPU-assisted implementation of our color balancing algorithm that achieves run times of 1ms and below on real-world applications. In this paper and the accompanying video, we show results for interactive

978-0-7695-4873-9/12 $26.00 © 2012 IEEE
DOI 10.1109/3DIMPVT.2012.36

color editing of images as well as for improved visual quality and consistency of video and computer generated images in augmented reality applications.

II. Related Work

In this section, we discuss related work on histogram matching, user controlled color transfer, and color calibration.

Statistics and Histogram Matching: This is the process of transferring the color distribution statistics from one image to another image, pioneered by Reinhard et al. [3]. They proposed computing the mean and standard deviation of the color distribution in the color space. Then, by scaling and translating the parameters onto the ones of the target image, they achieve automatic color transfer. The approach has been applied to histograms and extended to achieve a mapping in different color spaces [4], minimum cost linear mapping [5], and per-region execution of the algorithm [6]. Further extensions have been proposed that optimize the images after matching to avoid discontinuities in color gradients [7], [8].

While statistics and histogram matching methods for automatic color transfer can produce very appealing results, they provide only limited control. Direct control over color changes is often required in image editing and difficult to achieve with these methods.

Color transfer with user interaction: Abadpour et al. [9] first proposed an image color transfer algorithm with user interaction. In their work, the user is allowed to denote corresponding regions between images. Colors are then locally matched using fuzzy principal component analysis. Other methods have been proposed that aid the user in local color matching [10], [11]. The user is allowed to select a local region, and the actions are propagated throughout adjacent parts of the image. An et al. [12] propose a scribble-based framework for color transfer between images. Fast and interactive methods for editing images and video have also been proposed. An et al. [13] propagate local edits by assisting a user with iteratively refining rough edits to areas of spatial and appearance similarity. Farbman et al. [14] propose the use of diffusion maps to propagate image editing information. In these approaches the impact of user controlled edits is typically targeted towards local color changes. Our work, in contrast, targets global color balancing using only sparse and possibly incomplete input.

Li et al. [15] provide an approach for image and video editing with instant results. They employ radial basis function interpolation in a feature space to propagate sparse edits locally. In contrast to this work, we are formulating the problem as a global color space transformation independent from the image space and we optimize the shape of basis functions to achieve color balancing behavior found in example images. We show that the basis function used by Li et al. is suboptimal for the purpose of global image balancing.

Color calibration and balancing: Color calibration is the process of computing the color distribution function of a camera to allow recorded images to be consistently colored. Adams et al. [16] gives an overview of standard camera calibration techniques using linear transformations. Usually, a 3x3 matrix is computed using regression. A related area is color consistency. Humans perceive colors to be consistent even under different lighting conditions. This property is not available for digital images. Colors have different numerical values under different illuminations. In order to counter this issue, different algorithms have been proposed. An overview over different algorithms is provided by Argawal et al. [17] and Cohen [18]. In contrast to our approach, many of these methods do not apply additional information such as correspondences or known colors in a scene to balance the images. Therefore, these methods are less interactive and less robust when applied to videos.

There is also significant work that enhances the quality of imagery using a database of example images [19], [20], [21], [22]. The work of Yang et al. [23] achieve color compensation by replicating the non-linear relationship between the camera color and luminance. Wang et al. [24] transfer color and tone style of expert photographed images onto the images from cheap cameras by learning the complex transformation encoded in local descriptors. Recent work of Lin et al. [25] revisits the radiometric calibration of cameras by investigating 10,000 images from over 30 cameras and introduces a new imaging model for computing the color response curves. Our parameter optimization of the radial basis functions is also based on extracting image properties form data sets. However, we design our functions to be sparsely sampled so that they provide appealing results when interpolated in order to considerably increase performance.

Color Balancing in Augmented Reality: Only recently there has been research in improving the compositing of images in augmented reality. Klein et al. [1] examine and reproduce several image degradation effects of digital cameras to increase the realism of augmentation. However, they are not considering color shifts due to camera parameter changes. The works of Knecht et al. [2] propose to balance rendered images using colors in the scene. However, they are applying a linear regression model and do not correct for corrupted colors due to occlusions or reflections.

III. Vector Space Color Balancing

Given a sparse set of color correspondences, the goal is to transform the color gamut of an image such, that (i) colors that have references get as close as possible to that point in the color space, and (ii) colors, for which no references are available, are transformed in a plausible way.

