DOI: 10.1111/j.1467-8659.2009.01595.x

EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

User-Controllable Color Transfer

Xiaobo An and Fabio Pellacini

Computer Science, Dartmouth College

Abstract

This paper presents an image editing framework where users use reference images to indicate desired color edits. In our approach, users specify pairs of strokes to indicate corresponding regions in both the original and the reference image that should have the same color "style". Within each stroke pair, a nonlinear constrained parametric transfer model is used to transfer the reference colors to the original. We estimate the model parameters by matching color distributions, under constraints that ensure no visual artifacts are present in the transfer result. To perform transfer on the whole image, we employ optimization methods to propagate the model parameters defined at each stroke location to spatially-close regions of similar appearance. This stroke-based formulation requires minimal user effort while retaining the high degree of user control necessary to allow artistic interpretations. We demonstrate our approach by performing color transfer on a number of image pairs varying in content and style, and show that our algorithm outperforms state-of-the-art color transfer methods on both user-controllability and visual qualities of the transfer results.

1. Introduction

The need for image editing arises in a variety of applications, from post-processing in movie production to everyday photo quality enhancement. While many commercial software packages such as Adobe Photoshop [Ado07] and Lightroom [Ado08] provide a number of tools for fast image editing, using these packages to accomplish complex local edits remains very challenging since significant time is needed for precise selections [LFUS06].

Recently, a number of edit propagation methods [LLW04, LFUS06, AP08] have been proposed to help reduce the human labor involved in this tedious editing process. With these methods, users specify editing parameters (e.g. luminance scale or hue offset) at a set of stroke locations in the image and the algorithm automatically propagates these parameters to regions of similar appearance. These stroke-based formulations have proven to be intuitive for users while requiring minimal user effort. While these methods are very effective for simple editing operations, such as brightening up an under-exposed area in an image, performing more sophisticated edits remains cumbersome since a great amount of trial-and-error is often required to determine the editing parameters that achieve the desired "look". For example, the edits shown in Fig. 1 require complex combinations of luminance, hue and saturation changes that differ in various regions of the image. Finding the right parameters to achieve these edits is especially cumbersome for novice users, that often have trouble even deciding what "look" they want to achieve. In these cases, *using other images as color references helps in determining the final desired look*. Current edit propagation frameworks cannot take advantage of these color references since they rely on users to specify editing parameters.

Automatic color transfer algorithms have been proposed that alter the colors of the original image to globally match the color distribution in the reference [RAGS01, GD05a, PK07, PKD07]. Their usefulness is underlined by their adoption in commercial software package for professionals, e.g. the "match color" tool in Photoshop. In most cases however, strong artifacts are present in the edited image even when the content of the original and reference is similar, as shown in Fig. 1. To address this problem, region-by-region transfer can be performed using automatic image segmentation algorithms [TJT05, AK07]. These methods are better than global ones, but suffer from well-known issues found in all segmentation algorithms, as we will discuss later in Sec. 2. More importantly though, all automatic methods do not allow users to control the look of the final edits. *We believe that the definition of "color style" is highly subjective and that an ideal color transfer algorithm should retain a high*

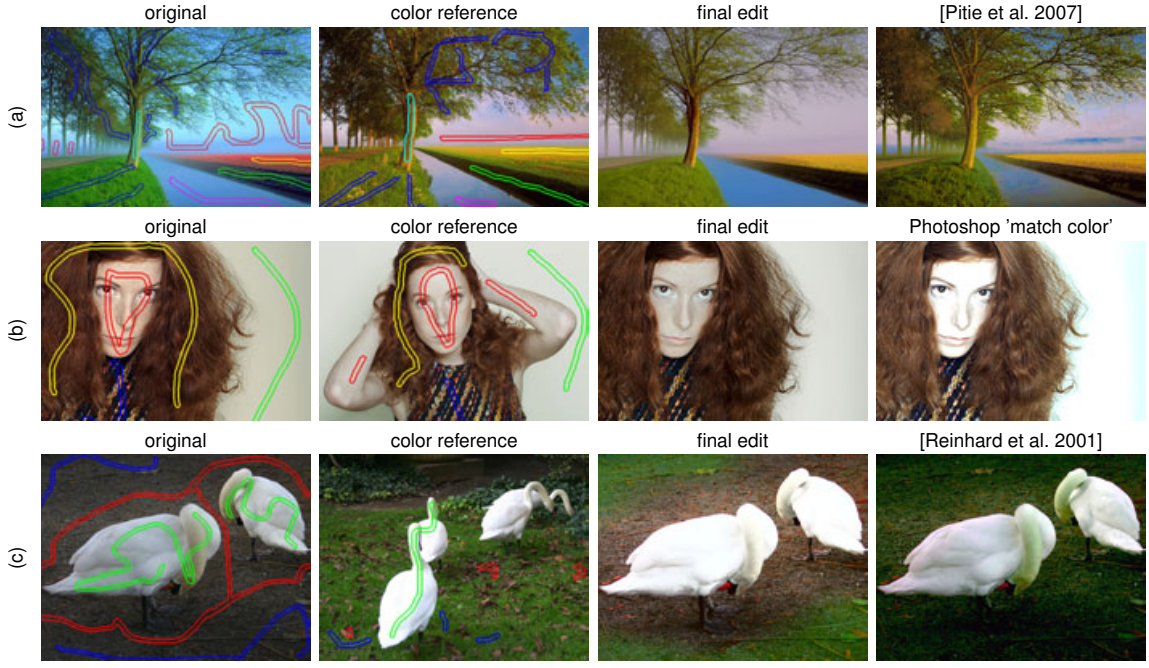

Figure 1: *To edit an image's colors, users specify pairs of corresponding strokes on both the original image and a color reference, to indicate regions that should have the same color "style". From these, our algorithm derives a per-pixel function that transfers the color of the reference to the original, allowing users to quickly perform complex local color adjustments. Compared to prior methods, our algorithm ensures that no visual artifacts are present in the final edits.*

degree of user control to allow for different artistic interpretations.

