ISSN 0924-9907, Volume 37, Number 3

J Math Imaging Vis (2010) 37: 220–231
DOI 10.1007/s10851-010-0204-8

Computing Color Transforms with Applications to Image Editing

Daniel Freedman · Pavel Kisilev

Published online: 19 May 2010

Abstract In this paper, we present a unified approach for the problem of computing color transforms, applications of which include shadow removal and object recoloring. We propose two algorithms for transforming colors. In the first algorithm, the detection of source and target regions is performed using a Bayesian classifier. Given these regions, the computed transform alters the color properties of the target region so as to closely resemble those of the source region. The proposed probabilistic formulation leads to a linear program (similar to the classic Transportation Problem), which computes the desired transformation between the target and source distributions. In the second algorithm, the detection and transformation steps are united into a single unified approach; furthermore, the continuity of the transformation arises more intrinsically within this algorithm. Both formulations allow the target region to acquire the properties of the source region, while at the same time retaining its own look and feel. Promising results are shown for a variety of applications.

Keywords Color transform · Transportation problem · Finite elements

1 Introduction

Recent years have witnessed great advances in digital photography, and with them the increasing accessibility of various image editing and manipulation tools. However, most of these tools, such as Photoshop, still require a skilled operator and are far from being automatic. In this paper, we propose a technique for computing a color transform between two objects within an image, or between two images. In our approach, source and target regions are specified, either automatically or semi-automatically, and the color properties of the target region are transformed so as to closely resemble those of the source region. We develop a unified scheme for these tasks based on the computation of a *flow* between target and source distributions. The approach is quite effective, and we show results on two applications: shadow removal and recoloring.

Before delving more deeply into details, we begin with a short review of prior work.

1.1 Prior Work

Many shadow detection and removal methods are based on the intrinsic image separation approach, wherein a single image is decomposed into its reflectance and illumination components (e.g., [1, 2]). In the algorithms employing the retinex theory [3], this separation is based solely on the assumption that large and sharp changes in lightness stem from the reflectance change, while small and slowly changing gradients are mainly due to the illumination changes. For example, in [4] the authors use the bilateral filter to separate small features, such as textures, from large features, such as strong edges. The approach requires various user inputs including the typical texture size.

In a different direction, a series of papers by Finlayson et al. ([5, 6], and references therein) exploits an approach based on a physical model, including Planckian light and Lambertian reflectance. The authors make the following simplifying assumptions: (i) the illumination is constant in the shadows region, (ii) the shadows have sharp edges, and (iii)

D. Freedman (✉) · P. Kisilev
Hewlett-Packard Laboratories, Haifa, Israel
e-mail: daniel.freedman@hp.com

P. Kisilev
e-mail: pavel.kisilev@hp.com

the reflectance does not change across the shadow boundaries. These assumptions are valid only in very specific settings.

Another set of techniques is based on the *matting* approach, where an observed image is modeled as a linear combination of fully lit and fully shadowed counterparts [7]. Further development of the matting approach is presented in [8], where the authors propose a shadow formation model in which the lit intensity at a given pixel is an affine function of the shadowed intensity. The illuminated pixel color is recovered by estimating the four parameters of the affine model, based only on the mean colors of the pixels in the lit and shadowed regions, and on standard deviations of their corresponding luminances.

Recoloring methods have developed separately from those dealing with shadow removal. Many recoloring papers focus on the problem of transferring color to gray scale images, and vice versa (e.g., [9]). In [10], the authors propose a recoloring method that is based on alpha matting and compositing, and a color transformation algorithm. The transformation function is postulated to be single-valued and monotonically increasing in the destination pixel intensity domain. The authors estimate the cumulative distribution function (cdf) of the destination region in an image as simply a scaled version of the cdf of the source region. Finally, the compositing operation is implemented using the alpha matte equation. In [11], the authors propose a natural colorization approach which is based on the assumption that pixels having similar intensities should have similar color. In this paper, the input image (or sequence) is gray scale and the output image is a colored image. In [12], the authors present a method for recoloring a destination image according to the color scheme from the source image. The image is first segmented into groups of pixels with similar color; then, the color palette for an image is constructed by choosing the most typical colors from the above segments. Color transfer is computed by matching the segment areas between the source and destination segments, and finding the closest Euclidean distance match between pairs of colors from the source and destination segments. In [13], the authors apply a linear transformation that scales the mean and the variance of the target area according to the ones from the source area. In [14], the authors use the approach in [13] along with a simple segmentation technique to perform recoloring operations. In [15], the authors present generic machinery for transforming probability distributions, which can be applied to the problem to the problem of color transfer.

Somewhat related work in [16] describes a generic interpolation machinery based on solving Poisson equations. This approach belongs to the gradient domain methods. The authors present various editing applications such as object importation from one region (or an image) to another, seamless cloning of textures, and others.

Note that a preliminary version of this paper appeared in [17]. This earlier version of the paper, however, only presented the cruder algorithm contained in Sect. 3. The algorithm described in Sect. 4, which is much more mathematically elegant, is entirely new to this paper.

1.2 Paper Outline

The remainder of this paper is organized as follows. In Sect. 2, we outline the problem of computing color transforms, and divide the problem into two stages: detection and learning of the color transform. In Sect. 3, we present a first pass for the color transform problem, in which separate algorithms are proposed for the detection and transformation stages. In Sect. 4, we present a second pass for the color transform problem, in which the detection and transformation stages are unified, and the algorithm is inherently continuous. In Sect. 5, we present results of applying the algorithm to the problems of shadow removal and object recoloring. Section 6 concludes.

2 The Problem

The problem we are interested is transfer of color properties from one object in an image to another. More specifically, given some knowledge of the source and target properties, we wish to accomplish two tasks:

1. **Detection:** Find the regions of the image which correspond to the source and target properties. These regions constitute the source and target objects, respectively.
2. **Photometric Transform:** Transform the color properties of the target object to resemble those of the source object, in perceptually meaningful way.

Before posing the problem in a more formal fashion, we discuss two applications of this framework. The first is shadow removal. In this problem, both the source and the target derive from the same material or object; however, the source is lit, while the target is in shadow. The second application is object recoloring. Here, the goal is to simply take a given object, and to transfer its colors to a second object. Often the two objects are of the same type (such as two cars), but this need not be the case. Both shadow removal and object recoloring are desirable operations for many computer graphics applications, as well as in the context of Photoshop-type software.