We interpret the given color correspondences as vectors in the color space. In order to produce a transformation of

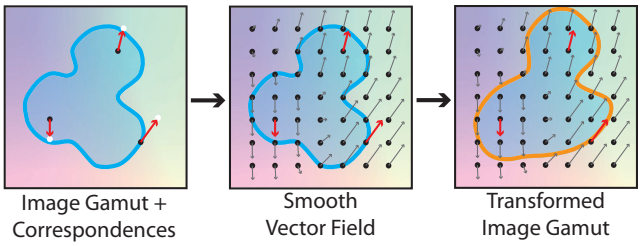

Figure 2. The idea of vector space color balancing. On the left, the source colors are shown in black and the corresponding target colors in white. The idea is to interpret all correspondences as vectors in color space and smoothly extend them to a vector field with which the gamut of the image is transformed. This procedure creates a continuous and flexible color transfer function.

the gamut, we compute a vector field around the reference vectors that achieves the aforementioned goals. The concept is demonstrated in Figure 2. The correspondence vectors themselves represent constraints. We use normalized radial basis function interpolation that propagates these constraints to the rest of the color space. Section III-A explains how this is performed. The selection of basis functions, however, influences how the correspondence vectors are propagated through the color space and ultimately how colors without constraints are balanced. Therefore, we propose optimizing the shape of a basis function based on example images. This is discussed in section III-B.

A. Vector Field Computation

For a given pair of colors (c_i, d_i) in the three-dimensional CIE Lab space, we define the c_i as support points, and the vectors $\mathbf{v}_i = ||d_i - c_i||$ as data values. For each vector $\mathbf{v}_i$, a basis function ϕ_i is provided that describes the weight with which the vector is distributed in space. For each new point e in the color space, an according translation vector $\mathbf{v}(e)$ is computed as a normalized sum of the weighted ϕ_i

$$\mathbf{v}(e) = \frac{1}{\sum_{i=1}^{n} \phi_i(e)} \sum_{i=1}^{n} \phi_i(e)\mathbf{w}_i. \qquad (1)$$

The summation of the support functions allows an independent evaluation of each point in the vector field, which suits a completely parallel implementation for each pixel in an image. The individual weights $\mathbf{w}_i$ allow the treatment of the ϕ_i as radial basis functions (RBF). RBF interpolation performs a least squares minimization on the overlapping support between the individual ϕ_i by choosing weights $\mathbf{w}_i$ such, that the squared difference between the data vectors $\mathbf{v}_i$ and the weighted sum of the basis functions at c_i is minimized in the least squares sense

$$\underset{\mathbf{w_i}}{\operatorname{argmin}}(||\mathbf{v}_i - \mathbf{v}(c_i)||^2). \qquad (2)$$

This results in a system of equations $v = \Phi w$ per color dimension, where the matrix Φ is the same in all three systems and contains, at each position (i, j), the

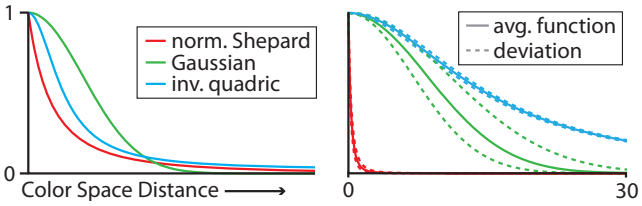

Figure 3. Plots of the basis functions we consider for our color transfer. On the left, the three functions are shown in relative scale. Note their different fall-off behaviors. On the right, the curves are shown with optimized shape parameters in their absolute scale. The optimized curves significantly differ in their final spread and variation.

normalized support of all c_j seen at the interpolation point c_i. The weight vectors $\mathbf{w}_i$ can be composed by inverting Φ and assembling the individual components from the three systems of equations.

Conflicting constraints of two source colors c_i and c_j can cause the matrix Φ to become near-singular. This causes components of the resulting $\mathbf{w}_i$ to become negative. Negative values create an inverted distribution of the constraint vectors $\mathbf{v}_i$ in the color space an produce undesired artifacts [15]. We avoid this problem by clustering support points that lie within the limit of perceivable distance (1.0 in the CIE Lab space). Additionally, we select basis functions that have a small spread to avoid large overlapping support between constraints. This keeps the matrix Φ from degenerating. In Section III-B, we discuss how to optimize basis functions and show that our normalized Shepard function combines optimal and robust interpolation results with a very small spread, in contrast to the generally utilized Gaussian function.