Recently, user guided color transfer methods have been proposed. In this paper, we present a new framework to perform tonal and color adjustments in images by using color references in a user-controllable manner that requires minimal user effort. Our framework integrates color transfer and edit propagation algorithms maintaining the benefits of both. In our approach, users specify pairs of strokes to indicate corresponding regions in the original and reference images that should have the same color "style". For each pair of strokes, our algorithm computes a transfer function that, when applied to the colors of the stroked pixels in the original, ensures that their distribution matches the one in the reference stroke. We choose to represent the transfer function as a nonlinear constrained parametric model. The nonlinearity of the model allows close matching of color distributions after transfer, while the constraints imposed on the model are designed to minimize visual artifacts in the transferred results. To apply the transfer function to the whole image, we propagate the model parameters, defined at each stroke location, to spatially-closeby regions of similar appearance using edit propagation methods [LFUS06, AP08, XLJ*09]. Since we designed our transfer model in a way that interpolation of our model parameters is well defined, applying edit propagation techniques ensures that the final edits are smooth across the whole image and no visual artifacts are present in the final results. In summary, this paper makes the following contributions:

- we propose a color transfer framework that uses a sketching interface, requiring minimal user effort while maintaining controllability
- we define a nonlinear constrained parametric transfer model that can precisely transfer color styles within strokes while minimizing visual artifacts in the results
- we extend current edit propagation frameworks by allowing the use of references in the editing process, which greatly improves their usability
- we demonstrate our approach on a number of image pairs of varying content and style and show that our algorithm outperforms previous methods

2. Prior Work

Automatic Color Transfer. Many automatic transfer algorithms have been proposed that attempt to globally match the color distribution of the original to the reference. We show some results using such algorithms in Fig. 1. Reinhard et al. model color distributions as gaussians and defines transfer as per-axis scales and offsets in $l\alpha\beta$ color space [RAGS01]. Xiao et al. improve upon the former by

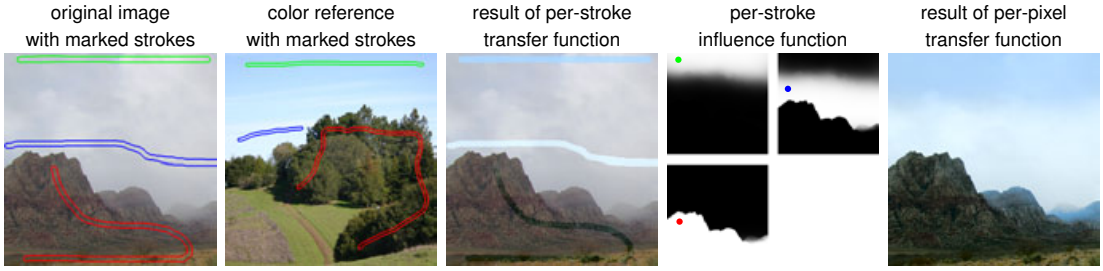

Figure 2: *Overview of our algorithm. See Sec. 3 for details.*

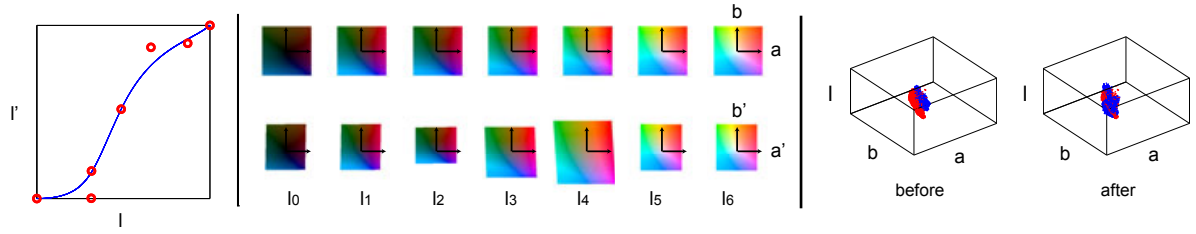

Figure 3: *An illustration of our nonlinear constrained parametric transfer model for a stroke pair.* Left*: mapping between the original luminance l and the final luminance l'.* Center*: (a,b) plane at l_0 to l_6 before and after transfer, showing how the original values area mapped into (a',b') planes. All colors correspond to original (l,a,b).* Right*: color distributions of reference (red) and original (blue) stroke before and after transfer. The model parameters and color distributions correspond to the stroke pair marked in red in Fig. 2.*

finding the best color axes to perform transfer with [XM06]. Pitie et al. extends these approaches by modeling image colors as gaussian mixtures and linearly transforming the mixtures [PK07]. The same authors later present a nonlinear model that perfectly matches color distributions with a modified histogram matching procedure [PKD07]. Even with this model, artifacts are generated in the results when original and reference distributions are dissimilar, which is the case for the majority of image pairs [PKD07]. Region-by-region methods have been introduced to attempt to address this well-known issue, relying on automatic image segmentation [TJT05, AK07]. These algorithms mostly vary in the image segmentation method adopted, while for each region a linear transformation is derived from [RAGS01]. While this improves the results, relying on automatic segmentation is problematic since these algorithms are known to fail on many cases. [WAM02, ICOL05] use references to colorize grayscale images. However, these algorithms do not generalize to transfer between color images. Furthermore, and most importantly, automatic methods do not allow for artistic interpretation which we believe is fundamentally important for any editing operation.