Let us now turn to a more formal statement of the problem. Let color be denoted by the vector $c \in \mathscr{C} = \mathbb{R}^d$. The color c could be represented in Lab space, RGB space, or any of many other possibilities. In principle, the techniques presented in this paper could also be applied to learning texture transforms, where texture is represented, for example, as the output of a filter bank. Let the image domain be

$\mathscr{X} \subset \mathbb{R}^2$, so that an image is given by $c : \mathscr{X} \to \mathscr{C}$. Then our two goals may be restated as follows:

1. **Detection:** Find two subsets of the image domain $\mathscr{X}$, the source region $\mathscr{S}$ and the target region $\mathscr{T}$, with $\mathscr{S} \cap \mathscr{T} = \emptyset$.
2. **Photometric Transform:** For each pixel $x \in \mathscr{T}$, compute a mapping $c(x) \to \Phi(c(x))$ such that the collection $\{\Phi(c(x)) : x \in \mathscr{T}\}$ is in some perceptual sense similar to the collection $\{c(x) : x \in \mathscr{S}\}$.

In Sect. 3, we begin by treating each of these problems independently. In Sect. 4, we then propose a unified algorithm to treat both detection and color transformation simultaneously. The second algorithm is also more inherently continuous than the first.

3 The Color Transform: First Pass

In this section, we present the algorithm for computing color transforms. Although logic would dictate that we begin our exposition with the detection of the source and target regions, we begin instead with a discussion of the color transform algorithm, which allows the detected target region to be transformed into something resembling the detected source region. This is done in order to emphasize the color transform algorithm, which is the more important contribution. The basis for the algorithm is a version of the Transportation Problem [18, 19], which can be posed as a linear program; this optimization, combined with an appropriate interpolation scheme, yields an effective technique for computing color transforms. We then move on to a discussion of the Detection Algorithm, including an examination of the form of user-input required.

3.1 The Color Transform

We begin with the main algorithmic contribution of this section: the transformation of the color properties of the target region so that they closely resemble those of the source region. To repeat our earlier formulation of this problem: for each pixel $x \in \mathscr{T}$, we wish to compute a mapping $c(x) \to \Phi(c(x))$ such that the collection $\{\Phi(c(x)) : x \in \mathscr{T}\}$ is in some sense similar to the collection $\{c(x) : x \in \mathscr{S}\}$. This is not a straightforward problem, as the two collections may be quite different. For example, probability distributions over the source and target pixels (i.e. over their colors) may have different numbers of modes, different shapes, and so on.

Our solution to this problem is to use the classic Transportation Problem to compute the transformation between the two distributions. In the Transportation Problem [18, 19], the goal is to match supplies of certain quantities with demands for these quantities, bearing in mind the absolute amounts of both supplies and demands. This is computed via a *flow* between supplied quantities and those demanded. In what follows, we make the analogy between supplied quantities with the target region, and demanded quantities with the source region. The flow thus computed by the solution to the Transportation Problem is exactly what we are after: it allows us to transform target pixels into source-like pixels.

Before turning to a more formal statement of the problem, it is of interest to note that the Transportation Problem has been used in computer vision applications before, most notably for the computation of the Earth Mover's Distance (EMD) [20]. Indeed, a by-product of the EMD computation is the flow that will be useful to us; however, in the context of EMD the flow is nothing more than a by-product, and is used only to facilitate the computation of the EMD metric. By contrast, in the development of our algorithm, the flow itself plays a critical role. It is also worth noting that a few works have used somewhat related ideas [21], albeit in the continuous setting and for different applications, such as registration.

Let us begin by fixing notation. The labels for source and target objects will be s and t respectively, and when speaking of an object defined for both, we will use $\alpha \in \{s, t\}$. We are given source and target probability distributions, which are learned as part of the Detection Algorithm (see Sect. 3.2). As usual, there are a variety of ways of representing distributions. In this section, we use simple histograms; we move on to more sophisticated representations in Sect. 4. We represent the source and target distributions compactly as a list of histogram bins with non-zero probability, i.e. $\{(\bar{c}_i^\alpha, \bar{p}_i^\alpha)\}_{i=1}^{n_\alpha}$, where $\bar{c}_i^\alpha$ is a bin-center, $\bar{p}_i^\alpha$ is the corresponding probability mass for that bin, and n_α is the number of such bins. Finally, for a given color c, let $[c]^\alpha$ be the bin in which it resides (where again $\alpha \in \{s, t\}$).

Given the above notation, we can now turn to the problem of computing the color transform, for which we use the Transportation Problem. Recall that the Transportation Problem is formulated as follows [18, 19]: let the *flow* between the target and source distributions be given by f_{ij}, where the indices i and j range over the (non-empty) bins of the target and source distributions, respectively. That is, f_{ij} can be thought of as the part of target bin i which is mapped to source bin j. Now, let the *color distance* between two colors[1] be given by $D(c_1, c_2)$. As we shall see in the sequel, while this distance is sometimes taken as the ordinary Euclidean distance, there are times when other choices are more appropriate. In any case, taking the color distance as given for the moment, we would like to solve the following

[1] In the work on the Earth Mover's Distance, D is generally called the *ground distance*.

optimization:

$$\min_{\{f_{ij}\}} \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} f_{ij} D(\bar{c}_i^t, \bar{c}_j^s)$$

$$\text{subject to} \quad \sum_{j=1}^{n_s} f_{ij} = \bar{p}_i^t, \quad i = 1,\ldots,n_t$$

$$\sum_{i=1}^{n_t} f_{ij} = \bar{p}_j^s, \quad j = 1,\ldots,n_s$$

The goal of the objective function is to map the target colors $\bar{c}_i^t$ to corresponding source colors $\bar{c}_j^s$ in such a way that the color distance between them is as small as possible. However, we cannot reasonably expect that each bin of the target distribution maps neatly to exactly one bin of the source distribution. Thus, we allow target bins to be spread over several source bins, subject to the two constraints of the original Transportation Problem, which ensure "conservation of probability" for both the target and source distributions.

