The Visual Computer manuscript No.
(will be inserted by the editor)

A Framework for Interactive Image Color Editing

Przemyslaw Musialski · **Ming Cui** · **Jieping Ye** · **Anshuman Razdan** · **Peter Wonka**

Received: date / Accepted: date

Abstract We propose a new method for interactive image color replacement that creates smooth and naturally looking results with minimal user interaction. Our system expects as input a source image and rawly scribbled target color values and generates high quality results in interactive rates. To achieve this goal we introduce an algorithm that preserves pairwise distances of the signatures in the original image and simultaneously maps the color to the user defined target values. We propose efficient sub-sampling in order to reduce the computational load and adapt semi-supervised locally linear embedding to optimize the constraints in one objective function. We show the application of the algorithm on typical photographs and compare the results to other color replacement methods.

Keywords image processing · computational photography · color manipulation · interactive image editing · recoloring

1 Introduction

In recent years, digital photography has become very popular in both the consumer as well as the professional domains.

P. Musialski
Arizona State University, Tempe, AZ, USA
Vienna University of Technology, Vienna, Austria
E-mail: pm@cg.tuwien.ac.at

M. Cui · J. Ye · A. Razdan
Arizona State University, Tempe, AZ, USA
E-mail: {ming.cui | arazdan | jieping.ye }@asu.edu

P. Wonka
Arizona State University, Tempe, AZ, USA
King Abdullah University of Science and Technology, Saudi Arabia
E-mail: pwonka@gmail.com

Fig. 1 We show two results of the image color replacement method presented in this paper. The first image is the original (copyrighted by Norman Koren `http://www.normankoren.com/`). In the second image we adjust the color of the sky. In the third, we also adjust the color of the grass.

This development brought about a demand for advanced image processing algorithms that are powerful on the one hand, but easy to use on the other. This includes also the manipulation of the color in pictures – perhaps the most fundamental image processing task ever.

Currently available commercial software, as for instance Adobe Photoshop [1], provide manual color processing tools that are relatively convenient to use, although, sill require a considerable amount of precise user input [20]. They rely on local image characteristics and do not incorporate any global color information into the editing process. Furthermore, there exist a number of approaches which process the image's colors as probability distributions [24,22,31,3], or approaches that provide user controllable adjustment of the colors. The latter ones are either constrained to local editing [15], or incorporate global edit propagation [2]. These methods have proven to provide most satisfying results, but their common disadvantage is usually quite a large computational load due to global optimization.

The approach we present in this paper can be classified as a user controllable one. We rely on rough strokes

drawn on an image, which is efficiently incorporated into the editing process. The main difference of our approach to the already existing work is that we propose a novel formulation of the optimization problem, where we draw from the non-linear manifold learning methodology. We formulate the problem as a global optimization task, and we show that this task can be solved as a sparse linear system. This combines global editing as in An and Pellacini [2], who use a dense solver, and a sparse optimization as utilized in Lischinski et al. [15], who do only local pixel propagation.

A sparse approach to global propagation has been proposed in the work of Pellacini and Lawrence [21] in the context of measured material appearance editing. Their work was also inspired by the manifold-learning methodology [25], however, their solution was not suitable for high-quality image appearance propagation as shown later in the paper of An and Pellacini [2]. Instead, An and Pellacini proposed a formulation which uses a dense least-squares solver that allows them to propagate the affinities of all pairs of pixels to each other in order to maintain the quality. However, their dense linear system generally does not fit into the computer memory for common images. Their remedy is to solve it approximately using the Nystrom-method [11], which is not accurate at small-scale edits and does not scale well for large input. In contrast, the method of Lischinski et al. [15] provides high-quality results and uses a sparse solver, but it propagates the edits only to spatially coherent nearby pixels and requires more accurate user inputs in order to perform well.

In this paper we provide a formulation of the optimization which strives for both a sparse solution as well as global pixel interaction. To achieve this we interpret the image color as a manifold in 3d space by utilizing the locally linear embedding algorithm [26]. We show how the color-manifold can be warped globally in order to achieve recoloring while its local relationships are preserved in order to maintain the appearance of the original image.

In addition, we introduce an efficient sub-sampling strategy in order to achieve interactive performance. Xu et al. [38] proposed a speed-up approach to the formulation of An and Pellacini [2] which exploits the fact that often pixels in the image can be approximated by a much smaller set of clusters. Driven by similar observations we sub-sample the image in order to greatly reduce the number of color points to be processed. We then approximate the manifold with the sub-sampled points and interpolate the remaining values. Unlike the method of Xu et al., we do not need to build piece-wise linear functions each time user provides new input strokes. Instead, we maintain the same sub-sampling for different user inputs, where we only update the user provided target color values. Our method has a small memory footprint, it scales linearly in the number of pixels, and it allows interactive editing. We also show that it delivers results of the same or better quality as others.

In the reminder of the paper we provide an overview of the related work in Section 2. In Section 3 we present the details of our approach and discuss its further aspects. In Section 4 we present the results and compare them to other works, and finally in Section 5 we conclude the work.

2 Related Work

Several papers aim at automatic color transfer between images, where usually one image serves as color mood source which is transferred to the others. Reinhard et al. [24] proposed a simple yet effective method for this purpose based on linear adjustment of color distribution parameters. This has been improved by Xiao et al. [36] and Pitie and Kokaram [22] who applied more sophisticated probabilistic models. In Pitie et al. [23] they extend their method in order to perform non-linear adjustment of color probability distribution between images. Also Chang et al. [6,5] presented global color transfer by perceptual color categorization for images and video. Yang and Peng proposed a method for color-mood transfer which preserves spatial coherence based on histogram matching [39]. This idea has been extended by Xiao et al. [37] who solve the problem of global transfer and local fidelity in two steps: histogram matching and a gradient-preserving optimization. Wang et al. proposed global color-mood exchange driven by predefined and labeled color palettes [31] and example images [32]. Cohen-Or et al. [9] introduced a framework which uses color-harmony rules in order to optimize the overall appearance after the user has altered some of the colors. Shapira et al. [27] proposed a solution which is based on navigation through the appearance of the image in order to obtain desired results. Also automatic methods to colorize grayscale images based on examples from internet images [16], and semantic annotations [8] have been introduced. In general, methods which transfer colors globally are not suitable for precise (re-)coloring of small objects or humans.

Other approaches try to introduce at least rough control over the results. Welsh et al. [34] proposed a global color-mood transfer aiming at colorization of grayscale imagery. It is based on texture and luminance matching across the images and it allows simple user interaction in the form of rectangular swatches, but it also fails in cases of detailed transfers. Tai et al. [29,30] attempted to solve these problems by providing a method for soft color segmentation based on a mixture of Gaussian approximation (GMM) which allow indirect user control. Further improvements of automatic but controllable color-mood transform based on Reinhard et al. is presented in [13].