User-Driven Color Transfer. User-driven color transfer methods allow users to directly mark image correspondences by using either gradient lines [GD05b], swatches [MV07] or strokes [LWX07, WHCO08]. Within each corresponding pair, a linear transfer model based on [RAGS01] is used to match color distributions. These methods, being the closest to our own, suffer from two main problems. First, the gaussian model cannot represent color distributions well even within strokes. Second, when extrapolating the transfer parameters to the whole image, these linear models generate artifacts such as out-of-gamut colors. We propose a nonlinear constrained transfer model that can closely match color distributions while minimizing visual artifacts in the results. Sec. 3.4 shows an in-depth comparison of transfer models.

To apply transfer to the whole image, [MV07, WHCO08] use radial basis functions for interpolating model parameters. As stated in [WHCO08], this interpolation leads to artifacts in the results which they mitigate by blurring the paramaters by using an edge-stopping weighted-least squares optimization [LFUS06]. Luan et al. uses a more sophisticated gradient-based technique that directly propagates colors, rather than model parameters [LWX07]. This in turn causes artifacts in textured regions and loss of contrast. In our framework, interpolation of model parameters is well defined and thus edit propagation methods can be used to smoothly propagate parameters to regions of similar appearance while ensuring no visual artifacts are present in the results.

Edit propagation. Stroke-based formulations were first introduced for colorization and local tonal adjustments in [LLW04, LFUS06]. In these algorithms, users specify edit-

ing parameters at a couple of sparse locations in the data by drawing strokes and the algorithm automatically propagates the editing parameters to the rest of the image. Recently, An et al. have generalized the previous frameworks for editing complex spatially varying appearance datasets, including both images and measured materials [AP08]. The fundamental principle in these edit propagation algorithms is that spatially-close regions of similar appearance should receive similar edits. While these methods are intuitive to use for simple operations such as exposure adjustments and saturation changes, using them to achieve complex edits remains impractical. For instance, to achieve the edits shown in Fig. 1, different hue shifts, saturation changes and contrast control are needed for each local region of the original image. Finding the right combination of these adjustments often requires tedious trial-and-error. In our framework, users can indicate desired editing effects by using references, rather than explicitly specifying the editing parameters. With the model parameters estimated at each stroke location, edit propagation methods can be used to propagate edits to regions of similar appearance.

3. User-Controllable Color Transfer

In our framework, users specify pairs of strokes on regions of the reference and original image that should have the same color "style". While it is hard to formally define style, we argue the perception of color style is directly associated with color tones, saturations, contrast, etc, which are primarily determined by color distributions. Therefore, we transfer color style between strokes by matching color distributions. Note that while matching distributions globally often leads to visual artifacts in the results, as shown in Fig. 1, this is rarely a problem within strokes since such distributions are significantly simpler.

Our transfer algorithm proceeds in three steps, shown in Fig. 2. Given a set of pairs of corresponding strokes on both the reference and the original, we compute a transfer function between each pair by matching their color distributions using a nonlinear constrained parametric model. An edit propagation method is then used to compute the influence of each stroke on regions of similar appearance in the original. Finally, for each pixel in the original, we compute model parameters by interpolating the parameters defined at the stroke locations using their influence functions. Applying this per-pixel transfer function computes the final result.

Note that unlike previous methods such as [ICOL05, LWX07], we do not compute the final transfer result by propagating edited colors at the stroke locations, but rather by propagating the transfer functions to the whole image. This is because direct propagation of colors can lead to artifacts such as loss of contrast and texture details. On the other hand, applying a smoothly-interpolated function to the original colors maintains the rich details found in the original image, as shown in [LFUS06, AP08].

3.1. Parametric Transfer Model

In our framework, the transfer function is represented as a nonlinear constrained parametric model that can closely match color distributions per-stroke-pair, while ensuring that no visual artifacts are present in the results. Our model works in any color space with separate luminance and chromaticity axes, such as CIE Lab or oRGB [BBS09], and can be written as a function $F_\rho : c \rightarrow c'$, that transforms a color triplet $c = (l,a,b)$ to $c' = (l',a',b')$ given the model parameters ρ.

Our model is illustrated in Fig. 3. Intuitively, we model the transfer between l and l' as a piecewise cubic Bezier spline in the plane (l,l'). We choose cubic splines to resemble the behavior of the "curve" tool in Photoshop or Lightroom, and since they are smooth and have flexible shapes. In the chromaticity domain, we choose to define the transfer as an affine transformation cubicly interpolated along l. We choose affine transformations of chromaticities since they capture most color operations well, i.e. adjustments to hue, saturation, and color warmth. We allow the affine transformations to vary along luminance to model more complex operations where colors within different luminance ranges are transformed differently. In all of our tests, we found that two segments of the cubic spline are sufficient to capture the transfer. For the reminder of this section, we will formally introduce our model on one segment for simplicity of explanation.