In fact, in our case requiring conservation of probability is too extreme[2] as it assumes that the source and target regions contain exactly the same amounts of "comparable colors." As a result, we modify the optimization as follows:

$$\begin{aligned} &\min_{\{f_{ij}\}} \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} f_{ij} D(\bar{c}_i^t, \bar{c}_j^s) \\ &\text{subject to} \quad \eta^{-1}\bar{p}_i^t \le \sum_{j=1}^{n_s} f_{ij} \le \eta \bar{p}_i^t, \quad i = 1,\ldots,n_t \\ &\qquad\qquad \eta^{-1}\bar{p}_j^s \le \sum_{i=1}^{n_t} f_{ij} \le \eta \bar{p}_j^s, \quad j = 1,\ldots,n_s \\ &\qquad\qquad \sum_{i,j} f_{ij} = 1 \end{aligned} \tag{1}$$

where the final constraint, which was enforced implicitly in the original Transportation Problem, is now made explicit. $\eta \ge 1$ is a parameter which describes the slackness of the conservation constraints; typically, we choose $\eta \approx 3$.

Now, given the solution to the modified Transportation Problem, the problem of computing the color transform is effectively solved. In particular, if we were interested only in mapping the bin centers $\bar{c}_i^t$, we have the following color transform rule:

$$\bar{c}_i^t \to \frac{\sum_{j=1}^{n_s} f_{ij}\bar{c}_j^s}{\sum_{j=1}^{n_s} f_{ij}} \equiv \Phi(\bar{c}_i^t) \tag{2}$$

That is, we use the flow to average over the source bin centers in the natural way, and then normalize. Notice that $\sum_j f_{ij} \le \eta \bar{p}_i^t \ll 1$, so that the normalization is crucial.

Note that the color transform Φ is defined only for the bin-centers $\bar{c}_i^t$; however, we wish to transform not just the bin centers of the target distribution, but all of the pixels in the target region $\mathcal{T}$. To achieve this, the simplest option is to map a pixel $c(x)$ in $\mathcal{T}$ to its corresponding bin $[c(x)]^t$, and then to use the transformation rule in (2) on the binned value. Predictably, however, this leads to binning artifacts; two colors which are quite close may in fact lie in different bins, and therefore be mapped to quite different values. Instead, we use the transformation rule given in (2) in combination with a simple interpolation scheme.

To wit, consider a pixel in $\mathcal{T}$ with color c (we drop the argument x to simplify notation), with histogram bin $[c]^t$. Now, given a bin i in the target distribution, then let $\mathcal{N}_i$ be the union of the bin itself, as well as the neighboring (non-empty) bins within the histogram; for example, if the histogram is 2-dimensional, and we use a standard 4-neighborhood, then $\mathcal{N}_i$ would contain at most 5 elements. (In general, for a d-dimensional histogram, we use a $2d$-neighborhood, so that $\mathcal{N}_i$ contains at most $2d+1$ elements.) For each bin i in the neighborhood of c, i.e. in $\mathcal{N}_{[c]^t}$, we compute a weight based on the distance $D(c, \bar{c}_i^t)$ between c and the center of bin i, $\bar{c}_i^t$. In particular, the weights are given by

$$w_i(c) = \frac{\xi(D(c, \bar{c}_i^t))}{\sum_{j\in\mathcal{N}_{[c]^t}} \xi(D(c, \bar{c}_j^t))}$$

where ξ satisfies $\xi'(\cdot) < 0$ and $\xi(0) = \infty$; the latter property ensures that the scheme is truly interpolatory, rather than an approximation scheme. We choose $\xi(d) = d^{-1}$, though other choices are possible. In this case, the final color transform rule is given by

$$\Phi(c) = \sum_{i\in\mathcal{N}_{[c]^t}} w_i(c)\Phi(\bar{c}_i^t) \tag{3}$$

3.2 Detection

In order to detect the source and target regions, we must have some prior knowledge of both the source and target images. To begin with, let us assume that we have probability densities over c for both the source and target, i.e. $\rho_s(c) = \rho(c|s)$ and $\rho_t(c) = \rho(c|t)$. Let us further assume a uniform distribution over the regions that are neither source nor target, i.e. $\rho_n(c) = \rho(c|n) = \theta$, where θ is a constant chosen so that

[2] For example, suppose that the source image is 50% light red and 50% dark red, and the target image is 40% light blue and 60% dark blue; assume further that the light colors have matching L values, as do the dark values, and that our color distance is simply the absolute value of the difference in L. Then the full conservation of probability will require coloring part (10%) of the *dark* blue section of the target image in *light* red, which is obviously not desirable.

$\rho_n(c)$ integrates to 1. Now, the standard Bayesian classifier will classify a value of c as belonging to the source if

$$\rho(s|c) > \max\{\rho(t|c), \rho(n|c)\}$$

(Note that if there is equality, we are on the boundary of at least two classes.)

Now, from Bayes' Rule we have that $\rho(s|c) = \rho(c|s) \times P(s)/\rho(c)$, where $P(s)$ is the probability that a given pixel belongs to the source. Assuming, in the absence of other knowledge, that $P(s) = P(t) = P$ and $P(n) = 1 - 2P$, then the Bayesian classifier becomes:

- Choose $x \in \mathscr{S}$ if $\rho_s(c(x)) > \max\{\rho_t(c(x)), \theta'\}$
- Choose $x \in \mathscr{T}$ if $\rho_t(c(x)) > \max\{\rho_s(c(x)), \theta'\}$
- Choose x as neither source nor target in all other cases

where $\theta' = (1-2P)\theta/P$. It is clear, therefore, that given the source and target densities $\rho_s(c)$ and $\rho_t(c)$, we have a simple classifier that depends only on the choice of the single parameter θ'.

The next question, then, concerns the origin of the source and target densities. In certain applications, it is possible that these densities are known *a priori*; for example, one might use any of a number of schemes to learn the color density of blue skies, grass, or skin (e.g., [22]). However, assuming that this is not the case, we use two variants of a semi-automatic scheme. In the simpler variant, used for recoloring or for shadow removal, the user simply chooses two rectangles, one which surrounds source pixels and the other target pixels. Based on these rectangles, we then compute histograms for each of the source and target in the relevant space—in our case, Lab color. Note that the histogram bin size in some sense determines the extent to which we extrapolate from the information conveyed by the pixels within the user-chosen rectangles.

The second variant of the semi-automatic scheme is more complex, and may be used for shadow removal. In this case, the user chooses a single rectangle, which encompasses both lit and shadowed pixels. The pixels, which are represented in Lab space, are then divided by performing a k-means clustering, with $k = 2$, on the L-channel. Then, corresponding source and target distributions are easily computed. This heuristic works very well in practice.