In contrast to methods mentioned above, locally controllable systems provide the user very accurate influence over

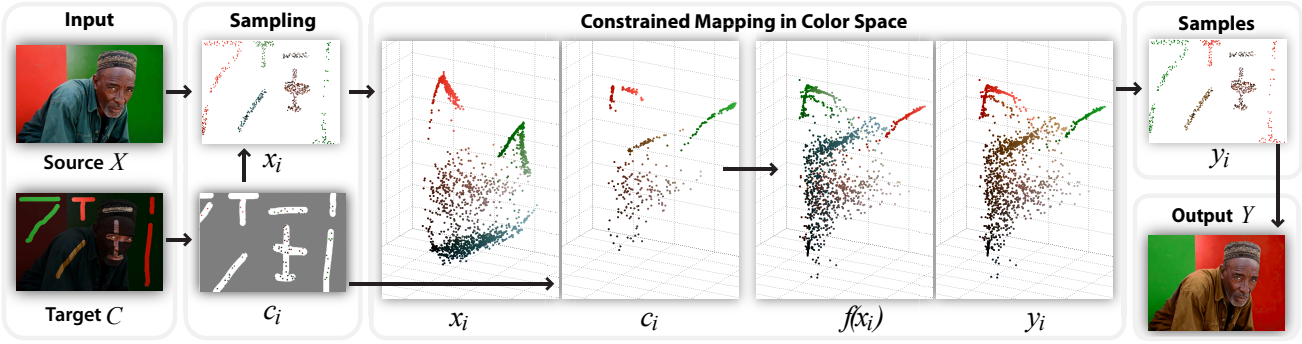

Fig. 2 The pipeline of our algorithm - refer to Section 3 for details. From left to right: input image X and target color map C. From both input images landmarks x_j and c_j are sampled. For all remaining pixels linear interpolation coefficients are computed. The sub-sampled input points x_j (in L*a*b* color space) are warped towards target points c_j under the constraint of mutual distance preservation. Next, points $y = f(x_j, c_j)$ indicate new positions of the landmark points, here shown with old colors. Than we show the final positions and colors after the mapping as points y_j. Finally, the output image is reconstructed from the y_j point set by the previously stored linear coefficients, converted to RGB, and displayed.

the results. In general, these methods allow the user to scribble over the image in order to alter the appearance of similar regions in some parts of the image. A simple example of such an approach is the color transfer brush [17] which applies locally the equations of Reinhard et al. [24], albeit its modeling capacity is very limited. Wen et al. [35] and An and Pellacini [3] also propose strokes driven methods for transfer of color from local parts across images. A scribble-driven method was presented by Yatziv and Sapiro [40] who introduced colorization based on chrominance blending and geodesic distance. It requires quite accurate user inputs in order to perform well. Similar user interaction has been successfully applied for grayscale colorization [12], local image adjustment [15] and edit propagation [2]. Also a bilateral filter based framework, e.g. Chen et al. [7], can be used to recolor particular image parts. Recently, Farbman et al. [10] utilized diffusion distances in the framework of Lischinski et al. [15] which partially allows for more global editing with their solver. On the other hand, they show that the usage of diffusion distance does not generally address the locality problem and can be seen as a complementary approach to Euclidian distance optimization.

The methods of Lischinski et al. and An and Pellacini are based on least-squares optimization and are similar to our approach. Though, we present a different formulation of the problem by drawing from the locally linear embedding approach [25,26]. The main difference of our system is the way how pixel neighborhood weights as well as how target colors are incorporated into the solution.

Further work are speed-up methods, like Xu et al. [38] who proposed an acceleration to the approach of [2] based on kd-tree-subdivision of the image, but they still utilize the dense solver. Li et al. [14] formulate the problem as Radial-Basis-Function kernels interpolation. We also utilize interpolation in the first step, but still perform global optimization in the second, since our observations have shown that pure local interpolation can provide artifacts.

Recently, Carroll et al. [4] proposed an interactive approach in order to decompose the input image into its Lambertian illumination components. Since their actual color replacement method is independent of their model, our method can be seen as complementary.

3 Controllable Optimization Algorithm

In this paper we introduce a strategy for color replacement that combines two apparently contradictorily goals. The first is *distance preservation* that ensures that two color-samples from the *source image* are mapped in such a way that the distance between them in the new image remains similar. The second goal is *color transfer* that aims at mapping of the samples from the source image as near as possible to the user-provided target values in the *target image*. The advantage of this idea is the fact that re-mapped colors generally retain their local variations on the one hand but change their global appearance to the desired values on the other. All together, this results in very naturally looking output images (cf. Figures 1,14).

3.1 User Interaction

As observed in previous work [12,15,40,21,2], user edits in form of rough strokes have proven to be an easy and efficient way of interaction. Pellacini et al. [21,2] defined user edits more generally as edit parameters that should be propagated over the output image. In our system the user indicates the desired output appearance (color, hue, saturation or lightness) in the form of rough strokes over the input image. The result is a sparsely scribbled image which we henceforth call the *target image*. The expected strokes do not have to be precise and do not have to match the boundaries of the underlying objects very well; only a clear assignment of the new color to an object is important. Our edits can be

sparse or dense, which is more similar to the way interaction is applied in the work of [2], while the interaction in other methods [12,15,40] has to be more precise. In general, we want the user to specify the color for all image parts, even those which should remain unchanged. Figure 3 shows a target image on the left and an alpha mask on the right. In the left image, the storks indicate the user input, where both the changed colors of the background as well as the kept colors of the face have been scribbled. The mask indicates that only the white regions contribute target values. This kind of interaction is rather easy and intuitive, even for untrained users.

3.2 Definitions

We define the source image as X, the target image as C and the output image as Y. The images have the size $N = width \times height$ pixels and can also be seen as sets of points $X = \{x_1, x_2, ...x_N\}$, $Y = \{y_1, y_2, ...y_N\}$, and $C = \{c_1, c_2, ...c_N\}$, where all $x_i, y_i, c_i \in \mathbb{R}^D$ reside in the same D-dimensional space. Note that all points x_i, y_i and c_i with the same index correspond to each other. We want to compute a non-linear mapping $f : \mathbb{R}^D \to \mathbb{R}^d$ that transforms the given input image X with respect to the user defined target image C to a new image Y, such that:

$$f : (X, C) \to Y\,. \tag{1}$$

We provide here a definition for the general D-dimensional case since our algorithm is not limited to a specified number of input and output dimensionality. The input space is of the dimension D and the output space of the dimension d, usually such that $d < D$. In Section 3.6 we discuss the possibility to use local image patches as points x_i as well as the case where a color image can be mapped to a user defined grayscale.

In practice, we usually work with 3d color spaces, thus for the rest of the paper we assume $D = d = 3$ w.l.o.g. Further, we follow the argumentation which suggests that the Euclidian norm in RGB color space is not a good measure for perceptual distance. Therefore, throughout the paper we measure and compute the distance between points in a perceptually Euclidian color space CIELAB (L*a*b*) [28] and all norms are L_2 vector norms denoted as $\|\cdot\|$.

Fig. 3 Example of a source and target image. Left: we ask the user to specify the target colors by simply drawing rough strokes over the original. Right: values from the selected regions (white) serve as target points. Note that also target values of non-changed points have been specified.

3.3 Optimization Formulation

We want to define the mapping f of vectors x_i to vectors y_i, such that it preserves pairwise Euclidean distances of x_i as well as forces y_i to be as close as possible to c_i. This task can be formulated as minimizing the following equation:

$$E = \sum_i \sum_j (\|x_i - x_j\| - \|y_i - y_j\|)^2 + \lambda \sum_i \|y_i - c_i\|^2\,, \tag{2}$$

where λ is a parameter that determines the relative importance of the two goals. Unfortunately, the function in Equation 2 is non-smooth and thus hard to solve effectively. To relax it to a solvable problem, we replace the first term in E, such that:

$$\tilde{E} = \sum_i \|y_i - \sum_j w_{ij} y_j\|^2 + \lambda \sum_i \|y_i - c_i\|^2\,. \tag{3}$$

This equation is quadratic in terms of the unknowns y_i. Note that with Equation 3 we do not provide a strict mathematical reformulation of Eq. 2 but rather than an approximation in the locality of each x_i.