We define our transfer function by specifying the control points $\rho = (l_i, l_i', M_i, t_i)$, where (l_i, l_i') are the original and transferred luminance of the control point and (M_i, t_i) are the linear and translation components of the affine transformation for this control point. Within the segment, the curve values can be written as $\bar{v}(u) = \sum_{i=0}^{3} b_i(u) v_i$ where $\bar{v}(u)$ is one of $\bar{l}(u), \bar{l'}(u), \bar{M}(u), \bar{t}(u)$; u is the parametric variable of the Bezier and $b_i(u)$ are the basis polynomials. Note that computing $\bar{M}$ by linearly weighting the matrix coefficients results in strong distortions of the transformation space. In our implementation, we employ the matrix interpolation proposed in [SD92] which will be further discussed in Sec. 4. Given this representation, the transfer $c = F_\rho(c')$ can be written as

$$l' = \bar{l}(u) \quad [a',b']^T = \bar{M}(u) \cdot [a,b]^T + \bar{t} \quad \text{where } u = \bar{l}^{-1}(l)$$

Note that to determine u we need to invert a cubic equation $\bar{l}(u)$. In practice, we simply use lookup tables to quickly evaluate it.

We impose several constraints on the parameters of this general model to minimize visual artifacts in the transferred values. In particular, we impose constraints on the luminance control points to ensure no clipping of the final values or flipping of luminance gradients. These can be simply avoided by setting $F(0,a,b) = (0,a',b')$ and $F(1,a,b) = (1,a',b')$ and by enforcing the spline to be monotonically increasing, i.e. $l'(\hat{l}) > l'(l)$ for $\hat{l} > l$. On the affine components, we impose the constraint that the control point at $l = 0$ is the same

as the next and that the one at $l = 1$ is the same as the previous. This accounts for the fact that in Lab, chromaticity values of nearly black or white points are consistently noisy and thus unreliable. Finally, we further ensure C^1 continuity of the spline for consecutive segments by appropriately setting shared control points. Note that we do not impose constraints to avoid out-of-gamut chromaticies since no cases were found in our tests, while theoretically this could arise.

3.2. Per-Stroke Parameter Estimation

Within each stroke pair, our goal is to solve for a set of model parameters that can maximally match the color distributions of the reference and original, under constraints we desire to impose on the model. We measure the differences of two color distributions by computing the sum of the L^2 differences between the three dimensional cumulative density functions of both distributions. We can formally express the parameter estimation problem as:

$$\arg\min_{\rho} E(\rho) = \int_c |cdf_r(c) - cdf_t(c)|^2 dc$$

subject to the constraints we impose on the model parameters. Here cdf_r and cdf_t are the three dimensional cumulative density functions of the reference and color distributions after transfer in the Lab space. We choose to measure the differences of three dimensional cdfs since this was proven to always outperform matching the distribution of the individual axes separately [PKD07]. Fig. 3 illustrates how accurately our model transforms the color distribution of the red stroke in the original to the blue stroke in the reference in Fig. 2. We found this accuracy to be common on most stroke pairs. Implementation details are discussed in Sec. 4.

3.3. Per-pixel Parameter Interpolation

Given the model parameters for each stroke, we compute the parameters for all pixels in the original image by adopting recent edit propagation algorithms [LFUS06, AP08] where stroke parameters are smoothly propagated to closeby-regions of similar appearance while respecting image edges and complex patterns. This not only ensures that no visual artifacts are present in the transfer result, but also allows users to locally control the transfer by changing stroke shape and position. The only requirement needed to use edit propagation is that linear combinations of model parameters generate well-defined transfer functions, which is the case for our constrained parameterization, and a specific strength of our approach in comparison with other color transfer models (as we will analyze in Sec. 3.4). In this section we quickly review edit propagation and discuss our implementation choices, referring the reader to the original papers for a detailed analysis.

Edit propagation computes the values of per-pixel editing parameters e from the one defined at stroke locations g by minimizing an energy function that ensures that similar edits are applied to regions of similar appearance. Two different flavors have been proposed in the literature corresponding to different propagation semantics. [LFUS06] propagates editing parameters in contiguous image regions separated by strong edges by minimizing the function

$$\arg\min_e \sum_i w_i(e_i - g_i)^2 + \lambda \sum_i \sum_{j \in N_i} z_{ij}(e_i - e_j)^2$$

where z_{ij} is the affinity between pixel i and j, w_i is 1 if the pixel is within a stroke (0 otherwise) and N_i is the four neighbors of pixel i in the image domain. [HMP*08] shows how this optimization can better reproduce edges if the affinity matrix is substituted by the matting laplacian [LLW08, LRAL08]. [AP08] propose a different formulation where edits are propagated to all image pixels, even when not found in contiguous regions, by minimizing a function defined over all image pairs

$$\arg\min_e \sum_i \sum_j w_j z_{ij}(e_i - g_j)^2 + \lambda \sum_i \sum_j z_{ij}(e_i - e_j)^2$$

In our framework, the user can choose which propagation semantic is most suitable for her edits. In our experience, we found that when editing natural scenes, exhibiting complex patterns and soft transitions, the all-pair method of [AP08] is the most convenient since it requires very few roughly placed strokes. On the other hand, when editing portraits of people, we use the edge-stopping method of [HMP*08], to gain more precise control on subtle edits.

