2015-04-01

Color Relationship Transfer for Digital Painting

Gregory Eric Philbrick
Brigham Young University - Provo

Color Relationship Transfer for Digital Painting

Gregory Eric Philbrick

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

April 2015

ABSTRACT

Color Relationship Transfer for Digital Painting

Gregory Eric Philbrick
Department of Computer Science, BYU
Master of Science

A digital painter uses reference photography to add realism to a scene. This involves making n colors in a painting relate to each other more like n corresponding colors in a photograph, in terms of value and temperature. Doing this manually requires either experience or tedious experimentation. Color relationship transfer performs the task automatically, recoloring n regions of a painting so they relate in value and temperature more like n corresponding regions of a photograph. Relationship transfer also has applications in computational photography. In fact, it introduces a new paradigm for image editing in general, based on treating an image's fundamental relationships.

Keywords: image processing, digital painting, color temperature, block-in, illustration, art, graphics, composition, segmentation, underpainting

ACKNOWLEDGMENTS

I first thank Dr. Bryan Morse, without whom I would not have entered graduate work. I also thank the rest of my advising committee: Dr. Bill Barrett, who thought mixing art and science was a good idea; and Dr. Quinn Snell, who assured me from the start that I would find a thesis topic, despite my skepticism.

The idea for relationship transfer comes from concept artist Nathan Fowkes, who came to BYU as a visiting lecturer while I was an undergraduate. He made a bold claim: that good representational painting is not about specific colors, but the value and temperature relationships between them. He was the first person I heard state this idea so succinctly. This project would not have happened without his clarity.

Relationship transfer also owes its existence to watercolorist Bruce MacEvoy, whose writing on artistic color temperature helped me treat that arcane subject with scientific rigor.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Realism in a Painting

All paintings, even stylized or fictional ones, need a degree of realism. "Realism" in this context means correct macroscopic relationships in value and temperature. Value, known also as "tone" [15, p. 81], is the dimension of light versus dark, whereas color temperature is a combined property of hue and saturation.

In physics, "color temperature" refers to the changing light emitted from a black body radiator. Physical color temperature is a mapping from degrees Kelvin to color, defining a curve in hue and saturation space. Artists' color temperature, on the other hand, is a largely intuitive scale from warm to cool. Warm colors appear in the subtractive color wheel around a warm pole, which is typically yellow or orange (Figure 1.1). Cool colors appear on the opposite side of the wheel, around blue. Colors between the poles are not intrinsically warm or cool.

Realism is not about specific values and temperatures, but the relationships between them (Figure 1.2). It is easy to understand this when considering value alone. If a convincing portrait has its lightness or contrast globally altered, it will still look convincing because its original value relationships survive. For instance, the values of the face's shadow side will remain lower than the values of the face's lit side. The same principle applies with temperature; an image will still read correctly after a global, temperature-changing operation, like a saturation change.

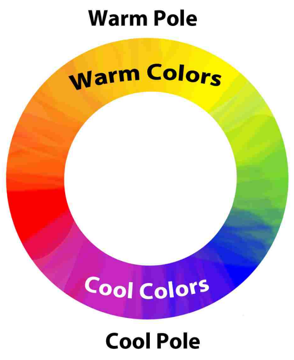


Figure 1.1: Warm and cool colors. Intrinsically warm colors appear around a warm pole in the subtractive color wheel. Cool colors are on the opposite side. In the example painting, the orange and yellow leaves are warmer than the sky and clouds because their colors are closer to the warm pole.