The main idea of this reformulation is to encode the geometric invariance in such a way that it can be expressed as a quadratic term of the unknown y_i. We do so by utilizing the locally linear embedding algorithm (LLE [25]), which generates a manifold in the underlying space which is linear at each sample point with respect to its local neighborhood. This is achieved by "encoding" the pairwise relations $\|x_i - x_j\|$ of the original samples and their neighbors (cf. Eq. 2) into the weights w_{ij} (cf. Eq. 3). Saul and Roweis [26] have shown that properly chosen weights w_{ij} are invariant under rotation, translation and scale. This means that each particular output color sample y_i is placed in the new image in such a way, that the distances to its neighbors reassemble the distances of the original color sample x_i in the original image as best as possible in the least squares sense (Fig. 4).

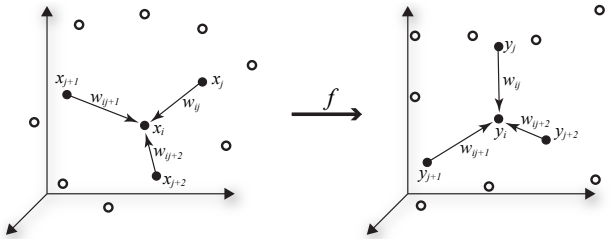

Fig. 4 Left: the weights w_{ij} are computed in the original image using the original samples x_i and their neighbors x_j. Right: the same weights are used to best reconstruct all output samples y_i from their respective neighbors y_j in one large linear system.

Fig. 5 A series of results of our algorithm. The first image is the original (copyrighted by Norman Koren http://www.normankoren.com/). The following results are obtained with optimization of the chroma-channels a* and b* only, while the L* channel is kept from the original.

The weights can be computed for each x_i as a linear combination of its nearby points x_j by minimizing the following energy for each x_i independently:

$$F = \sum_{i=1}^{N} \|x_i - \sum_{j \in \mathcal{N}_i} w_{ij} x_j\|^2 , \tag{4}$$

with respect to the invariance constraint: $\sum_j w_{ij} = 1$ and to the sparseness constraint: $w_{ij} = 0$ if $j \notin \mathcal{N}_i$. Here $\mathcal{N}_i$ denotes a (small) set of local neighbors of the point x_i in X. The optimal weights w_{ij} can be computed in closed form. Due to the mentioned constraints, we can rewrite Equation 4 for one data point x as:

$$\|x - \sum_j w_j x_j\|^2 = \|\sum_j w_j (x - x_j)\|^2 = \sum_j \sum_k w_j w_k g_{jk} ,$$

where g_{jk} is an entry of a local Gram matrix $\mathbf{G} = \{g_{jk}\}$ with elements: $g_{jk} = (x - x_j)(x - x_k)$, with x_j and x_k as neighbors of x. This matrix is symmetric positive semi-definite and the weights of x can be computed by its inversion. A more efficient way is to solve a linear system of the form:

$$\sum_k g_{jk} w_k = 1 ,$$

and rescale the weights to $\sum_j w_j = 1$. This system can be solved for all N D-dimensional points with K neighbors in together $O\left(DNK^3\right)$ time. In practice this system is overdetermined in the case when the number of neighbors is bigger then the dimensionality of the space. In our application this is usually always the case, since we work with only 3d points and we have empirically figured out that the number of neighbors should be about 11 in order to provide good results. Thus, to solve for unique weights we utilize Tikhonov-regularization by adding a small multiple of the identity to the coefficient matrix as proposed in [26]. This provides weights which distribute the contribution of the nearest points to each x_i more uniformly.

In fact, the described weighting is the main difference of our approach to the others, e.g, Lischinski et al. [15], Chen et al. [7], or An and Pellacini [2]. In those systems the weighting of the neighbors is usually accomplished by the exponential fall-off function of their (Euclidian) distance d to the particular point: $w_{ij} = \exp(\|x_i - x_j\|)$. While those weights are in general edge-aware and smooth, they do not represent the particular point as a linear combination of its neighbors as the LLE weights do.

Having well-defined weights, Equation 3 can be minimized. Since for each data point in the original space w_{ij} are invariant to rotation, scaling and translation of this point w.r.t. its local neighbors, minimizing $\tilde{E}$ has locally the same effect as of minimizing E. Globally, the manifold is bend and in general E is not enforced for distant points, but this solution is even more desirable, since it allows to fulfill the color transfer more easily: the manifold is warped towards the target values c_i (cf. Figure 2). One might imagine this operation as pulling the entire manifold on the selected points x_i towards new values c_i. Since each of the selected points is connected to its local neighbors and each such a local vicinity can be transformed linearly, the pulling process affects the entire manifold and results in new positions y_i which ideally respect our both goals: *preservation of local distances* as well as *global color transfer*.

3.4 Acceleration

The presented algorithm is designed to work with theoretically all pixels in the image. Unfortunately, it would require target values for all pixels and providing such targets is tedious and not desirable. Further, the computation time would be very high. In order to address both problems our approach is a sub-sampling strategy which deals with sparse target values and reduces the computational load significantly. It is based on the observation that all color-points can be expressed by linear combinations of other points. Thus, our idea is to determine a number of significant sample points which we call *landmark points* and to run the optimization only on these. The remaining points are reconstructed as linear combinations of the landmarks.

We determine the landmarks using the original point set X: we draw a random index set $\mathcal{J}$ of the size $|\mathcal{J}| = M << N$ from the full index set $\mathcal{I} = \{1 \ldots N\}$ of all points. In order to get significant points into $\mathcal{J}$, we require the chosen points x_j to be (1) unique and (2) linearly independent such that they form a (generalized) Delaunay triangulation in $\mathbb{R}^D$. For

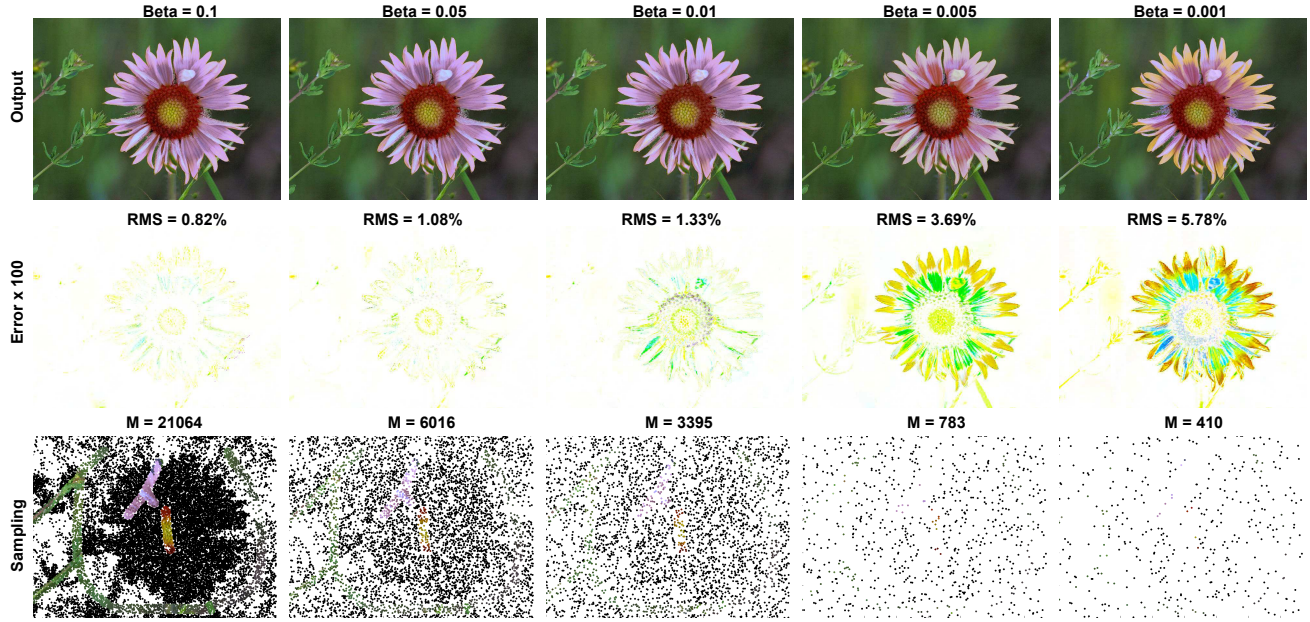

Fig. 6 Comparison of the influence of the parameter β on the results with respect to a reference image which we have computed with $\beta = 1$. The middle row shows the normalized RMS-error image, inverted and amplified by factor 100 for visualization propose. The last row shows the actual landmarks sampled with respect to β. We used the original shown in in Figure 5, left.