Regardless of the propagation method, [LFUS06] has shown that these optimization formulations implicitly define influence functions for each stroke such that per-pixel edit parameters can be computed as weighted averages of the parameters defined at stroke locations. We adopt this formulation and compute, at each pixel j, a set of interpolated values for the control points for our model as $p_j = \sum_k f_j^k p^k$ where p is one of l_i, l_i', M_i, t_i and f_j^k is the value of the influence functions f_k for stroke k at pixel j. Again note that when interpolating M we adopt [SD92].

3.4. Comparisons of Transfer Models

Reinhard et al. proposed a color transfer that matches color statistics per channel, assuming that the underlying distributions are gaussians, by computing for each channel $c' = \mu_r + \sigma_r/\sigma_t(c - \mu_t)$, where μ_r, μ_t, σ_r, σ_t are the mean and standard deviation of the original and reference distributions respectively [RAGS01]. This model is used within stroke pairs in [LWX07, WHCO08]. More complex linear transformations are used as the transfer model in [XM06, GD05b]. The problem with all linear models is that the underlying gaussian assumptions rarely hold in natural images, e.g. see Fig. 3. This in turn causes severe artifacts in the final edits. To show some such artifacts, we swapped our nonlinear constrained parametric model with [RAGS01] and generated Fig. 4 using the same propagation framework for each. Compared to our model, we found three major recurring issues with linear models. First, since they are unconstrained,

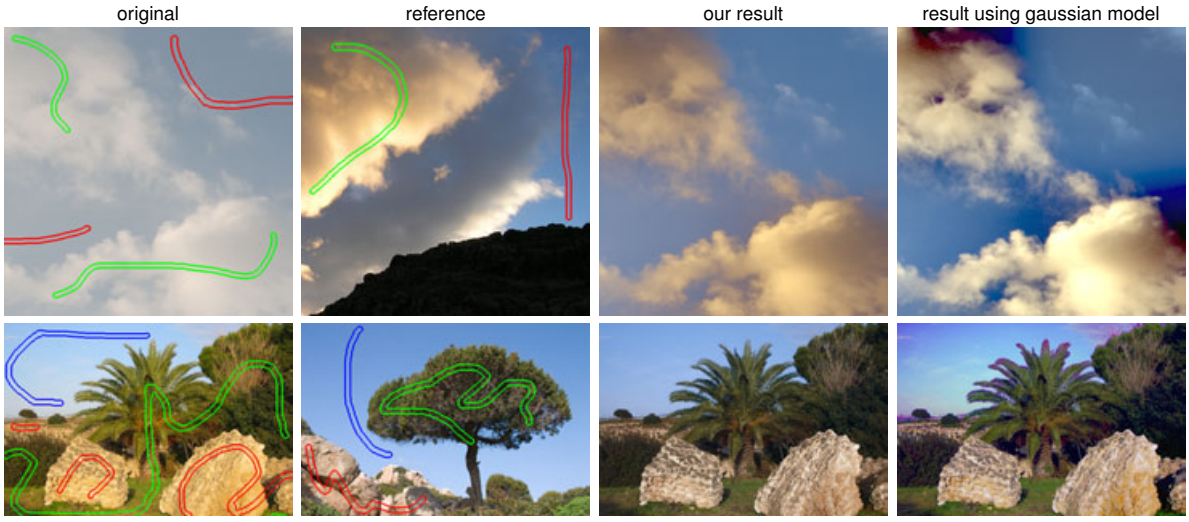

Figure 4: *Comparison between our constrained parametric model and a gaussian model as transfer functions within our propagation framework. The gaussian model displays several issues. Top: an example of a challenging transfer pair where the original image exhibits very little color variance in the clouds and the reference image contains high contrast clouds. Note how our algorithm increases the overall contrast without introducing artifacts, while the gaussian model leads to obvious clipping when extrapolating the model parameters. Bottom: note the hue noises and color shifts introduced using the gaussian model.*

transferred results often end up out-of-gamut, shown as clippings. Second, hue noise and shifting result when using only one linear transformation for the chromaticities across the whole luminance range. Third, linear models often lose contrast since they cannot capture typical nonlinear contrast adjustments.

Histogram matching is a nonlinear transfer model that can obtain exact matches of color statistics within strokes. However, the bijections that histogram matching computes have no guarantee of monotonicity constraints in the luminance channel, and thus can lead to artifacts. More importantly, it is unclear how to interpolate the bijections computed by histogram matching. Our model can match color statistics closely while minimizing visual artifacts and providing well-defined interpolations of model parameters.

3.5. Limitations

We found our transfer model to work well in most image pairs, but there are cases where we fail to properly transfer style within strokes. This happens when matching color "style" would require breaking the monotonicity constraints of our transfer model. An example of such case is shown in Fig. 5. Note that between the stroke pair specified on the two buildings, there is no monotonic mapping that can transfer the color tones because a gradient inversion is needed on parts of the luminance channel. While this is a clear limitation of our approach, to the best of our knowledge this is in fact true for all color transfer models, whether linear or histogram matching, since they all share the underlying assumption that a monotonic transfer should hold. In fact, in cases like this, it is impossible to mathematically define what the desired color edits should be. In our framework, we rely on the user to place more strokes to disambiguate these situations.

In addition, our current framework does not adjust local contrast. To handle this situation we can either attempt to parameterize local transfer within our model or use our algorithm on the base layer of the biscale transfer model and adopt [BPD06] to model local contrast. These additions are largely orthogonal to our work, and as such we leave their investigation to future work.