B. Choice of Basis Functions

The choice of an appropriate basis function both influences the global impact of color constraints and the stability of the interpolation [15]. Using RBF interpolation, we can make sure that the effect of overlapping support between the basis functions is minimized. However, it is still unclear which functions should be used for interpolation and how to select their parameters. To find out how to best interpolate colors given a sparse set of correspondences, we test the following three functions with different fall-off behavior over distance (see Figure 3).

normalized Shepard $\quad s_i(c_j) = (1 + ||c_i - c_j||)^{-\epsilon}$

Gaussian $\quad g_i(c_j) = e^{-(\epsilon||c_i - c_j||^2)}$

inverse quadric $\quad q_i(c_j) = \frac{1-\alpha}{1+(\epsilon||c_i - c_j||)^2} + \alpha$

The normalized version of the Shepard function has the quickest fall-off. It gets surpassed after a while by the Gaussian function that decreases exponentially. The constant value we add to the inverse quadric function ($\alpha = 0.01$ in our experiments) guarantees a minimum value in one of our

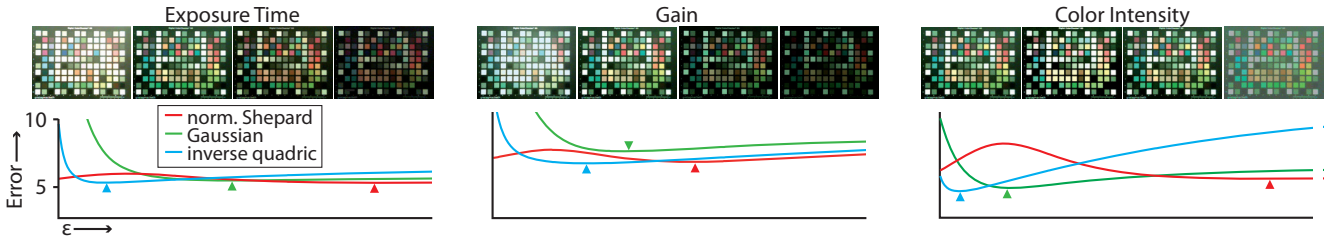

Figure 4. Average reconstruction error of the considered basis functions while interpolating 50% of the color checker fields under changing camera parameters. The error is shown as a function of the ϵ-values for the normalized Shepard (red), the Gaussian (green), and the inverse quadric (blue) basis functions. The plots show that the ϵ value significantly influences the result.

functions to distances at infinity.

Each of the functions is parameterized with a variable ϵ that influences the width of the function spread. This parameter allows to find the optimal reconstruction capability of each function by optimizing for ϵ.

To compare the performance of each parameterized function on reconstructing colors, we first create data sets by filming a color calibration checker (Digital ColorChecker SG in our experiments). During video capture, internal camera parameters (e.g. exposure or contrast) are changed individually so that the global color changes are part of the data.

In each data set, one frame with default camera settings is selected as the base frame. In this frame, 50% of the color fields are selected at random, and used as data points. The corresponding color fields in all other frames in the data set are used as references. The score of a function (with a specific ϵ-value) is now computed as the average reconstruction error of the other 50% (not referenced) colors in the base frame across all frames of the data set. Figure 4 shows plots from some of our tests. The error is shown for the normalized Shepard (red), the Gaussian (green), and the inverse quadric (blue) functions for an increasing ϵ-value. The plots are shown with differently scaled ϵ for each function, but with the same scaling across all tests.

For each of the three functions there is an optimal ϵ. This surprising result shows that there is a clear correlation between the spread of a basis function and its capability of reconstructing global color changes. Second, across all tests, the magnitude of the optimal ϵ is in a similar range. The ϵ-values we determined though our tests are $\epsilon = 3.7975$ ($\sigma = 0.5912$) for the normalized Shepard, $\epsilon = 0.0846$ ($\sigma = 0.0289$) for the Gaussian, and $\epsilon = 0.0690$ ($\sigma = 0.0289$) for the inverse quadric function, where σ-values are the standard deviations.

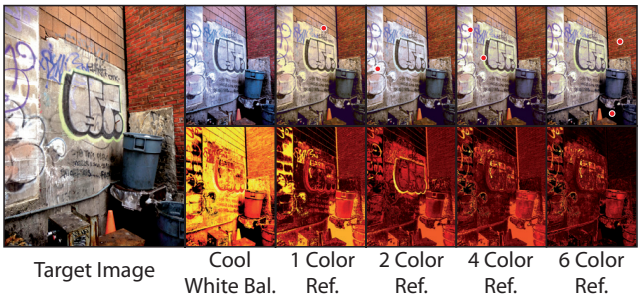

Figure 5. Example of the global impact of our color balancing using only a few correspondences. The target image on the left is reconstructed using a photograph of the same scene with a different white balance as source image. The top row shows the successive addition of references (red dots). The images in the bottom row visualize the difference in CIE Lab space between the balanced image and the original.