4 The Color Transform: Second Pass

In this section, we outline a second version of the color transformation algorithm. The advantages of this second version over the first version presented in Sect. 3 are twofold:

- The color transformation is inherently continuous. Whereas the flow computation presented in Sect. 3 was discrete, and the color transformation was extended to a continuous one in an essentially ad hoc manner, the flow computation presented here is continuous from the beginning.
- Detection is included within the color transformation itself. Rather than breaking the problem up into detection and transformation steps, the second version of the algorithm treats detection and transformation simultaneously. This is both more elegant mathematically and potentially more efficient.

We begin by outlining the basic continuous formulation. We then show how to convert this into a problem amenable to solution by computer using a finite element representation. This yields a finite dimensional optimization problem, but with an infinite number of constraints. We subsequently derive a sufficient set of conditions for the constraints to hold; these sufficient conditions consist of a finite number of constraints, leading to a traditional finite-dimensional optimization problem. Finally, we show how to compute a key quantity used by the algorithm in an efficient manner.

4.1 The Basic Continuous Formulation

The modified transportation problem in (1) can be naturally replaced by the following continuous version.

$$\min_{\{f(\cdot,\cdot)\}} \iint f(c^t, c^s) D(c^t, c^s) dc^t dc^s \tag{4a}$$

$$\text{subject to} \quad \eta^{-1} p^t(c^t) \le \int f(c^t, c^s) dc^s \le \eta p^t(c^t), \quad \forall c^t \tag{4b}$$

$$\eta^{-1} p^s(c^s) \le \int f(c^t, c^s) dc^t \le \eta p^s(c^s), \quad \forall c^s \tag{4c}$$

$$\iint f(c^t, c^s) dc^t dc^s = 1 \tag{4d}$$

$$f(c^t, c^s) \ge 0, \quad \forall c^t, c^s \tag{4e}$$

where the continuous flow $f(c^t, c^s)$ has replaced the discrete flow f_{ij}; and $p^t(c^t)$, $p^s(c^s)$ are continuous versions of the target and source densities, respectively.

Given the flow f, the color transformation can be computed as

$$\Phi(c^t) = \frac{\int f(c^t, c^s) c^s dc^s}{\int f(c^t, c^s) dc^s} \tag{5}$$

in direct analogy with (2).

In general, some numerical scheme must be adopted so that this infinite-dimensional continuous problem can be solved by computer. In what follows, we will use a finite element representation of the relevant continuous objects, i.e. f, p^t, and p^s. The problem will then become finite-dimensional once again, and therefore amenable to a computational solution.

4.2 Finite Element Representations

We will begin by specifying finite element representations for the probability densities for both target and source. The basic specification is that of a Kernel Density Estimate (KDE), which we write in the usual form,

$$p^t(c^t) = \frac{1}{n_t}\sum_{i=1}^{n_t} K(c^t - c_i^t),$$

$$p^s(c^s) = \frac{1}{n_s}\sum_{j=1}^{n_s} K(c^s - c_j^s)$$

where K is a kernel. Kernels may have compact support, such as the uniform or Epanechnikov kernels, or may have infinite support, such as the Gaussian kernel.[3] Kernels will generally have a bandwidth parameter; we suppress this parameter for notational convenience, but the bandwidth will need to be set in any implementation. The collections $\{c_i^t\}$ and $\{c_j^s\}$ can be simply the pixels selected by the user in the first step of the algorithm. Or, if one prefers a more compact representation, then one can use a combined histogram-KDE style representation, as in [23].

One of our two main goals is incorporation of the detection process within the color transformation itself. First, from now on we denote by c^t any pixel in the image, which *might* be part of the target, or might not. That is, c^t is the color of a pixel which might or might not be modified. It is the job of the detection algorithm—which will be built in to the transformation algorithm—to determine whether or not the pixel is to be modified.

Given this, we can emend the target density as follows:

$$\begin{aligned} p^t(c^t) &= \text{prob}(c^t \in \text{frg})\, p(c^t | c^t \in \text{frg}) \\ &\quad + \text{prob}(c^t \in \text{bkg})\, p(c^t | c^t \in \text{bkg}) \\ &= \xi_t \frac{1}{n_t}\sum_{i=1}^{n_t} K(c^t - c_i^t) + (1-\xi_t)\, p_{\text{bkg}}^t(c^t) \end{aligned} \tag{6}$$

where frg denotes the foreground (i.e. the target), and bkg denotes the background. The idea behind this modification is straightforward. If we look at the target density resulting from pixels in the entire image, the pixels may either be drawn from the true target region, with probability ξ_t, in which case the density is given by the original KDE; or they may be drawn from the background, with probability $1-\xi_t$, in which case the density is given by the background density $p_{\text{bkg}}^t(c^t)$. Without any extra information on the background, we take the background density to the uninformative or uniform density, i.e. $p_{\text{bkg}}^t(c^t) = V^{-1}$, where V is the volume of the color space. We may similarly rewrite the source density as

$$p^s(c^s) = \xi_s \frac{1}{n_s}\sum_{j=1}^{n_s} K(c^s - c_j^s) + (1-\xi_s)\, p_{\text{bkg}}^s(c^s) \tag{7}$$

Now, the final object that needs a finite element representation is the object we are optimizing over, that is the flow f itself. Given the original KDE representation of the target and source densities, a natural finite element representation for the flow is

$$f(c^t, c^s) = \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s)$$

In this case, the optimization over the flow will turn into an optimization over the variables β_{ij}. However, due to our desire to incorporate detection directly into the color transformation, and hence our use of the modified densities in (6) and (7), we modify the flow slightly as well:

$$f(c^t, c^s) = \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) + \beta_0 \delta(c^t - c^s) \tag{8}$$

where $\delta(\cdot)$ is the usual delta-function. Thus, the flow is specified by the $n_t n_s + 1$ variables β_{ij} and β_0.

We may substitute this representation for the flow into the color transformation given in (5). We have that

$$\int f(c^t, c^s) dc^s = \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) + \beta_0$$

using the fact that each kernel integrates to 1. We also have that

$$\int f(c^t, c^s) c^s dc^s = \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) c_j^s + \beta_0 c^t$$

using the sampling property of the δ-function and the fact that $\int K(x-y)x\,dx = y$ when K is symmetric. Thus, the color transformation becomes

$$\Phi(c^t) = \frac{\sum_{i=1}^{n_t}\sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) c_j^s + \beta_0 c^t}{\sum_{i=1}^{n_t}\sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) + \beta_0} \tag{9}$$

4.3 The Finite Elements Color Transformation Problem

Given the finite element representations for the target and source densities and the flow, our goal is to convert the functional optimization in (4) to an ordinary mathematical program in which the unknowns are vectors, i.e. the β variables.