4. Implementation

Per-Stroke Parameter Estimation. To estimate our model parameters, we need to solve a constrained nonlinear optimization problem, that is generally NP hard. Rather than approaching this problem directly, we approximate the solution with a soft constraint problem where we seek to minimize the function $E'(\rho) = E(\rho) + \delta E_{\text{inf}}$ where δ is 1 when the constraints are disobeyed, 0 otherwise, and E_{inf} is a sufficiently large value. Since the derivatives of function E' cannot be written analytically, we use a simplex algorithm [LJRW96] to solve this optimization. For the initial configuration, we start with a cubic spline that defines an identity mapping on the luminance channel and use the gaussian model of [RAGS01] to initialize the affine matrices on the chormaticity domain.

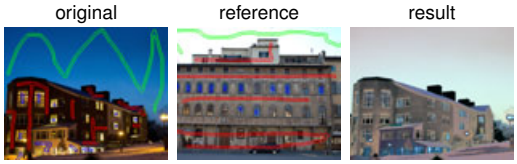

Figure 5: *A failure case of our algorithm.*

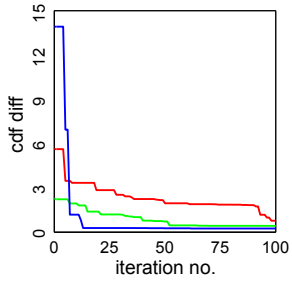

Figure 6: *Convergence of the nonlinear optimization. Colored lines correspond to stroke pairs of the same color in Fig. 2.*

We optimize for a total of four control points corresponding to two cubic segments minus the constrained points. In our model, each control point has eight degrees of freedom, leading to a high-dimensional search space. Since the simplex algorithm can often get stuck at local minima in such high dimensional space, especially when soft constraints are added, we obtain better convergence by alternatively optimizing for the luminance and affine matrices. That is, we first solve the optimization for the luminance control points while fixing the affine matrices and then solve the optimization for all the affine matrices while fixing the luminance control points. We repeatedly perform this procedure until convergence, which we determine by the size of the simplex. In all our experiments, repeating this procedure twice is sufficient for good convergence. This corresponds to roughly 100 iterations of the simplex algorithms. See Fig. 6 for an example of typical convergence behavior of our algorithm.

The inner loop of this optimization requires the computation of the L^2 difference of three dimensional cdfs, which is quite slow in practice. We avoid this bottleneck by approximating this difference as the average L^2 difference of the one dimensional cdfs computed for a set of randomly chosen directions. This method was previously used in [PKD07] and proven to work very well when comparing color distributions. In our implementation, we use ten random projections in the estimate.

Matrix Interpolation. Our method requires the computation of weighted averages of two dimensional linear transformations $M = \sum_i w_i M_i$ that we implement following [SD92]. Here we quickly review the proposed method. First, we factor each matrix as the product of a rotation, non-uniform scaling and shear, such that $M_i = R(\theta_i)Sc(s_i^x, s_i^y)Sh^X(x_i)Sh^Y(y_i)$. In our implementation, we perform this decomposition by first using the polar decomposition method to determine R and then deriving the other factors by solving a linear system of the remaining coefficients. Given this decomposition, we compute the weighted sums of the parameter of each factor (e.g. $\theta = \sum_i w_i \theta_i$) and compute the final matrix as $M = R(\theta)Sc(s^x, s^y)Sh^X(x)Sh^Y(y)$.

5. Results

Edits. We have tested our algorithm by performing color transfer on a number of image pairs that require a variety of color adjustments to obtain the final results. These adjustments often involve complex combinations of common image operations, such as contrast control, saturations changes, hue shifts and temperature adjustments. Fig. 1 and Fig. 7 show complex combinations of these basic editing operations on a variety of scenes. We chose our test to show a variety of scene content, e.g.landscapes and portraits, and image "patterns", .e.g soft gradients, hard edges and textured regions. In all these cases, we found that our nonlinear constrained transfer model captures these complex edits well, while minimizing the presence of visual artifacts in the result images.

Our model works well for landscapes, where it captures edits to the complex patterns of vegetation and buildings and to the gradients in sky, water and fog (e.g.Fig. 1.a, 7.a, 7.b, 7.c). Furthermore, our model captures well edits to portaits, retaining the subtleties of skin appearance even for drastic alterations (e.g.Fig. 1.b, 7.d). Finally, our model allows artistic interpretation of the transferred results by directly controlling the position and shape of the strokes (e.g.Fig. 1.c). For all tested pairs, we include full resolution versions in supplemental materials, where the reader can also find a variety of additional examples.

In all examples, we found our framework to be easy to use and control. For all the image pairs we tested on, stroke pairs took no more than one minute to create.

Together with stroke placement, users choose which propagation semantic they prefer, i.e.whether to softly propagate to the whole image [AP08] or to employ a stricter hard-stopping method [HMP*08]. For example, we use [AP08] for the trees and fog in Fig. 1.a and 7.a where many thin tree branches and soft fog make complex soft selections and smooth interpolation required. On the other hand, we use [HMP*08] for the portrait pairs in Fig. 1.b and 7.d that require precise selections to separate the hair from the background. This flexibility in adopting different propagation models is important to adapt our method to different scenes. Furthermore, the use of very soft selection in the fog scene shows the importance of having a model that correctly interpolates across regions as ours provide.