The shape of the optimized functions (and their deviation due to σ as dashed curves) are shown Figure 3 on the right. Our normalized version of the Shepard function clearly shows the smallest spread of all three functions, rendering it the most stable function for RBF interpolation. Note, that the Gaussian function has significantly wider spread and also a much higher variance, making it a suboptimal choice for global color balancing. The impact on the number of references using our normalized Shepard function with optimal ϵ for global color balancing is demonstrated in Figure 5. With only a few correspondences, the error in the CIE Lab space is reduced considerably.

IV. Temporal Color Tracking

A typical scenario where the permanent update of the color balance improves the visual quality is augmented reality, where a synthetic rendering should be embedded in a seamless manner into a video stream. In order to tackle this problem, we propose to use known color patches in the scene (See Figure 1). If these colors can be robustly tracked over time, they can be used as color references for their supposed values. These references then automatically balance rendered images to the video footage, adapting the renderings as the color of the video frames change due to camera adjustments.

We base our color tracking on the observation that desired color changes in the video stream are global. These changes are due to the camera automatically adjusting internal parameters such as exposure or gain. The color patches in the scene that receive these color changes will be used as constraints for the balancing of their known unbalanced values. However, due to occlusions or specular reflections, some of the tracked color patches will be corrupt. These color distortions (e.g. occlusions, reflections) happen locally. This fact can be utilized to remove them as outliers.

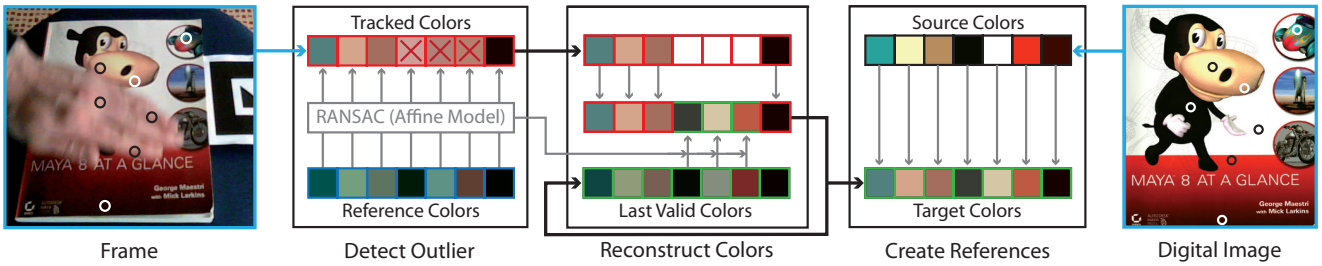

Figure 6. Overview of our stable color tracking pipeline. First, color values are extracted from the video frame and corrupt ones are rejected using RANSAC and temporal coherence. Then, missing colors are added using information from the previous frames. Finally, the stable color list is combined with colors from a digital image to create the transfer constraints that can be used for balanced augmentation of rendered objects.

Figure 6 gives an overview over our robust color tracking pipeline. For each frame, the following steps are performed:

1 Extract Colors. The first step is to extract the colors d_i from the video frame. The position tracking provides the positions in the image where the target colors are found. In our implementation we use ARToolKit marker tracking [26], but any other position tracking solution can be used as well. To reduce per-pixel noise we perform a neighborhood averaging over a small window (usually 7 by 7).

2 Detect Outlier. To detect corrupted colors we can fit a global model between the extracted d_i and a set of reference colors. These reference colors r_i can be taken from the digital image of the marker or extracted from the first frames of the video in a pre-processing step. In order to keep the number of false positives low a conservative model should be fitted between the c_i and r_i. We chose the affine transformation model $c_i = \mathbf{A}\, r_i + \mathbf{t}$ [16], which can be computed analytically and avoids over fitting to corrupted colors.

Outlier in the scene colors d_i can now be detected by fitting the affine model using the random sample consensus algorithm (RANSAC) [27]. This first step allows removing most of the corrupted colors. However, false positives may slip though this outlier detection. In order to further increase the robustness of the tracking, we separate the remaining inliers into two groups. Trusted inliers are colors that have not been detected as outliers for more than κ_t frames. These colors are used as valid target colors. Inlier colors that have been detected as outlier in one of the past κ_t frames may be false positives. These colors are also treated as outlier. In our experiments we found that false positives rarely appear for more than two frames. Therefore, in our experiments, we mostly used $\kappa_t = 3$.