[3] Of course, in practice the Gaussian kernel is truncated, so that it effectively has compact support.

We begin by converting the objective function to a function of vectors, and then treat each of the constraints in turn. Given the finite element representation (8) for the flow, the objective function $T[f]$ becomes

$$\begin{aligned} T[f] &= \iint f(c^t, c^s) D(c^t, c^s) dc^t dc^s \\ &= \iint \left[\sum_{i,j} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) \right. \\ &\quad \left. + \beta_0 \delta(c^t - c^s) \right] D(c^t, c^s) dc^t dc^s \\ &= \sum_{i,j} \tilde{D}_{ij} \beta_{ij} + \beta_0 \int D(c^t, c^t) dc^t \\ &= \sum_{i,j} \tilde{D}_{ij} \beta_{ij} \end{aligned}$$

where in the third line, we have defined

$$\tilde{D}_{ij} = \iint K(c^t - c_i^t) K(c^s - c_j^s) D(c^t, c^s) dc^t dc^s \tag{10}$$

and in fourth line we have used the fact that $D(c^t, c^t) = 0$.

Next we turn to each of the constraints. The right-hand inequality of constraint (4b) becomes

$$\begin{aligned} &\int f(c^t, c^s) dc^s \le \eta p^t(c^t) \quad \Rightarrow \\ &\int \left[\sum_{i,j} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) + \beta_0 \delta(c^t - c^s) \right] dc^s \\ &\quad \le \eta \xi_t \frac{1}{n_t} \sum_i K(c^t - c_i^t) + \eta(1 - \xi_t) V^{-1} \quad \Rightarrow \\ &\sum_i \left(\sum_j \beta_{ij} - \eta \frac{\xi_t}{n_t} \right) K(c^t - c_i^t) + \beta_0 - \eta(1 - \xi_t) V^{-1} \le 0 \end{aligned}$$

where in the last line, we have used the fact the kernel K integrates to 1. Similarly, the left-hand inequality is

$$\sum_i \left(\sum_j \beta_{ij} - \frac{1}{\eta} \frac{\xi_t}{n_t} \right) K(c^t - c_i^t) + \beta_0 - \frac{1}{\eta}(1 - \xi_t) V^{-1} \ge 0$$

In direct analogy, we may rewrite the two inequalities for constraint (4c) as

$$\sum_j \left(\sum_i \beta_{ij} - \eta \frac{\xi_s}{n_s} \right) K(c^s - c_j^s) + \beta_0 - \eta(1 - \xi_s) V^{-1} \le 0$$

and

$$\sum_j \left(\sum_i \beta_{ij} - \frac{1}{\eta} \frac{\xi_s}{n_s} \right) K(c^s - c_j^s) + \beta_0 - \frac{1}{\eta}(1 - \xi_s) V^{-1} \ge 0$$

The overall conservation of probability constraint (4d) may be rewritten as

$$\begin{aligned} &\iint f(c^t, c^s) dc^t dc^s = 1 \quad \Rightarrow \\ &\iint \left(\sum_{i,j} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) \right. \\ &\quad \left. + \beta_0 \delta(c^t - c^s) \right) dc^t dc^s = 1 \quad \Rightarrow \\ &\sum_{i,j} \beta_{i,j} + \beta_0 V = 1 \end{aligned}$$

where again, V is the volume of the color space.

Finally, the non-negativity condition (4e) on the flow reduces to

$$\begin{aligned} &\sum_{i,j} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) + \beta_0 \delta(c^t - c^s) \ge 0 \quad \Rightarrow \\ &\sum_{i,j} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) \ge 0 \end{aligned}$$

where we have used the fact that the δ-function is 0 almost everywhere.

Thus, we may rewrite the continuous optimization problem (4) as

$$\min_{\{\beta_{ij}\}, \beta_0} \sum_{i,j} \tilde{D}_{ij} \beta_{ij}$$

subject to:

$$\begin{aligned} &\sum_i \left(\sum_j \beta_{ij} - \eta \frac{\xi_t}{n_t} \right) K(c^t - c_i^t) + \beta_0 \\ &\quad - \eta(1 - \xi_t) V^{-1} \le 0, \quad \forall c^t \\ &\sum_i \left(\sum_j \beta_{ij} - \frac{1}{\eta} \frac{\xi_t}{n_t} \right) K(c^t - c_i^t) + \beta_0 \\ &\quad - \frac{1}{\eta}(1 - \xi_t) V^{-1} \ge 0, \quad \forall c^t \\ &\sum_j \left(\sum_i \beta_{ij} - \eta \frac{\xi_s}{n_s} \right) K(c^s - c_j^s) + \beta_0 \\ &\quad - \eta(1 - \xi_s) V^{-1} \le 0, \quad \forall c^s \\ &\sum_j \left(\sum_i \beta_{ij} - \frac{1}{\eta} \frac{\xi_s}{n_s} \right) K(c^s - c_j^s) + \beta_0 \\ &\quad - \frac{1}{\eta}(1 - \xi_s) V^{-1} \ge 0, \quad \forall c^s \end{aligned} \tag{11}$$

$$\sum_{i,j} \beta_{i,j} + \beta_0 V = 1$$

$$\sum_{i,j} \beta_{ij} K(c^t - c_i^t) K(c^s - c_j^s) \geq 0, \quad \forall c^t, c^s$$

The result is a finite-dimensional optimization problem; that is, the optimizing variables, β_{ij} and β, are finite in number. However, there are an infinite number of constraints, as evidenced by the $\forall c^t$ and $\forall c^s$ notation. In the following section, we show how to derive a finite number of sufficient conditions which ensure that the constraints are satisfied.

4.4 The Sufficient Conditions

Consider the first constraint of the optimization problem (11):

$$\sum_i \left(\sum_j \beta_{ij} - \eta \frac{\xi_t}{n_t} \right) K(c^t - c_i^t) + \beta_0 - \eta(1 - \xi_t) V^{-1} \leq 0, \quad \forall c^t$$

A sufficient condition can be derived by producing a separate inequality for the coefficients of each $K(c^t - c_i^t)$ term, as well as a single inequality for the constant terms. That is,

$$\sum_j \beta_{ij} \leq \eta \xi_t \frac{1}{n_t}, \quad i = 1, \ldots, n_t$$

$$\beta_0 \leq \eta(1 - \xi_t) V^{-1}$$

are sufficient conditions for the first constraint of (11) to hold. Similar sufficient conditions may be derived for the second, third, and fourth constraints of (11).