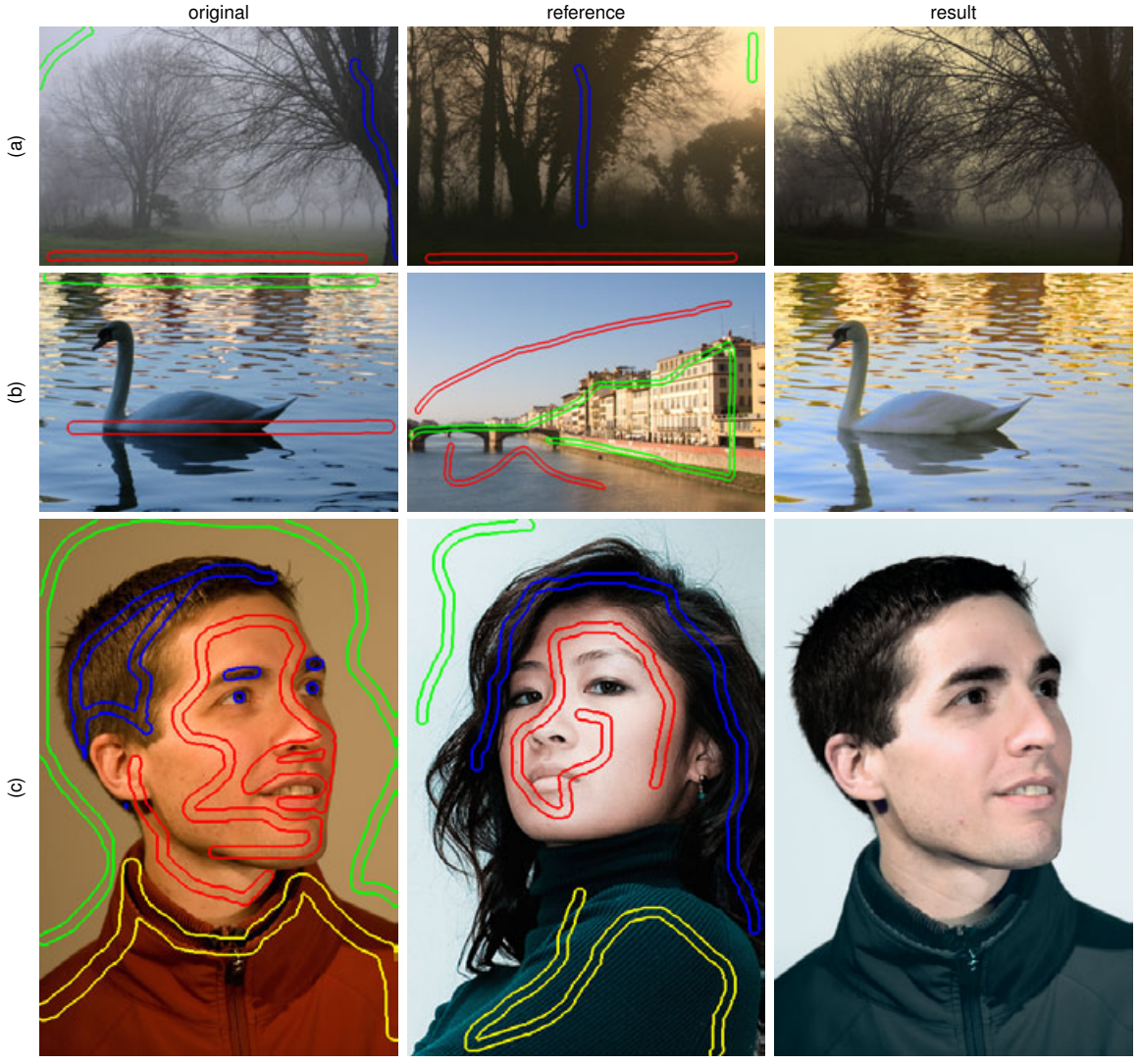

Figure 7: *Color transfer results on a variety of image pairs.*

Performance. We measure the performance of our algorithm on an Intel Core 2 Duo 2.6 GHZ with 4GB of RAM using only one core. Our algorithm has three phases: per-stroke parameter estimation, influence function computation and per-pixel parameter interpolation. We report the timings for parameter estimation and interpolation in Table 1 for all images in the order they appear in the paper. Performance varies with the number of strokes and image size. For a 600×400 image with 3 pairs of strokes, the total time spent on these two phases of the algorithm is two seconds on average using an unoptimized C implementation. For the computation of influence functions, we obtained the implementation from the authors. The implementation of [AP08] requires roughly two seconds for a 600×400 image. We derive the influence functions for [LRAL08] from their unoptimized MATLAB implementation requiring roughly one minute for a 600×400 image, although we believe that this could be drastically optimized. As shown in these timings, our method has roughly the same time requirements as propagation, thus maintaining the near-interactivity of the original models while removing the need for most trial-and-error iterations for parameter settings.