each of the remaining points x_i in the set $\{i|i \in \mathcal{I} \backslash \mathcal{J}\}$ we determine the $(D+1)-$dimensional simplex $\mathcal{S}$ in which it is contained and compute its linear coefficients $\mathcal{L}_i$ with respect to $\mathcal{S}$. Now, all points x_i can be reconstructed as linear combinations of the vertices of their Delaunay-simplices, thus, $\mathcal{L}_i$ are in fact barycentric coordinates. Note that they have to be computed only once in the preprocessing stage.

Now we solve the problem of Equation 3 only for the landmark points $\{y_j|j \in \mathcal{J}\}$ and all other points $\{y_i|i \in \mathcal{I} \backslash \mathcal{J}\}$ are computed as linear combinations of the known points y_j using the previously computed linear coefficients $\mathcal{L}_i$. Also the target values can be assigned in a user interaction pass to landmarks points $\{c_j|j \in \mathcal{J}\}$ only.

The sub-sampling rate of the points is controlled by the ratio β, such that $M = \beta \cdot N$. This value has influence on the computation speed but also on the quality of the resulting images. Increasing this value provides more accurate results since the reconstruction error of the images is lower. The rationale is that the more landmark points are sampled the underlying manifold is better approximated. The drawback is the longer computational time. In empirical experiments we have found that the value of $\beta = 0.01$ is a good tradeoff between speed and quality. Figure 6 depicts this relationship.

3.5 Constrained Sparse Least-Squares Solution

In Section 3.3 we have formulated the problem of color-mapping as sparse optimization and in Section 3.4 we proposed an approach to further reduce the number involved points. In this section we propose an efficient solution.

The minimization problem of Equation 3 is quadratic and can be formulated in matrix form as:

$$\tilde{E} = \|\mathbf{M}\mathbf{y}\|^2 + \lambda\|\mathbf{y} - \mathbf{c}\|^2, \tag{5}$$

where $\mathbf{c}$ are the sampled target points arranged in a vector and $\mathbf{y}$ are respective points in the output image Y. The matrix $\mathbf{M}$ is the sparse coefficient matrix of the pairwise weights given by $\mathbf{M} = [\mathbf{I} - \mathbf{W}]$, where $\mathbf{W}$ contains all respective w_{ij} as entries. Notice that henceforth we operate only on the sub-sampled points. In addition, the system is sparse because for each point the weights are zero except for a small neighborhood of the size K. The total number of elements is thus MK.

In the terms of LLE, the d-smallest eigenvalues of $\mathbf{M}^{\mathrm{T}}\mathbf{M}$ provide a lower dimensional manifold of the underlying data (refer to [26] for details about LLE). In our case we strive for another solution since we have prior information provided by the user in the target image C. Even if this information is incomplete because the target values are not given to all pixels c_i, we can still resort to constrained least squares and solve Equation 5 by incorporating only partial prior into the solution – for instance using the method of weighting. Without loss of the generality we can assume that the points which correspond to user-assigned target values are stored in the vector $\mathbf{y}_1$, and the respective target points are in the vector $\mathbf{c}_1$. Points with unknown target are stored in $\mathbf{y}_2$. Finally, rearranging the rows of $\mathbf{M}$ such that the $\mathbf{y}_1$ points correspond

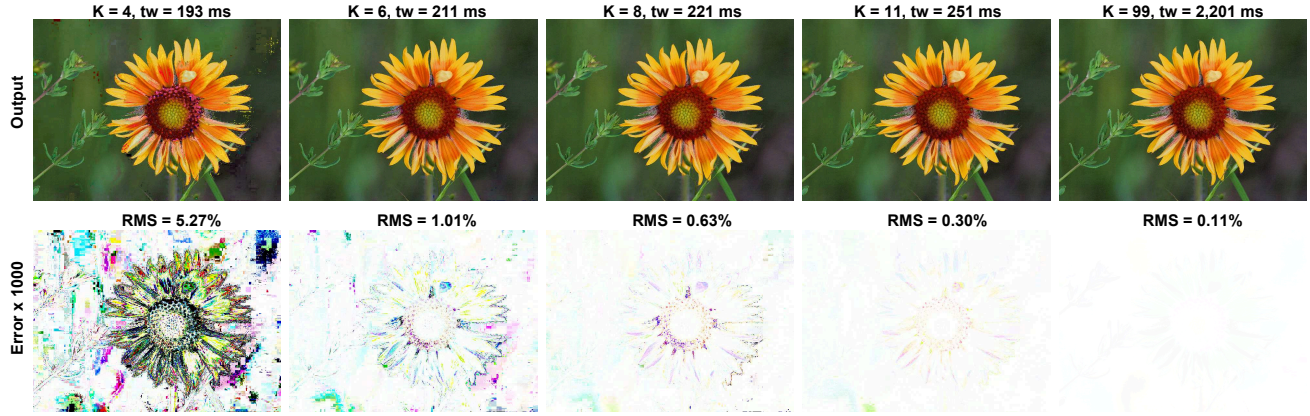

Fig. 7 Comparison of the influence of the number of neighbors K of the LLE embedding with fixed $\beta = 0.01$. The error is measured with respect to the original image (Figure 5, left). The error-image is inverted and multiplied by the factor of 1000 and for visualization purposes. The time t_w is the computation time of LLE weights with respect to K.

to the $\mathbf{M}_{11}$ rows in:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{12}^{\mathrm{T}} & \mathbf{M}_{22} \end{bmatrix},$$

we can rewrite Equation 5 as:

$$\tilde{E} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}^{\mathrm{T}} \mathbf{M}^{\mathrm{T}}\mathbf{M} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} + \lambda \begin{bmatrix} \mathbf{y}_1 - \mathbf{c}_1 \\ \mathbf{0} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{y}_1 - \mathbf{c}_1 \\ \mathbf{0} \end{bmatrix}. \quad (6)$$

The problem can now be solved as an augmented system of linear equations of the form:

$$\begin{bmatrix} \mathbf{M}_{11} + \lambda\mathbf{I} & \mathbf{M}_{12} \\ \mathbf{M}_{12}^{\mathrm{T}} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \lambda\mathbf{c}_1 \\ \mathbf{0} \end{bmatrix}. \quad (7)$$

We observed that we obtain smooth results by setting the value of $\lambda = 0.001$ using the direct sparse MATLAB solver.

3.6 Discussion

Choice of the Parameters. The presented method has two free parameters: the first one is the sub-sampling factor β and the second one is the number of nearest neighbors K in the optimization part. In general both influence how well the color-manifold will be approximated. Furthermore, the quality of the generated manifold also depends on the input provided by the user which we discus in the next paragraph. In empirical experiments we have determined that usually 1% of the pixels of the image ($\beta = 0.01$) is enough to approximate the color manifold for the color exchange purpose. Figure 6 depicts this issue. The number of nearest neighbors for the locally linear embedding approximation is in all our examples (except stated otherwise) $K = 11$. In Figure 7 we show the RMS-error of the result depending on the choice of K, which we measure on the reconstruction of an image with $\beta = 0.01$ with respect to the original image. Here we see that a higher number of neighbors does not result in significantly better approximation. This is not surprising since also Saul and Roweis [26] have reported that LLE performs best in a certain range of chosen neighbors. For this reason we have fixed the parameter at $K = 11$.