3 Reconstruct Colors. At this point, corrupted colors have been removed from the list of d_i. Additionally, the risk of false positives has been decreased by taking temporal consistency into account. Now, in order to maximize the amount of colors available for color balancing, we can use the last valid colors of each tracked point from the previous frames. These colors are stored when a tracked value is regarded as trusted inlier (being not an outlier for more than κ_t frames). Colors values, that are suddenly corrupt due to occlusions or specular reflections can then be replaced by an updated version of their last valid value. The update of these last valid colors is performed by applying the global affine transformation performed by all trusted inlier colors since the last frame. In our implementation, we perform this color reconstruction only for outlier that have been inliers for more than κ_c frames. This removes colors that were only detected as inliers for a short period and increases the robustness. We call these colors comeback points. In our implementation we have set $\kappa_c = 10$.

4 Create References. The final color references (c_i, d_i) for the balancing of rendered objects can now be created. For all d_i, that are either trusted inlier or a comeback points, the corresponding color c_i can be extracted from the digital image of the marker. The constraints (c_i, d_i) now describe the color transfer function from rendered footage to the current video frame color space.

With our fast global color balancing for sparse color correspondences we are able to adapt the colors to fit to a target image. With our robust color tracking, we can extend this into the temporal domain. Using a known marker in the scene (e.g. the cover of a book) we can balance newly rendered footage to augment the video and increase the realism of augmented reality applications.

V. Implementation

Our color balancing algorithm can be implemented by simply extending any given fragment shader. The reference colors c_i and weights $\mathbf{w}_i$ are transferred to the GPU as texture data. The matrix inversion is calculated beforehand on the CPU as it only needs to be performed once. The GPU fragment shader algorithm is described in Algorithm 1.

First, the original color e of the pixel is determined (line 1). The pixel color is the result of the given shading algorithm. This color is then transformed to CIE Lab space (line 2). Then, the variables $\mathbf{v}_e$ and s_e are initialized (line 3 & 4). They are used to compute intermediate results of the transformation vector computation. Now, a loop is performed n-times, which is the number of color correspondences. In each loop, first, the color c_i and weight vector $\mathbf{w}_i$ are acquired through texture lookup (line 6). Then, the function ϕ is applied using the current color e_{Lab} and the

support color c_i (line 7). This gives the function value r, which is used to add the current weight $\mathbf{w}_i$ to $\mathbf{v}_e$ (line 8). Additionally, the sum of all the function values r is stored in s_e (line 9), so that after the loop the transformation vector $\mathbf{v}_e$ can be normalized and used to translate the original pixel color in CIE Lab space (line 11). The final color is then transformed back into RGB space and returned to the frame buffer (line 12).

We have implemented the color balancing algorithm on an Intel Core i7 3.2 GHz with Nvidia GeForce GTX 460. The run-time in ms is shown in Figure 7 for an increasing number of references. The solver for the equation system (blue graph) is the bottleneck, as it involves a matrix inversion in the size of the number of references. The RBF interpolation (blue graph) however performs with around 0.05 ms almost independently of the number of references.

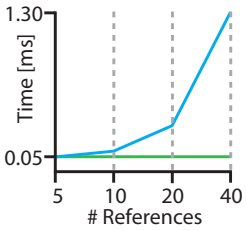

Figure 7. The performance of our implementation. The green function shows the run-time for the color balancing on the GPU, the blue function the time to solve the RBF equation system on the CPU.

VI. Results and Applications

In this section we present various color balancing results for image editing and augmented reality applications, and compare them to realated works. Please see also the accompanying video where we show several examples for both color balancing and augmented reality.

A. Interactive Color Transfer and Correction

Given a source image and a target image with a desired look, the user can interactively define a color mapping by clicking correspondences. The result is instantly visible, and the user can drag the correspondence points around to see the effect. A number of examples are shown in Figure 8 that demonstrate the flexibility of our approach. Due to the global nature of the vector field color transformation, it is very easy to balance an image to exactly match an example photograph. Even completely changing the color composition of an image can be achieved robustly using the same kind of input.

Figure 9 shows a series comparisons of our color balancing method with other color balancing approaches. Since our method is designed and optimized for global color balancing, it is able to transfer the colors from the target image to the source image with only a sparse set of references. User edits in Photoshop changing indirect parameters (hue, saturation, etc.) can be unintuitive and time consuming, and it is difficult to achieve exact results. Achieving the correct balance between the yellow and red colors in the bottom example are difficult with only indirect manipulation. Both the Photoshop Color Match function and the approach of Reinhard et al. [3] perform very similarly. Both methods automatically achieve a global balancing of the image but cannot recreate the exact shade of the Taj Mahal. Using the sparse input used for our color balancing, the approach of Li et al [15] struggles, as their method is specifically designed for local edits. In areas with strong gradients (e.g. the dome of the Taj Mahal) their approach produces artifacts. Linear regression is, due to its limited flexibility, not able to satisfy the constraints.