6. Conclusions and Future Work

This paper presents an image editing framework where users indicate color adjustments by marking corresponding strokes in the original image and on a color reference. For each stroke pair, we match the color distributions of the ref-

image	*size*	*s*	*p*	*est*	*int*	*P*
Fig. 1.a	600×400	6	8946	2.28 s	0.46 s	*A*
Fig. 1.b	600×400	4	11833	1.95 s	0.31 s	*E*
Fig. 1.c	600×460	3	15546	1.78 s	0.26 s	*A*
Fig. 2	512×512	3	14309	1.35 s	0.30 s	*A*
Fig. 4.a	512×512	2	10254	0.73 s	0.23 s	*A*
Fig. 4.b	600×400	3	17952	2.14 s	0.24 s	*A*
Fig. 7.a	600×400	3	10501	1.44 s	0.25 s	*A*
Fig. 7.b	600×400	2	11342	1.15 s	0.22 s	*A*
Fig. 7.c	400×600	4	18347	2.31 s	0.32 s	*E*

Table 1: *Performance of our algorithm: size is the image size (for simplicity, we scaled target and reference to be the same size), s is the number of stroke pairs, p is the average number of pixels within each stroke pair (including both target and reference), est is the time spent on nonlinear optimizer (in seconds), interp is the time spent on per-pixel model parameter interpolation, and P is the edit propagation method used: A for AppProp and E for edge-stopping*

.

erence and the original by applying a nonlinear constrained parametric transfer function. Within strokes, this transfer model can match color distributions well, while ensuring that no artifacts are visible in the edited result. To transfer the color to the reminder of the image, we apply known edit propagation methods to propagate the parameters of our model to spatially-close regions of similar appearance. This is possible since our model has well-defined linear interpolation. We demonstrated the effectiveness of our algorithm by performing color transfer on a variety of image pairs varying in image content and style. While all these edits require complex combinations of color and tonal adjustments, they were all performed within a minute with our approach showing how our method drastically reduces the trial-and-error required to perform these adjustments. In the future, we are interested in integrating local contrast control into our framework and extend it to video color transfer.

References

[Ado07] ADOBE SYSTEMS INC: Photoshop CS 3, 2007.

[Ado08] ADOBE SYSTEMS INC: Lightroom 2, 2008.

[AK07] ABADPOUR A., KASAEI S.: An efficient PCA-based color transfer method. *Journal of Visual Communication and Representation 18*, 1 (2007), 15–34.

[AP08] AN X., PELLACINI F.: Appprop: All-pairs appearance-space edit propagation. *ACM Transactions on Graphics 27*, 3 (2008), 40:1–40:9.

[BBS09] BRATKOVA M., BOULOS S., SHIRLEY P.: oRGB: a practical opponent color space for compute graphics. *IEEE Computer Graphics and Applications 29*, 1 (2009), 42–55.

[BPD06] BAE S., PARIS S., DURAND F.: Two-scale tone management for photographic look. *ACM Transactions on Graphics 25*, 3 (2006), 637–645.

[GD05a] GRUNDLAND M., DODGSON N. A.: Color histogram specification by histogram warping. *SPIE Color Imaging X: Processing, Hardcopy and Applications 5667* (2005), 610–621.

[GD05b] GRUNDLAND M., DODGSON N. A.: Color search and replace. In *Eurographics Symposium on Computational Aesthetics* (2005), pp. 101–109.

[HMP*08] HSU E., MERTENS T., PARIS S., AVIDAN S., DURAND F.: Light mixture estimation for spatially varying white balance. *ACM Transactions on Graphics 27*, 3 (2008), 70:1–70:7.

[ICOL05] IRONY R., COHEN-OR D., LISCHINSKI D.: Colorization by example. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering* (2005), pp. 201–210.

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM Transactions on Graphics 25*, 3 (2006), 646–653.

[LJRW96] LAGARIAS J., J. REEDS M. W., WRIGHT R.: Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization 9* (1996), 112–147.

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Transactions on Graphics 23*, 3 (2004), 689–694.

[LLW08] LEVIN A., LISCHINSKI D., WEISS Y.: A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 2 (2008).

[LRAL08] LEVIN A., RAV-ACHA A., LISHINSKI D.: Spectral matting. *IEEE Transaction on Pattern Analysis and Machine Intelligence 30*, 10 (2008), 1699–1712.

[LWX07] LUAN Q., WEN F., XU Y.: Color transfer brush. *Pacific Conference on Computer Graphics and Applications* (2007), 465–468.

[MV07] MASLENNIKOVA A., VEZHNEVETS V.: Interactive local color transfer between images. *GraphiCon* (2007).

[PK07] PITIE F., KOKARAM A.: The linear Monge-Kantorovitch colour mapping for example-based colour transfer. *IEE European Conference on Visual Media Production* (2007), 1–9.

[PKD07] PITIE F., KOKARAM A., DAHYOT R.: Automated color grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1 (2007), 123–137.

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer Graphics and Applications 21*, 5 (2001), 34–41.

[SD92] SHOEMAKE K., DUFF T.: Matrix animation and polar decomposition. *Proceedings of the Conference on Graphics Interface* (1992), 258–264.

[TJT05] TAI Y., JIA J., TANG C.: Local color transfer via probabilistic segmentation by expectation-maximization. *IEEE Computer Vision and Pattern Recognition 1* (2005), 747–754.

[WAM02] WELSH T., ASHIKHMIN M., MUELLER K.: Transferring color to greyscale images. *ACM Transactions on Graphics 21*, 3 (2002).

[WHCO08] WEN C., HSIEH C., CHEN B., OUHYOUNG M.: Example-based multiple local color transfer by strokes. *Pacific Conference on Computer Graphics and Applications 27*, 7 (2008), 1765–1772.

[XLJ*09] XU K., LI Y., JU T., HU S.-M., LIU T.-Q.: Efficient affinity-based edit propagation using k-d tree. *to appear in ACM Transactions on Graphics* (2009).

[XM06] XIAO X., MA L.: Color transfer in correlated color space. *Virtual Reality Continuum and Its Applications* (2006), 305–309.