The fifth constraint of (11) does not need any modification, which leaves the sixth and final constraint. Once again, a sufficient condition may be derived by considering the inequality for the coefficients of each $K(c^t - c_i^t) K(c^s - c_j^s)$ term separately. This leads to $\beta_{ij} \geq 0$ for all i, j.

Using the sufficient conditions, we have the following optimization:

$$\min_{\{\beta_{ij}\}, \beta_0} \sum_{i,j} \tilde{D}_{ij} \beta_{ij}$$

subject to:

$$\begin{aligned}
&\eta^{-1} \xi_t \frac{1}{n_t} \leq \sum_j \beta_{ij} \leq \eta \xi_t \frac{1}{n_t}, \quad i = 1, \ldots, n_t \\
&\eta^{-1} \xi_s \frac{1}{n_s} \leq \sum_i \beta_{ij} \leq \eta \xi_s \frac{1}{n_s}, \quad j = 1, \ldots, n_s \\
&\eta^{-1}(1 - \min\{\xi_s, \xi_t\}) V^{-1} \leq \beta_0 \leq \eta(1 - \max\{\xi_s, \xi_t\}) V^{-1}
\end{aligned} \tag{12}$$

$$\sum_{i,j} \beta_{i,j} + \beta_0 V = 1$$

$$\beta_{ij} \geq 0, \quad i = 1, \ldots, n_t, \ j = 1, \ldots, n_s$$

Since these constraints are sufficient, then any set of β's which satisfies them will be feasible for the original program (11). Thus, minimizing the objective function subject to these sufficient conditions, which by inspection can be achieved via linear programming, will typically yield a suboptimal solution.

When will the solution using these constraints be optimal? When the conditions are not only sufficient, but necessary. This will be the case, for example, if for all i, there exists a $\breve{c}_i^t$ such that $K(\breve{c}_i^t - c_{i'}^t) > 0$ only if $i' = i$, and there exists a $\breve{c}$ such that $K(\breve{c} - c_i^t) = 0$ for all i. We must also have an equivalent condition on the source colors c_j^s. Note that these conditions are exactly fulfilled if we take the kernel K to be δ-functions, which brings us back to the original discrete formulation of Sect. 3.

Nonetheless, we would expect the solution to be relatively close to optimal, as the overlap between kernels will in general not be extreme. As a rule of thumb, we would expect the kernel of a given point to overlap with $O(1)$ other points.[4] As a result, we will be close to the situation in which the above sufficient conditions are also necessary.

4.5 The Transform Φ when $\eta = 1$

In the special case that $\eta = 1$, that is, that probability is conserved, the color transform Φ takes on a particularly nice form. The color transform was given in (9) as

$$\Phi(c^t) = \frac{\sum_{i=1}^{n_t} \sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) c_j^s + \beta_0 c^t}{\sum_{i=1}^{n_t} \sum_{j=1}^{n_s} \beta_{ij} K(c^t - c_i^t) + \beta_0}$$

In the case that $\eta = 1$, the first constraint of (12) reduces to

$$\sum_j \beta_{ij} = \frac{\xi}{n_t}, \quad i = 1, \ldots, n_t$$

where for simplicity, we have also taken $\xi_t = \xi_s = \xi$. The third constraint of (12) becomes

$$\beta_0 = (1 - \xi) V^{-1}$$

Now, for convenience, let us define

$$\hat{c}_i^t = \frac{\sum_j \beta_{ij} c_j^s}{\sum_j \beta_{ij}}$$

$\hat{c}_i^t$ is very similar to the discrete version of the color transform, see Sect. 3. Then substituting these expressions into

[4] Of course, if the bandwidth of the kernel is chosen to be very large, a given point could overlap with $O(n)$ other points. This is not the situation one typically encounters in practice, as users of Kernel Density Estimates and/or the Mean Shift algorithm will readily recognize.

the expression for Φ gives

$$\begin{aligned}\Phi(c^t) &= \frac{\sum_i K(c^t - c_i^t)\sum_j \beta_{ij} c_j^s + \beta_0 c^t}{\sum_i K(c^t - c_i^t)\sum_j \beta_{ij} + \beta_0} \\ &= \frac{\sum_i K(c^t - c_i^t)\hat{c}_i^t(\sum_j \beta_{ij}) + (1-\xi)V^{-1}c^t}{\sum_i K(c^t - c_i^t)\frac{\xi}{n_t} + (1-\xi)V^{-1}} \\ &= \frac{\xi\frac{1}{n_t}\sum_i K(c^t - c_i^t)\hat{c}_i^t + (1-\xi)V^{-1}c^t}{\xi\frac{1}{n_t}\sum_i K(c^t - c_i^t) + (1-\xi)V^{-1}}\end{aligned}$$

This last expression makes clear both aspects of the color transform: its continuous nature, and the fact that the detection process is built in. To see the continuous nature, suppose that c^t is such that $K(c^t - c_i^t) > 0$ for a small set I of indices i, and that $K(c^t - c_i^t) \gg V^{-1}$ for all such $i \in I$. Then the transformation can be written

$$\Phi(c^t) \approx \sum_{i\in I} w_i(c^t)\hat{c}_i^t$$

where $w_i(c^t) = K(c^t - c_i^t)/\sum_{i'\in I} K(c^t - c_{i'}^t)$. To see the built in detection process, consider the case when $K(c^t - c_i^t) \approx 0$ for all i, so that $K(c^t - c_i^t) \ll V^{-1}$. In this case,

$$\Phi(c^t) \approx \frac{(1-\xi)V^{-1}c^t}{(1-\xi)V^{-1}} = c^t$$

that is, the pixel is left alone, as desired. The actual expression for Φ interpolates smoothly between these cases.

4.6 Efficient Computation of $\tilde{D}_{ij}$

The one remaining item is the computation of the integrals

$$\tilde{D}_{ij} = \iint K(c^t - c_i^t)K(c^s - c_j^s)D(c^t, c^s)dc^t dc^s$$

Note that we will need to compute $n_t n_s$ such integrals; as a result, a method which could speed up the computation is to be preferred.