Algorithm 1 Fragment Shader Color Balancing

1: $e \leftarrow$ ***pixelColor***$()$
2: $e_{Lab} \leftarrow$ ***rbg2lab***(e)
3: $\mathbf{v}_e \leftarrow (0,0,0)$
4: $s_e \leftarrow 0$
5: **for** $i = 1 \rightarrow n$ **do**
6: $\quad [c_i, \mathbf{w}_i] \leftarrow$ ***dataTexture***(n/i)
7: $\quad r \leftarrow$ ***phi***(c_i, e_{Lab})
8: $\quad \mathbf{v}_e \leftarrow \mathbf{v}_e + r\,\mathbf{w}_i$
9: $\quad s_e \leftarrow s_e + r$
10: **end for**
11: $e_{Lab} \leftarrow e_{Lab} + \mathbf{v}_e / s_e$
12: **return** ***lab2rgb***(e_{Lab})

B. Augmented Reality Color Balancing

While the color transfer tool only utilizes the flexible vector space color transfer, applications in augmented reality become possible when combining it with our robust color tracking over time. A known marker in the scene, like the cover of a book, can be tracked and the colors referenced with a digital copy of the image. Figure 10 shows two examples of color balancing for augmented reality.

The top example shows a series of comparisons between our color correction and linear regression. It can be observed that our approach is more accurate for colors that are available in the video stream and more robust for colors that are not tracked. The bottom row shows an example where an animated image is placed in a book. Due to the color tracking of the real image on the same page we are able to more realistically embed the animated figure into the video. Note that the adjustment of the colors in the flat shaded examples would only work for one frame as the white balance and global light change over time.

Generally, in augmented reality applications, it is crucial to perform proper image balancing for convincing augmentation. A viewer is especially sensitive to colors that already appear in the video stream, and will notice when similar colored renderings are balanced differently. The results in Figure 10 and in the accompanying video demonstrate, that our color tracking and balancing is able to achieve the desired accuracy, while linear regression [2] struggles.

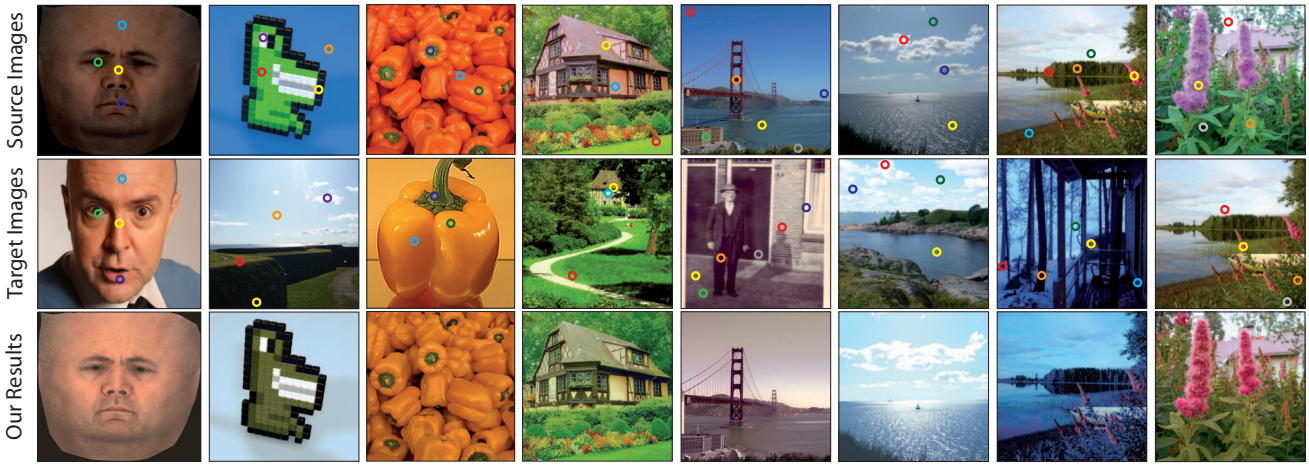

Figure 8. Several examples of our sparse color balancing. The colored circles in images mark the color correspondences.