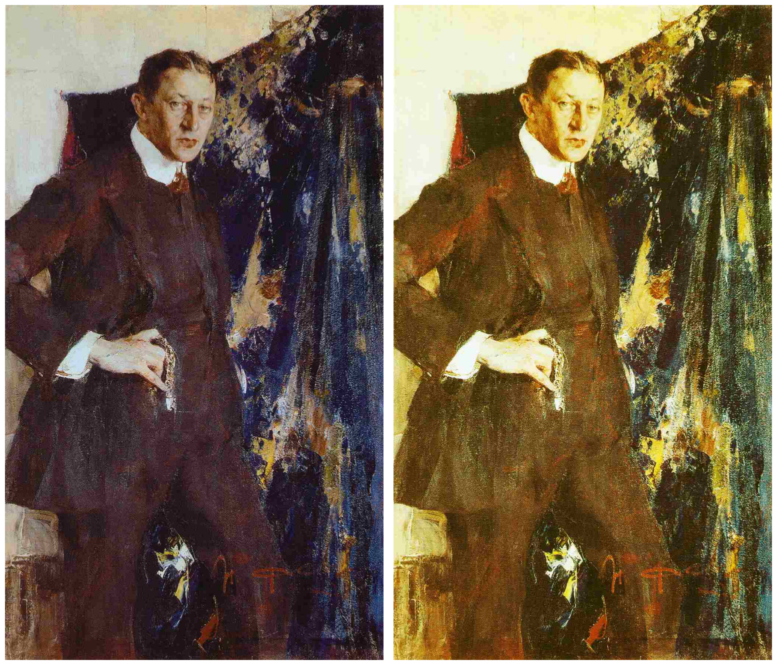
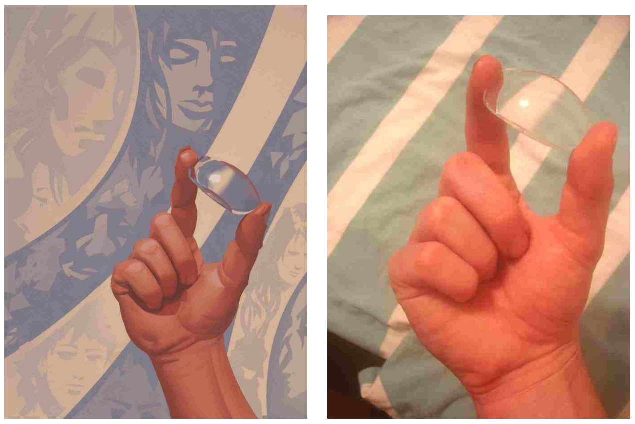


Figure 1.2: Relationships versus specific colors. Realism does not require correct values and temperatures, just correct relationships between them. Nicolai Fechin could not precisely match the colors he saw while painting *Tikhov*, but he imitated their relationships (left). For instance, the shadow side of the subject's face is darker than the lit side. Also, the subject's skin is painted warmer where subcutaneous blood is more visible, such as on the fingers, nose, and cheeks. Correct relationships are more important than the specific colors used, as shown by changing an image's color balance, lightness, or contrast (right). Image courtesy of the State Russian Museum (rusmuseum.ru).



(a) Digital painting (b) Reference photograph

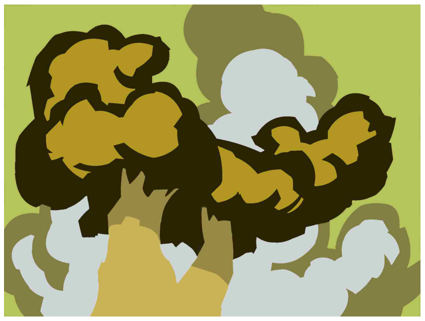
Figure 1.3: Painting from reference. A digital painting (a) made from reference (b) needs to emulate the reference's value and temperature relationships in order to look realistic. The specific colors in the painting need not, and probably will not, be the same as the colors in the reference.

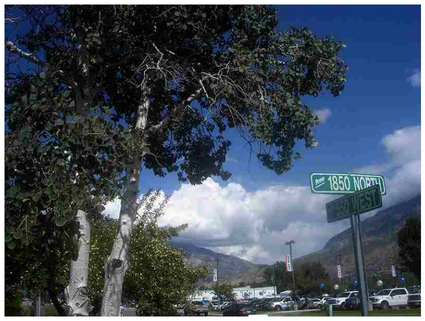
An artist can paint the same scene ad infinitum with different values and temperatures. Every depiction will look realistic as long as crucial relationships hold. Some important relationships are simple, such as "a needs to be lighter than b" or "a needs to be warmer than b". Some could be more complex, like "The value difference between a and b needs to be greater than the value difference between c and d."