Let us begin by defining

$$\Gamma(x^t, x^s) = \iint K(c^t - x^t)K(c^s - x^s)D(c^t, c^s)dc^t dc^s$$

so that $\tilde{D}_{ij} = \Gamma(c_i^t, c_j^s)$. Further, we take $D(c^t, c^s) = E(c^t - c^s)$, as will be the case in all of the examples we consider. For example, we might have $E(c^t - c^s) = \|c^t - c^s\|_2$. Then we may rewrite the equation for Γ as

$$\begin{aligned}\Gamma(x^t, x^s) &= \iint E(c^t - c^s)K(c^t - x^t)K(c^s - x^s)dc^t dc^s \\ &= \int\left[\int E(c^t - c^s)K(x^t - c^t)dc^t\right] \\ &\quad \times K(x^s - c^s)dc^s \\ &= \int\left[\int E(y)K((x^t - c^s) - y)dy\right] \\ &\quad \times K(x^s - c^s)dc^s \\ &= \int (E * K)(x^t - c^s)K(x^s - c^s)dc^s \\ &= \int (E * K)((x^t - x^s) - z)K(z)dz \\ &= (E * K * K)(x^t - x^s)\end{aligned}$$

In the second line, we have used the symmetry of the kernels; in the third line, we have substituted variables, with $y = c^t - c^s$; in the fifth line, we have substituted variables, with $z = c^s - x^s$, and again used the symmetry of K; and $*$ indicates convolution.

In summary, we can compute the entire function Γ as a result of two convolutions. As usual, we can compute these convolutions quickly by transforming to the Fourier domain, and using FFTs. Once Γ has been computed, the $\tilde{D}_{ij}$ are simply computed by sampling the function Γ at the appropriate places.

4.7 Implementation Details

In order to solve the color transformation problem, we must solve either of the two optimization problems, (1) or (12). In both cases, the optimization is a linear program, for which many solvers exist. We use the MATLAB solver, which automatically switches between simplex methods (good for small problems, but with a potential exponential complexity) and interior point methods (better for large scale problems, with a guaranteed polynomial worst case complexity) based on the size of the problem. However, other solvers such as CPLEX could easily be used.

5 Applications

As has already been noted, the framework that has been developed thus far is a general one. In what follows, we show results from the application of this framework to the specific problems of image recoloring and shadow removal. In the examples shown, both versions of the color transform give similar results; we tend to favor the second method since it is more mathematically elegant. However, at the end of the section, we provide an example in which the two methods differ.

5.1 Recoloring

For the problem of recoloring, we seek a mapping from target to source which aims to impose the color of the source

Fig. 1 (Color online) Recoloring example, wherein the source and target seeds are taken from the same picture. *Left*: the original. *Right*: the recolored image

upon the target. In order to achieve this goal, we use Lab color, and define the color distance as the regular L_2 distance in Lab space:

$$D^2((L_1,a_1,b_1),(L_2,a_2,b_2)) = (L_1-L_2)^2 + (a_1-a_2)^2 + (b_1-b_2)^2$$

As noted earlier, we use the more straightforward form of user interaction here, in which the user selects two small rectangles, one each from the source and target regions.

Figure 1 shows an example of a recoloring experiment wherein both the source and target seeds are taken from the same picture. In this case our task was to paint the left car that was originally red using the blue color of the right car. Note that while the colors have been changed to match that of the right car, the shading and highlights of the left car have been preserved, as in the original image; see Fig. 2. Figure 3 shows a recoloring example wherein the source and target seeds are taken from two different pictures. In particular, we wanted to "embed" the blue sky from the left picture into the middle picture with gray sky. The result is depicted in the right image. Here again, we preserved the sky texture from the original lightness channel. Notice that in the above examples our method yields a natural look and feel in the reconstructed images.

Fig. 2 Detail from the recoloring example in Fig. 1. Note the way in which the shading and highlights are preserved

5.2 Shadow Removal

For the problem of shadow removal, we seek a mapping from target (shadowed pixels) to source (lit pixels) which aims to retain the chroma characteristics of the target pixels, while at the same time inserting the lightness characteristics of the source pixels. In order to achieve this goal, we use Lab color, and define the color distance as

$$D^2((L_1,a_1,b_1),(L_2,a_2,b_2)) = (a_1-a_2)^2 + (b_1-b_2)^2$$

That is, the color distance depends only on chroma, and not on lightness. Recall that the color distance is used only in computing the flow f or parameters β, i.e. in matching target pixels with source pixels; the transformation rule Φ it-

Fig. 3 (Color online) Recoloring example, wherein the source seed region comes from the *left image*, and the target seed region is taken from the original (in the *middle*). *Right*: the recolored image

Fig. 4 Shadow removal examples. *Top*: the original images. *Bottom*: reconstructed images with shadows removed using the proposed approach

self (see (3) and (9)) operates on the target pixels, and hence inserts the desired lightness.

Figure 4 depicts two examples of shadow removal. The top row shows the original images, while the bottom row shows reconstructed images with shadows removed using the proposed approach. Notice how the 'look and feel' from the lit areas is accurately reproduced in the shadowed areas.

Figure 5 depicts the source and target seed region clouds of points in Lab space, from the grass shadow example (the upper left in Fig. 4). Notice that the location of the point clouds suggests that in order to preserve a natural look in the reconstructed image, the color transform needs to adjust both lightness and chroma values in some non-trivial way. In areas with homogeneous color the linear approach might be sufficient. However, in areas with large color variations, simple approaches fail. To further prove this point, we compare our method with the linear transformation method of [13], and with a more sophisticated pyramid based method of [8]. The results are presented in Fig. 6. The images from left to right are: (1) the original; (2) a linear transformation that scales the mean and the variance of the shadowed area according to the ones from the lit area (as in [13]); (3) a pyramid-based approach from [8]; (4) the proposed approach. Note that, although there are artifacts at the boundary of the shadowed areas as a result of misdetection (these are treated in [8] by a separate algorithm), our method yields the most natural looking image.