User Input. A limitation of our method is the fact that we have to provide prior information to all objects present in the image. This means that the user has to provide input strokes also for those regions of the original image that should remain unchanged. Figure 8 depicts this issue. The reason of this limitation lies in the nature of the computed manifold – the weights which encode the geometric invariance are relative to the chosen neighborhood. Thus, color-points relations in a neighborhood are kept with respect to each other, but the global position of a particular neighborhood is undefined. In order to handle this, we provide target values for chosen points, and since these are connected to others, forcing them towards the target affects their neighbors as well. If we do not provide target values to some particular regions, it is not ensured that they will keep their original position, as shown in Figure 8, right.

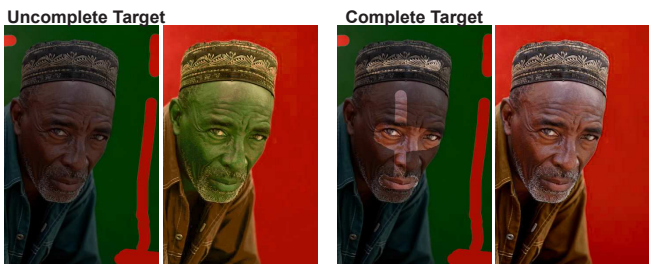

Fig. 8 Left: target is only provided to the background. Right: target additionally provided to the unchanged parts. In order to obtain correct results the user has to provide target values to all image objects, even those which remain unchanged.

Chroma Distance. In the L*a*b* space the color is expressed by the a*b* chromatic components whereas L* holds the lightness [28]. Thus, we can change the goal to preserve the chromatic distances only by computing the distance as $C^*_{ab} = \sqrt{a^{*2} + b^{*2}}$ and the output y_i will be calculated only in the 2d a*b* plane. Since the lightness channel is not taken into account, we are now free to assign it to any value without affecting the metric. The obvious choice is to set it to the values as in the lightness channel of the original image. In praxis it has turned out that if the desired changes do not affect the lightness, like contrast corrections, best results can be achieved by only using the a*b* components, as shown in Figure 5.

Spatial Components. One interesting issue not mentioned sofar is the incorporation of the spatial distance of the pixels into the computation. This allows to modify colors only locally. We facilitate it by adding two spatial dimensions to the input data points x_i and normalize them into the range $[0..1]$. We also add an additional spatial scaling parameter α such that we can weight the spatial components. These two dimensions encode the spatial relationship for the input pixels and the nearest neighbor search algorithm will consider both color similarity and spatial distances. The rationale behind this modification is that, for the task of image re-colorization, we do not need the mapping to be a globally consistent. A similar approach has also been used in the method of Pellacini and Lawrence [21]. Using such a formulation constrains the propagation of colors to spatially nearby points for $\alpha > 0$, which makes our system to act more similar as the approach presented in Lischinski et al. [15] (cf. Figure 9). Without the spatial component, the color is exchanged globally over the entire image. Note that higher-input dimensionality introduces more computational effort to the interpolation stage, since we have to compute 5d barycentric coordinates.

Varying Dimensionality. As mentioned in Section 3.2 our method is derived from the LLE approach which is also an unsupervised dimensionality reduction tool. This is since the weights computed on the original input X are not subject to any specific dimensionality, but rather contain local geometric properties. These are than propagated to the target Y which can be in general of any other dimensionality. By using local image patches of the size, e.g., 5×5 pixel we can obtain 75-dimensional input points. Weights can be computed on this input in exactly the same manner as in the 3d case.

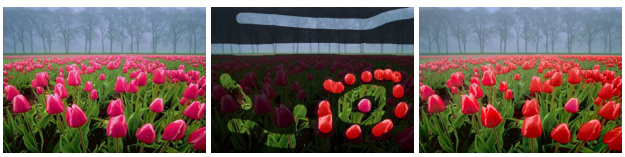

Fig. 9 Mapping the one of the red flowers in the left image to a different hue by incorporating spatial coordinates.

We have experimented with this approach, but similar like Farbman et al. [10] we could not achieve any significant improvements in the appearance changes. Note that a bigger neighborhood results in more computational effort. Furthermore, high-dimensional input introduces a problem to the sub-sampling and interpolation stage. Thus, it remains future work to investigate the possibilities of the dimensionality reduction properties of our solver.

Another issue is the fact that in our approach the desired output dimensionality is also free to be chosen. Usually this will be either a 3d-color or a 1d-grayscale. In the latter case the user is free to assign custom grayscale values to particular colors or to partially map color to grayscale. In Figure 15 we show an example where the output is partially mapped to a black-and-white image.

Finally, one might consider using the method for lifting the dimensionality of grayscale imagery by providing color priors. We have experimented with this idea and approached a number of difficulties due to the ambiguity of the one-dimensional signal. While our method does work in cases where each region of the grayscale image can be mapped uniquely to a color, this remains an exception from the general case. Colorization requires a more involved integration of gradients in the spatial domain [12], which is not directly part of our framework. We consider to explore this issue in a future project.

4 Results and Applications

We implemented the algorithm in MATLAB 2010a. We use the ANN toolkit [18] to determine nearest neighbors, which is a C++ library. Our prototype is currently not optimized for speed, but for easy distribution and maintenance. For user interaction we used Adobe Photoshop CS5 to draw target strokes, which we imported directly in MATLAB using the MATLAB-Photoshop interface.

4.1 Performance

If run on the whole image our optimization algorithm would take a few minutes on an average desktop PC (we use Intel-I7@3.6GHz, 8GB RAM and Windows 7-64bit). However, with the acceleration presented in Section 3.4 the computation can be speed-up considerably. Table 1 shows the running times of examples presented in Figure 14 with sampling rates of $\beta = 0.01$ and $K = 11$, where we distinguish between preprocessing time for computation of the linear coefficients and interactive editing time, where the optimization is solved. The bottleneck is currently the quite slow

Table 1 Running times for the examples shown in Figure 14 with sampling ratio $\beta = 0.01$. Time given in milliseconds and M is the number of used samples. The reported times are t_d for L*a*b* conversion and barycentric coordinates computation, t_w for weights computation, t_e for computation of the mapping, t_r for reconstruction and the conversion from L*a*b* to RGB. Note that only the last two operations have to be performed after user interaction.

			Preproc.		Interactive		Total
Fig.	Size	M	t_d	t_w	t_e	t_r	t
14.1	820 × 547	3797	1,073	481	23	145	1,722
14.2	820 × 546	4271	1,153	276	41	222	1,692
14.3	820 × 546	3646	964	397	24	204	1,589
14.4	820 × 547	3690	1,055	491	33	191	1,770

implementation of the barycentric coordinates computation, where we are using the method of MATLAB. Note that this step as well as the step of weights computation are highly parallelizable since each point is processed independently. Further, both steps are preprocessing done after loading the image; during the interaction only the optimization and interpolation steps have to be performed. Here we can see in the table that the optimization is very fast, even without code optimization. This is due to the quite small and sparse linear system. The interpolation and color conversion steps are again state-of-the-art routines.

Moreover, all particular operations of our method scale linearly with the number of pixels, which is visible in Figure 10 and Table 2. For this test we have used the same image at 7 different resolutions, using the same user input. Notice that we double the number of pixels in each measurement depicted in the chart.