1.2 Painting from Reference Photography

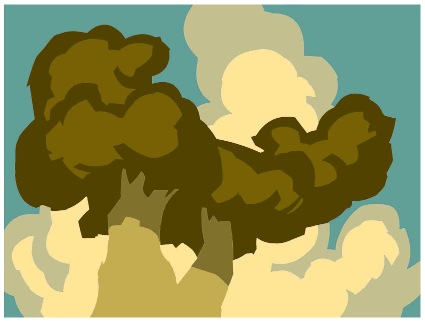
Artists learn many rules of thumb about value and temperature relationships. For example, the darkest value in the light should usually be brighter than the brightest value in shadow. Another rule is that the shadows from a warm light should be cooler than their surroundings, and the shadows from a cool light should be warmer than their surroundings. Even with numerous such rules, it is difficult to paint believable scenes completely from imagination. This is why so many painters use photographic reference (Figure 1.3), even when painting fictional scenes.

In this context, a reference photograph does not provide aesthetic direction; that is the artist's job. In fact, reference photographs are typically of low quality, since artists prefer


(a) Original painting (source)



(b) Reference photograph


(c) Result painting (destination)

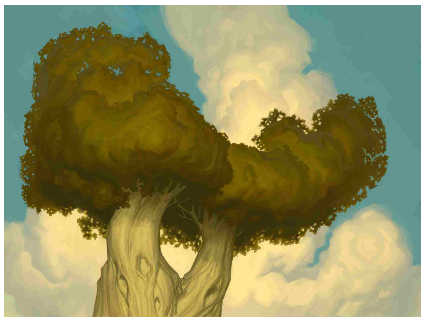

(d) Detail added by painter

Figure 1.4: Relationship transfer. An initial digital painting (a) becomes more realistic (c) by adopting the inter-region value and temperature relationships of a reference photograph (b). The result painting preserves the value/temperature palette of the original painting. Typically, both the initial painting and the result painting are "block-ins", rough and undetailed. The painter adds detail to the result painting to complete it (d).

to take their own pictures rather than rely on stock images. A reference photograph's only job is providing value and temperature relationships for the artist to imitate. (An artist may also use a photograph as a guide for texture or as a drawing reference, but those uses are not relevant here.)

The scene in a reference photograph does not have to be identical to the scene in a painting. There only needs to be enough commonality in content or lighting to provide the relationships the artist needs. For instance, a reference photograph might only provide guidance for one object in a scene.

1.3 Relationship Transfer

Observing the value or temperature relationship between just two regions in a reference photograph takes a veteran eye, and matching that one relationship in a painting can be tricky. Observing and matching the relationships between n of a photograph's regions can be very hard, as the number of crucial relationships scales at least quadratically with n.

Previous work in local color transfer does not solve the problem. Local color transfer moves color distributions from regions of a reference image to corresponding regions of a source image, producing a recolored destination image. It might implicitly make destination regions relate to each other like reference regions, but at the expense of making the destination image take on the aesthetic properties of the reference. This is clearly not desired when painting from reference photography. A reference photograph should not impart its aesthetic properties to a painting, only its relationships.

Relationship transfer as proposed here resembles local color transfer, but it transfers color relationships rather than color distributions. It recolors n regions of a digital painting so that they relate to each other in value and temperature more like n corresponding regions of a reference photograph (Figure 1.4). Where local color transfer makes a_1 equal a_2 and b_1 equal b_2, relationship transfer just makes a_1 and b_1 relate more like a_2 and b_2.

In relationship transfer, the recolored painting regions meet three requirements:

1. Their colors stay within the palette of the original painting as much as possible.
2. They exhibit the same value and temperature relationships as the corresponding regions of the reference photograph.
3. Results are invariant with respect to the reference photograph's global properties: lightness, contrast, and other aesthetic characteristics. The specific values and temperatures in a reference photograph do not matter, only the relationships between them.

Relationship transfer assumes a painting region or photograph region is approximately homogeneous in color, which enables its representation by a single value and a single temperature.