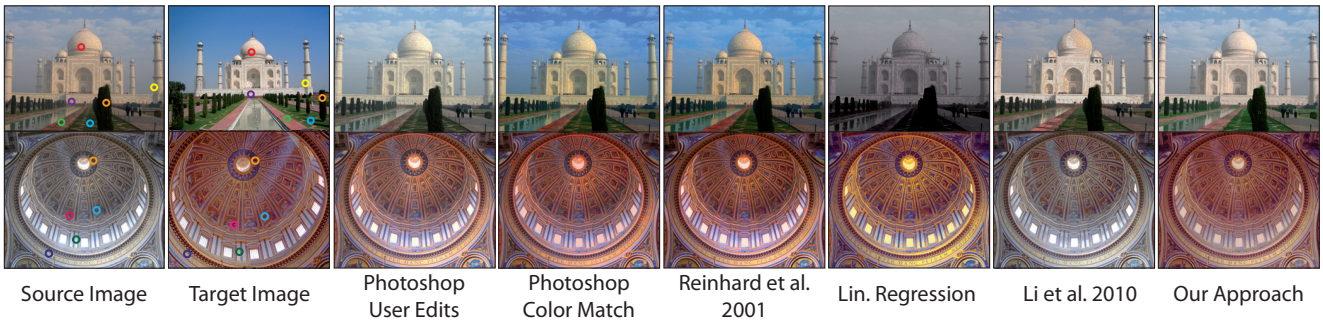

Figure 9. Comparison of our approach with other methods for image balancing using sparse correspondences.

VII. Discussion

In this paper, we have presented an approach for interactive image-based color balancing using only a sparse set of correspondences, as well as an extension for temporally-consistent rendering for augmented reality applications. Through our proposed global optimization of interpolation functions, we provide a tool to optimize function parameters to mimic color changes from example images and optimally adjust colors accordingly.

The proposed color correction and tracking algorithms have some limitations that direct us to areas of future work. The color correction is not able to reconstruct missing information. Over- or underexposed images can flatten color gradients to the point where only a few colors remain. Our color transfer method will not smooth spatially, and therefore is not able to increase the amount of colors present in a gradient. Coupling the color transfer algorithm to camera exposure setting is an area of future work that could address this issue.

The color tracking algorithm relies on flat color areas in the image to work well. This requirement is orthogonal to many tracking algorithms that use features like corners or edges. However, our algorithm will reject such colors instead of producing false results. In our current implementation we employ equidistant sampling of points on the marker surface where the colors near features get removed by our tracking. Improving this sampling using information such as edges or gradients in the marker image would be an interesting topic to explore.

References

[1] G. Klein and D. W. Murray, "Simulating Low-Cost Cameras for Augmented Reality Compositing," *IEEE Transactions on Vision and Computer Graphics*, vol. 16, no. 3, pp. 369–380, 2010.

[2] M. Knecht, C. Traxler, W. Purgathofer, and M. Wimmer, "Adaptive Camera-Based Color Mapping for Mixed-Reality Applications," in *ISMAR*, 2011, pp. 165–168.

[3] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color Transfer Between Images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, 2001.

[4] X. Xiao and L. Ma, "Color Transfer in Correlated Color Space," in *VRCIA*, 2006, pp. 305–309.

[5] F. Pitié and A. C. Kokaram, "The Linear Monge-Kantorovitch Linear Colour Mapping for Example-Based Colour Transfer." *Visual Media Production*, 2007.