Finally, the memory of the solver is bounded by the number of samples M, where the sparse matrix $\mathbf{W}$ contains KM entries and the target vector at most M. For the interpolation we have to maintain $(D+1)N$ linear coefficients and

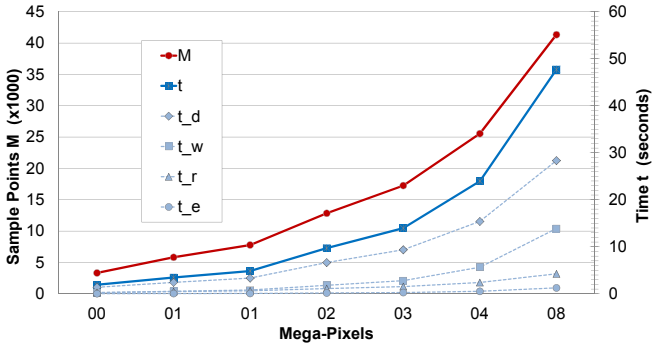

Fig. 10 Comparison of total running time (in seconds) with respect to image size in mega pixels and the number of sample points (M). The total time curve (t) represents the sum the respective timings: t_d, t_w, t_e, and t_r as described in Table 1. Note that the image size in mega-pixels is doubled in each step, thus the running time is linear in the number of pixels. Refer to Table 2 for details.

Table 2 Comparison of total running time (in seconds) with respect to image size in mega pixels (MP) and the number of sample points (M). The respective timings t_d, t_w, t_e, and t_r as described in Table 1, t denoted the total time. Refer to Figure 10 for a graphical interpretation.

MP	M	t_d	t_w	t_e	t_r	t
0.4	3333	1.379	0.246	0.025	0.260	1.911
0.8	5820	2.427	0.518	0.050	0.459	3.455
1.1	7778	3.335	0.804	0.073	0.627	4.840
1.9	12846	6.638	1.800	0.168	1.100	9.704
2.6	17267	9.365	2.793	0.267	1.530	13.955
4.2	25564	15.391	5.653	0.517	2.395	23.956
7.5	41342	28.332	13.885	1.245	4.243	47.706

$(D+1)N$ indices, where N is the number of pixels. Our method is also generalizable to video input, similar as proposed by Levin et al. or Xu et al. [12,38]. We relegate the implementation of video-processing to future work, but we do expect to retain the same performance.

4.2 Applications

Appearance Propagation. In Figure 11 we compare our results to these of the appearance propagation (AppProp) method [2]. The first two images are taken from their webpage. In general the results are comparable, but note that we use a sparse solver with additional sub-sampling, while AppProp uses a dense approximation. The last example is generated by our implementation of the AppProp algorithm (we have implemented it in MATLAB). Here we perform a very drastic color swap usually not shown in the examples of AppProp. In the close-up view in Figure 12 we show that our method provides much smoother transitions on the boundaries of distinct parts within an image.

Figure 13 (f) shows the result of the propagation method of Pellacini and Lawrence [21] generated with an appearance graph with 10 nearest neighbors. Our result is also generated with $K = 10$ and our solution does not provide artifacts. There are two major differences between these two methods: (1) our formulation uses different weights for the neighbors which are computed in a linear system, such that they reconstruct the input. The weights in AppWand [21] are defined by an exponential fall-off function of the Euclidian distance between the neighbors to the actual (BRDF) samples. While these weights reflect the distance of the points in the BRDF-space, they to not reproduce the point from its neighbors in the least-squares sense as our weights do (cf. Section 3.3). The second difference (2) is that our nearest-neighbor graph is fully connected since we determine a fixed number of neighbors for every point. This ensures that our color-manifold is a single connected component.

Figure 13 (c) shows a comparison to the results of Farbman et al. [10] where we can see that our method propagates the color more exact than the other. We can see it in the lower

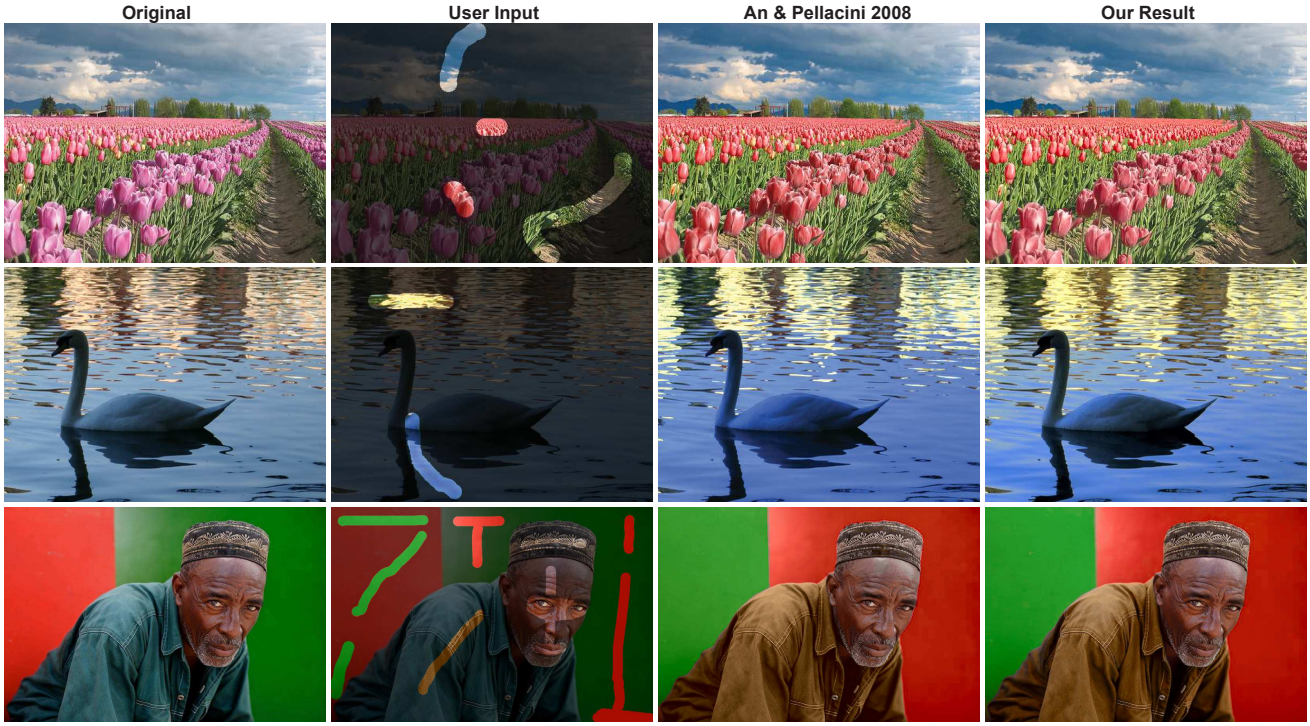

Fig. 11 Comparison to the results presented by [2]. We can observe that we can reproduce the results. The last result is done on our input and we perform very drastic color swap (red is swaped with green and the shirt is recolored). Here we can see (cf. Figure 12) that our method provides smoother transitions between the image objects. Last image copyrighted by Tom Ang (http://www.tomang.com/).

left corner of the fruits-container which is not fully covered by the diffusion distance.[1]

In Figure 13 (d) we compare our results to those of the bilateral-grid framework [7]. Here we see that our method exchanges only the selected colors and does not bleed over to neighboring objects as it partially happens in the bilateral-grid example. Again, the main difference here is that the grid-framework is much more dependent on the local neighborhoods in the spatial domain, unlike our system which establishes neighborhood links across the entire image.