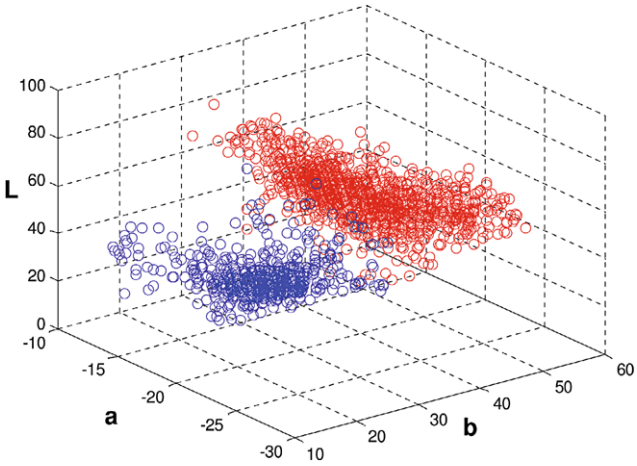


Fig. 5 (Color online) The source and target seed region clouds of points in Lab space, taken from the left picture in Fig. 4. *Blue dots* correspond to the target, *red dots* correspond to the source

Fig. 6 Comparison of the proposed method with competing approaches. From *left* to *right*: (1) the original; (2) the linear transformation from [13]; (3) the pyramid method from [8]; (4) the proposed approach

Fig. 7 Shadow removal examples. *Left*: the optimization in (1). *Right*: the optimization in (12)

Finally, we note that in the shadow image, the two color transforms as given by the optimizations in (1) and (12) differ somewhat, see Fig. 7. In this example, the colors provided by the optimization of (1) may be slightly more accurate; on the other hand, close inspection of the images reveals that the image of the optimization of (1) is distinctly noisier. This is a result of the built-in detection scheme of the

optimization of (12), which leads to smoother results. In this example, both methods suffer from the same edge artifacts described above.

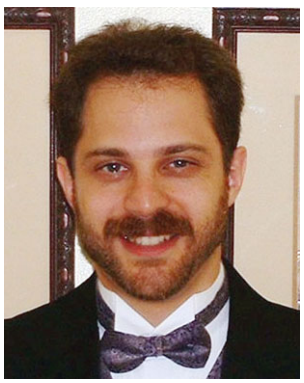
6 Conclusions and Future Directions

We have presented a novel approach for the problem of learning color transforms. Given source and target regions, the proposed approach uses a modified Transportation Problem formulation to learn the color properties of the source region. Then, the color properties of the target region are transformed so as to closely resemble those of the source region. The usefulness of the approach is evidenced by promising results on applications: object recoloring and shadow removal.

Future work will investigate the use of the technique on more complex photometric feature spaces, by complementing color with additional dimensions corresponding to a texture descriptor.

References

1. Weiss, Y.: Deriving intrinsic images from image sequences. In: Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings, vol. 2 (2001)
2. Tappen, M.F., Freeman, W.T., Adelson, E.H.: Recovering intrinsic images from a single image. IEEE Trans. Pattern Anal. Mach. Intell. **27**(9), 1459–1472 (2005)
3. Land, E.H., McCann, J.J.: Lightness and retinex theory. J. Opt. Soc. Am. **61**(1), 1–11 (1971)
4. Oh, B.M., Durand, F., Chen, M.: Image-based modeling and photo editing. In: Proc. SIGGRAPH 2001, pp. 433–442. ACM, New York (2001)
5. Finlayson, G.D., Hordley, S.D., Drew, M.S.: Removing Shadows from Images. Lecture Notes in Computer Science, pp. 823–836 (2002)
6. Finlayson, G.D., Hordley, S.D., Lu, C., Drew, M.S.: On the removal of shadows from images. IEEE Trans. Pattern Anal. Mach. Intell. **28**(1), 59–68 (2006)
7. Chuang, Y.Y., Goldman, D.B., Curless, B., Salesin, D.H., Szeliski, R.: Shadow matting and compositing. ACM Trans. Graph. **22**(3), 494–500 (2003)
8. Shor, Y., Lischinski, D.: The shadow meets the mask: pyramid-based shadow removal. In: Computer Graphics Forum, vol. 27, pp. 577–586. Blackwell, Oxford (2008)
9. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 277–280. ACM, New York (2002)
10. Xiao, X., Ma, L., Kunze, M.: Object-Based Image Recoloring Using Alpha Matte and Color Histogram Specification. Lecture Notes in Computer Science, vol. 4270, p. 194 (2006)
11. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM Trans. Graph. (TOG) **23**(3), 689–694 (2004)
12. Greenfield, G.R., House, D.H.: Image recoloring induced by palette color associations. J. WSCG **11**(1), 189–196 (2003)
13. Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Comput. Graph. Appl., 34–41 (2001)
14. Tai, Y.W., Jia, J., Tang, C.K.: Local color transfer via probabilistic segmentation by expectation-maximization. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 1 (2005)
15. Pitie, F., Kokaram, A.C., Dahyot, R.: N-dimensional probability density function transfer and its application to colour transfer. In: ICCV, pp. 1434–1439 (2005)
16. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Trans. Graph. **22**(3), 313–318 (2003)
17. Kisilev, P., Freedman, D.: Color transforms for creative image editing. In: Color Imaging Conference. CIC 09 (2009)
18. Hitchcock, F.L.: The distribution of a product from several sources to numerous localities. J. Math. Phys. Mass. Inst. Tech. **20**, 224–230 (1941)
19. Rachev, S.T.: The Monge–Kantorovich mass transference problem and its stochastic applications. Theory Probab. Appl. **29**, 647 (1985)
20. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**(2), 99–121 (2000)
21. Haker, S., Zhu, L., Tannenbaum, A., Angenent, S.: Optimal mass transport for registration and warping. Int. J. Comput. Vis. **60**(3), 225–240 (2004)
22. Wu, Y., Huang, T.S.: Color tracking by transductive learning. In: IEEE Conference on Computer Vision and Pattern Recognition, 2000. Proceedings, vol. 1 (2000)
23. Girolami, M., He, C.: Probability density estimation from optimally condensed data samples. IEEE Trans. Pattern Anal. Mach. Intell. 1253–1264 (2003)



Daniel Freedman received the AB degree in physics from Princeton University in 1993 and the PhD degree in engineering sciences from Harvard University in 2000. He was at Rensselaer Polytechnic Institute (RPI) from 2000–2009, first as an Assistant Professor and then as an Associate Professor in the Department of Computer Science. He is currently a Senior Research Scientist at HP Laboratories, Israel. His research interests include computer vision, image processing, and computational geometry and topology. Dr. Freedman is a member of Sigma Xi and the IEEE.

Pavel Kisilev received his BSc in Electrical Engineering and Computer Sciences from Ben Gurion University of the Negev (1994), and his MSc (1998) and PhD (2002) in Electrical Engineering from the Technion. He worked as an algorithm development engineer at the Tensor Systems (1994–1995), and as a research associate at the Vision Laboratory at the Technion (2001–2003). Since 2003 Pavel has been a Senior Research Scientist at HP Laboratories, Israel. Pavel's main interests are Image and Video analysis and processing, and biological signals and systems.