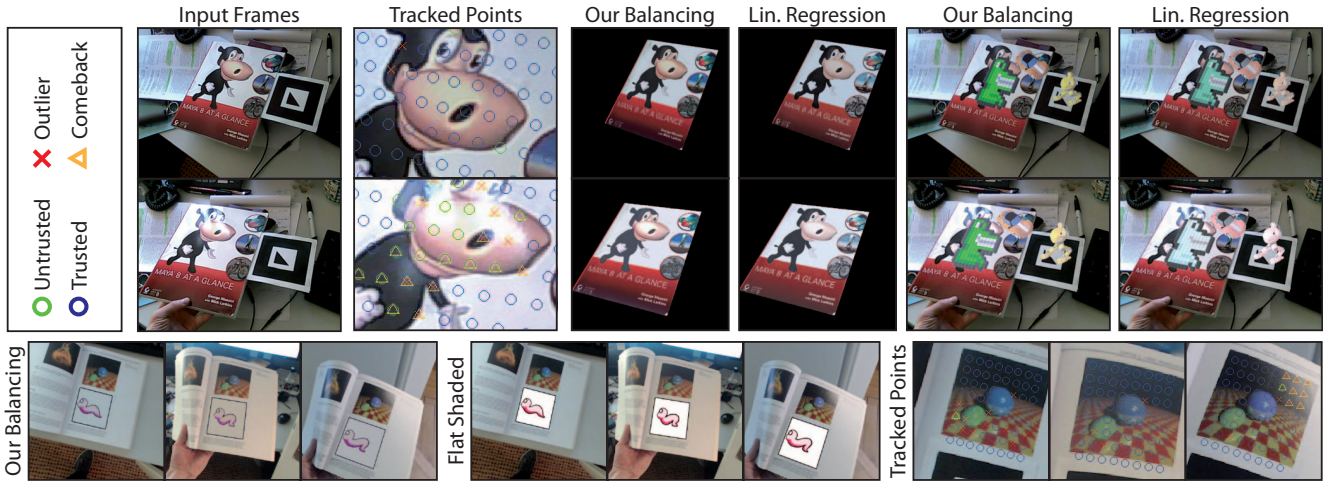

Figure 10. Two examples of our color tracking and balancing in augmented reality applications. Outliers (red x) arising due to specular reflection are detected and supported with comeback colors (yellow triangle). The top row shows a series of comparisons between our color correction and linear regression. Our approach is both more accurate in recreating the exact image colors as well as produce more plausible color balancing for unknown colors (i.e. green frog). The bottom row shows an example where an animated image is placed in a book and balanced by tracking the colors of a real image.

[6] S. Kagarlitsky, Y. Moses, and Y. Hel-Or, "Piecewise-Consistent Color Mappings of Images Acquired Under Various Conditions," in *ICCV*, 2009, pp. 2311–2318.

[7] A. Neumann and L. Neumann, "Color Style Transfer Techniques using Hue, Lightness and Saturation Histogram Matching," in *Computational Aesthetics*, 2005, pp. 111–122.

[8] X. Xiao and L. Ma, "Gradient-Preserving Color Transfer," *Computer Graphics Forum*, vol. 28, no. 7, pp. 1879–1886, 2009.

[9] A. Abadpour and S. Kasaei, "A Fast and Efficient Fuzzy Color Transfer Method," in *Signal Processing and Information Technology*, 2004, pp. 491 – 494.

[10] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive Local Adjustment of Tonal Values," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 646–653, 2006.

[11] Y.-W. Tai, J. Jia, and C.-K. Tang, "Local Color Transfer via Probabilistic Segmentation by Expectation-Maximization," in *CVPR (1)*, 2005, pp. 747–754.

[12] X. An and F. Pellacini, "User-Controllable Color Transfer," *Computer Graphics Forum*, vol. 29, no. 2, pp. 263–271, 2010.

[13] X. An and F. Pelacini, "AppProp: All-Pairs Appearance-Space Edit Propagation," *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[14] Z. Farbman, R. Fattal, and D. Lischinski, "Diffusion Maps for Edge-Aware Image Editing," *ACM Transactions on Graphics*, vol. 29, no. 6, p. 145, 2010.

[15] Y. Li, T. Ju, and S.-M. Hu, "Instant Propagation of Sparse Edits on Images and Videos," *Computer Graphics Forum*, vol. 29, no. 7, pp. 2049–2054, 2010.

[16] J. Adams, K. Parulski, and K. Spaulding, "Color Processing in Digital Cameras," *IEEE Micro*, vol. 18, no. 6.

[17] V. Agarwal, B. R. Abidi, A. Koschan, and M. A. Abidi, "An Overview of Color Constancy Algorithms," *Journal of Pattern Recognition Research*, pp. 42–54, 2006.

[18] N. Cohen, "A Color Balancing Algorithm for Cameras," *EE368 Digital Image Processing*, 2011.

[19] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister, "Image Restoration using Online Photo Collections," in *ICCV*, 2009, pp. 2217–2224.

[20] S. B. Kang, A. Kapoor, and D. Lischinski, "Personalization of Image Enhancement," in *CVPR*, 2010, pp. 1799–1806.

[21] H. Siddiqui and C. A. Bouman, "Hierarchical Color Correction for Camera Cell Phone Images," *IEEE Transactions on Image Processing*, vol. 17, no. 11, pp. 2138–2155, 2008.

[22] B. Wang, Y. Yu, T.-T. Wong, C. Chen, and Y.-Q. Xu, "Data-Driven Image Color Theme Enhancement," *ACM Transactions on Graphics*, vol. 29, no. 6, p. 146, 2010.

[23] S. Yang, Y.-A. Kim, C. Kang, and B.-U. Lee, "Color Compensation Using Nonlinear Luminance-RGB Component Curve of a Camera," in *ISVC (2)*, 2011, pp. 617–626.

[24] B. Wang, Y. Yu, and Y.-Q. Xu, "Example-Based Image Color and Tone Style Enhancement," *ACM Transactions on Graphics*, vol. 30, no. 4, p. 64, 2011.

[25] H. T. Lin, S. J. Kim, S. Susstrunk, and M. S. Brown, "Revisiting Radiometric Calibration for Color Computer Vision," *ICCV*, 2011.

[26] ARToolKit, "Software Library for Building Augmented Reality Applications." 2012, [Online; accessed 29-June-2012].

[27] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.