Illumination Color Transfer. Recently Carroll et al. [4] introduced a method which decomposes the input image according to an illumination model. Their method produces very naturally looking results, but the cost is a more sophisticated model. Furthermore, their work aims mainly at the decomposition of the image, the actual color adjustment is performed using Photoshop and Robust Matting [33]. In contrast, our image color model is not physically driven thus we are not able to accurately reconstruct their results. Nevertheless, we try to create a similar output as shown in Figure 13 (a). In fact, the final appearance depends on the strokes provided and thus on the user.

[1] Note that there is another high-level relationship between diffusion distance and locally linear embedding since both methods are based on spectral graph analysis. However this issue does not affect our algorithm and is beyond the scope of this paper (cf. Nadler et al. [19]).

Fig. 12 Close-up of the comparison shown in Figure 11. We can see that our method provides smoother transitions between objects of different color, like wall and skin.

Global Color Transfer. Color transfer is usually established by probability distribution adjustment, which is in general more or less sophisticated histogram adjustment. In Figure 13 (e) we show the results of our method in comparison to the global method of Pitie and Kokaram [22]. While we do not transfer the structure of the background, our method provides a result which brings the colors of the example over the original structure. Moreover, small detail, like the blooms are well preserved.

Color Replacement Tool. Figure 13 (b) shows the reverse of the color replacement tutorial for Photoshop [20]. In this tutorial the author shows particular steps how to manually replace a color in an image with Photoshop. We have re-

versed the results of the tutorial and replaced the violet color of the horse back to brown. We did so due to the lack of the original, brown horse image, but the workflow of the process is essentially the same. The pure editing time to re-color the image with Photoshop as described in the tutorial took us over one minute. Additionally, in Photoshop the user has to adjust several parameters, like brush-size, tolerance, mode, etc. On the other hand, scribbling 3 or 4 rough strokes and the solving time took all together about 10 seconds. This difference would become even more evident if the task were to recolor many different objects in an image, which would require a lot of precise interaction in Photoshop.

5 Conclusions

In this paper we proposed a framework for editing of the color in images and photographs. It allows to replace the color appearance in a smooth and seamless manner with simple user input which has proven to be convenient. Our method shows to perform well for wide range of motives, like landscapes, humans, animals, plants and fuzzy objects. In general the algorithm proves to be convincing and delivers results which appear highly natural. We compare our results to these of related work and we show that we can achieve the same or better quality. On the technical side, we propose a sparse solution to the global least squares problem, while we still maintain global propagation of the color appearance. To achieve it, we draw from the non-linear unsupervised manifold learning methodology and show how to utilize it for image processing. This has not been done in the previous works. In addition, we propose a simple acceleration technique based on sub-sampling and multi-linear interpolation.

One of our goals for future work is to extend the approach in order to process video. Further, we want to investigate more involved ways to incorporate spatial control.

Acknowledgements

This research was financially supported by Science Foundation Arizona, US Navy, and NSF. We would like tom thank Tom Ang (Fig. 14) and Norman Koren (Fig. 1, 5) for the permission to use their outstanding photographs.

References

1. ADOBE Inc. Photoshop. http://www.adobe.com/products/photoshop.html, 2012.
2. Xiaobo An and Fabio Pellacini. AppProp: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics*, 27(3):1, August 2008.
3. Xiaobo An and Fabio Pellacini. User-Controllable Color Transfer. *Computer Graphics Forum*, 29(2):263–271, June 2010.
4. Robert Carroll, Ravi Ramamoorthi, and Maneesh Agrawala. Illumination decomposition for material recoloring with consistent interreflections. *ACM Transactions on Graphics*, 30(4):1, July 2011.
5. Youngha Chang, Suguru Saito, and Masayuki Nakajima. Example-Based Color Transformation of Image and Video Using Basic Color Categories. *IEEE Transactions on Image Processing*, 16(2):329–336, February 2007.
6. Youngha Chang, Suguru Saito, Keiji Uchikawa, and Masayuki Nakajima. Example-Based Color Stylization of Images. *ACM Transactions on Applied Perception*, 2(3):322–345, July 2005.
7. Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics*, 26(3):103, July 2007.
8. Alex Yong-Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu-Wing Tai, Siu-Yeung Cho, Ping Tan, and Stephen Lin. Semantic colorization with internet images. *ACM Transactions on Graphics*, 30(6):1, December 2011.
9. Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. Color harmonization. *ACM Transactions on Graphics*, 25(3):624, July 2006.
10. Zeev Farbman, Raanan Fattal, and Dani Lischinski. Diffusion maps for edge-aware image editing. *ACM Transactions on Graphics*, 29(6):1, December 2010.
11. Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nyström method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–25, February 2004.
12. Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689, August 2004.
13. Meng-Tsan Li, Ming-Long Huang, and Chung-Ming Wang. Example-based color alternation for images. In *2010 2nd International Conference on Computer Engineering and Technology*, pages V7–316–V7–320. IEEE, April 2010.
14. Yong Li, Tao Ju, and Shi-Min Hu. Instant Propagation of Sparse Edits on Images and Videos. *Computer Graphics Forum*, 29(7):2049–2054, September 2010.
15. Dani Lischinski, Zeev Farbman, Matt Uyttendaele, and Richard Szeliski. Interactive local adjustment of tonal values. *ACM Transactions on Graphics*, 25(3):646, July 2006.
16. Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *ACM Transactions on Graphics*, 27(5):1, December 2008.
17. Qing Luan, Fang Wen, and Ying-Qing Xu. Color Transfer Brush. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 465–468. IEEE, October 2007.
18. David M. Mount and Sunil Arya. ANN: A Library for Approximate Nearest Neighbor Searching. http://www.cs.umd.edu/~mount/ANN/, Jan 2010.
19. B Nadler, S Lafon, R R Coifman, and I G Kevrekidis. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. *Advances in Neural Information Processing Systems 18*, 18(1):955–962, 2005.
20. David Nagel. Color Replacement in Photoshop CS. http://www.digitalmediadesigner.com/2004/01_jan/tutorials/pscs-cr040129.htm, 9 2004.
21. Fabio Pellacini and Jason Lawrence. AppWand. *ACM Transactions on Graphics*, 26(3):54, July 2007.
22. F. Pitie and A. Kokaram. The linear Monge-Kantorovitch linear colour mapping for example-based colour transfer. *Visual Media Production, 2007. IETCVMP. 4th European Conference on*, pages 1–9, 2007.
23. F. Pitie, A Kokaram, and R Dahyot. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1-2):123–137, July 2007.
24. E Reinhard, M. Adhikhmin, B Gooch, and P Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(4):34–41, 2001.

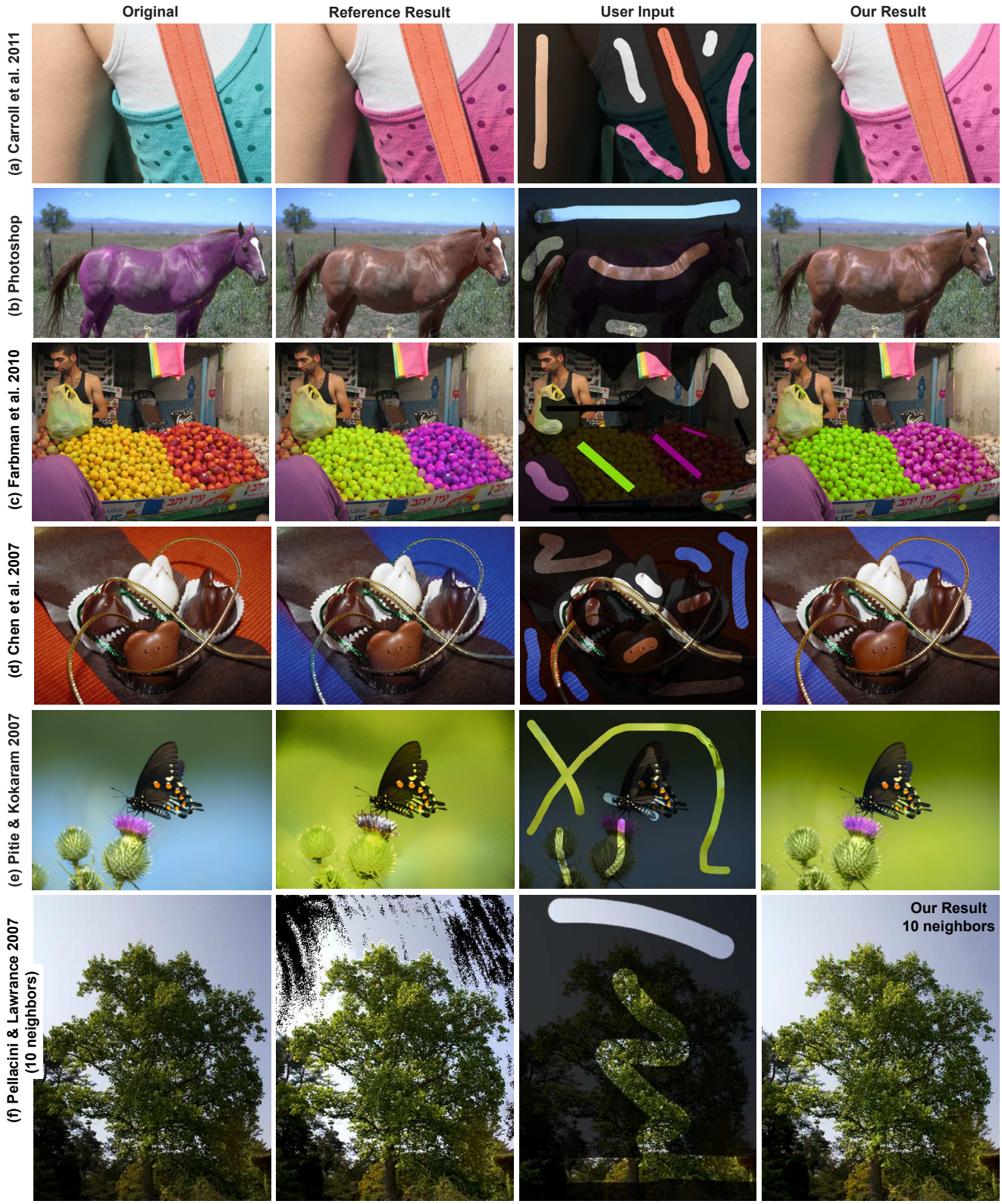

Fig. 13 Comparison to our results with those of other systems. Refer to Section 4.2 for the discussion of the particular results. Best seen in electronic version in close-up.

Fig. 14 Result of our re-coloring method. In each row, from left to right: original, user input in form of strokes, our output. All original images in this figure are copyrighted by Tom Ang (http://www.tomang.com/). Best seen in the electronic version in close-up.

Fig. 15 Our method can also be used to custom black-white conversion and it also allows selective conversion of spatial regions. Best seen in the electronic version in close-up.

25. S T Roweis and L K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science (New York, N.Y.)*, 290(5500):2323–6, December 2000.
26. Lawrence K. Saul and Sam T. Roweis. Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds. *Journal of Machine Learning Research*, 4(2):119–155, February 2004.
27. L. Shapira, Ariel Shamir, and Daniel Cohen-Or. Image Appearance Exploration by Model-Based Navigation. *Computer Graphics Forum*, 28(2):629–638, April 2009.
28. Maureen Stone. *A Field Guide to Digital Color*. A K Peters/CRC Press, 2003.
29. Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Local Color Transfer via Probabilistic Segmentation by Expectation-Maximization. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 747–754. IEEE, 2005.
30. Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Soft color segmentation and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1520–37, September 2007.
31. Baoyuan Wang, Yizhou Yu, Tien-Tsin Wong, Chun Chen, and Ying-Qing Xu. Data-driven image color theme enhancement. *ACM Transactions on Graphics*, 29(6):1, December 2010.
32. Baoyuan Wang, Yizhou Yu, and Ying-Qing Xu. Example-based image color and tone style enhancement. *ACM Transactions on Graphics*, 30(4):1, July 2011.
33. Jue Wang and Michael F. Cohen. Optimized Color Sampling for Robust Matting. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007.
34. Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. *ACM Transactions on Graphics*, 21(3):277, July 2002.
35. Chung-Lin Wen, Chang-Hsi Hsieh, Bing-Yu Chen, and Ming Ouhyoung. Example-based Multiple Local Color Transfer by Strokes. *Computer Graphics Forum*, 27(7):1765–1772, October 2008.
36. Xuezhong Xiao and Lizhuang Ma. Color transfer in correlated color space. *Virtual Reality Continuum And Its Applications*, page 305, 2006.
37. Xuezhong Xiao and Lizhuang Ma. Gradient-Preserving Color Transfer. *Computer Graphics Forum*, 28(7):1879–1886, October 2009.
38. Kun Xu, Yong Li, Tao Ju, Shi-Min Hu, and Tian-Qiang Liu. Efficient affinity-based edit propagation using K-D tree. *ACM Transactions on Graphics*, 28(5):1, December 2009.
39. Chuan-Kai Yang and Li-Kai Peng. Automatic mood-transferring between color images. *IEEE computer graphics and applications*, 28(2):52–61, March 2008.
40. L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*, 15(5):1120–1129, May 2006.

Przemyslaw Musialski received the PhD degree in computer science in 2010 from the Vienna University of Technology and the MSc degree in media systems in 2007 from the Bauhaus University Weimar. From 2007 to 2011 he was with VRVis Research Center in Vienna. From 2011 to 2012 he was postdoc at the Arizona State University. Since 2012 he is postdoc at Vienna University of Technology conducting research in interactive modeling and image processing.

Ming Cui received a Ph.D. form the Arizona State University (ASU) in 2010, a M.Sc. in Computer Science and a B.E. in Civil Engineering from Zhejiang University, Hangzhou, China in 2005 and 2002, respectively. Currently he is with Google, prior to that he worked at the ASU in Partnership for Research in Spatial Modeling lab (PRISM) from 2005. His research interests include computer graphics and image processing.

Jieping Ye received the Ph.D. degree in computer science from the University of Minnesota Twin Cities in 2005. He is associate professor in the Department of Computer Science and Engineering, Arizona State University. He has been a core faculty member of the Center for Evolutionary Medicine and Informatics, The Bio-design Institute, Arizona State University, since August 2005. His research interests include machine learning, data mining, and biomedical informatics. He received the NSF CAREER award in 2010.

Anshuman Razdan received a Ph.D. degree in Computer Science and a M.Sc. and B.S. degrees in Mechanical Engineering. He is Associate Professor in the Division of Computing Studies and the Director of Advanced Technology Innovation Collaboratory and the I3DEA Laboratory at Arizona State University, Polytechnic campus. His research interests include geometric design, computer graphics, document exploitation, and geospatial visualization and analysis. He is the principal investigator and a collaborator on several federal grants, including NSF, NGA, and NIH.

Peter Wonka received his Ph.D. and M.S. from the Vienna University of Technology in Computer Science and an M.S. in Urban Planning from the same institution. He is associate professor at the King Abdullah University of Science and Technology (KAUST) and Arizona State University. His research interests include various topics in computer graphics, visualization, and image processing.