1.4 Block-ins

Digital and traditional painters often begin a painting with a block-in, which consists of rough, quickly-placed blocks of color. A block-in locks important macroscopic properties so that the artist knows the image "works" before continuing. By incrementally adding detail to a block-in without violating its macroscopic properties, the artist brings the painting to completion in an efficient, coarse-to-fine manner (Figure 1.5). A painting's macroscopic properties include object placement, aesthetic choices like color scheme, and, of course, value and temperature relationships.

The painter should lock macroscopic properties at the block-in stage rather than later. This is obvious in traditional painting, since making macroscopic changes to a detailed painting requires painting over previous details and having to redo them. In digital painting, this is not strictly the case. For instance, a digital painter can easily modify global aesthetic properties like contrast, brightness, or color balance without having to redo details. A digital painter could also alter macroscopic value and temperature relationships in a detail-preserving

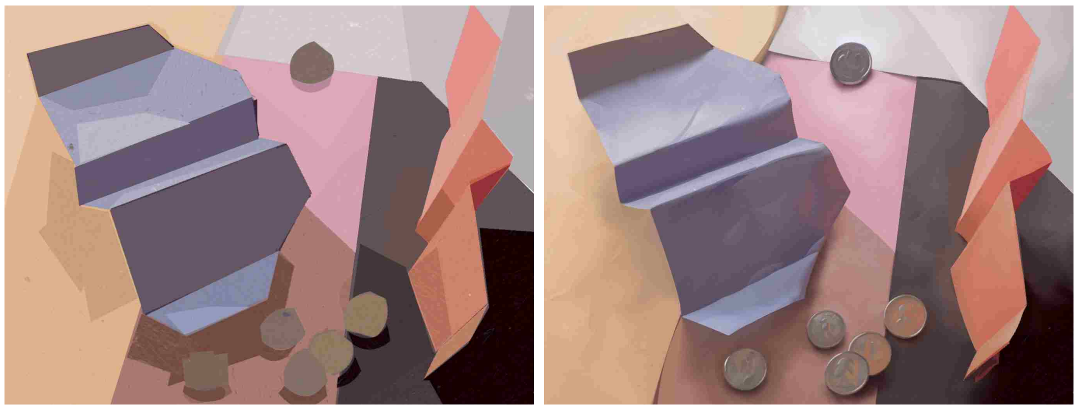


Figure 1.5: Block-in. A block-in (left) becomes a finished painting with the addition of detail (right).

manner using mask selection tools. However, making macroscopic corrections at the end rather than the beginning is amateurish.

Thus, relationship transfer should be applied to block-ins rather than detailed paintings. Relationship transfer can fix detailed paintings as well, but this is needlessly inefficient. Relationship transfer expects the user to break down a painting in a "paint by numbers" fashion, but the semantic regions of a detailed painting are unlikely to be homogeneous in color. There are two choices for handling a detailed painting, both unappealing. First, the user can just violate the assumption that regions are homogeneous in color. This will diminish the power of relationship transfer to produce a realistic looking destination image. The alternative is for the user to keep subdividing regions until every piece is homogeneous in color. This can be very tedious in a detailed image.

An exception is "dots mode", where relationship transfer does not recolor the painting but simply gives the user the new colors required. Here, it does not matter how detailed the painting is. Section 2.7 describes this alternative mode.

To avoid confusion, note that a "block-in" is not strictly the same thing as an "underpainting". An underpainting, like a block-in, is a rough initial pass on a painting. However, unlike a block-in, an underpainting typically lacks the full value or temperature range of the final image. An underpainting serves as a neutral, initial field where the artist can flesh out forms by boosting values or temperatures locally. Admittedly, the difference between an underpainting and a block-in is not always clear, especially in digital painting. For clarity, however, only the term "block-in" will be used from here forward.

1.5 Photograph Editing

Relationship transfer has promising applications in computational photography. Here, the notion of "transfer" recedes, and the focus turns to relationship-preserving edits and "editing in the relationship domain". The goal of relationship transfer reverses in photographic applications. When adding realism to a painting, relationship transfer should change the painting's aesthetic properties as little as possible. In photographic applications, the whole point of relationship transfer is to aid in altering a photograph's aesthetics.

Broadly speaking, relationship transfer either preserves a photograph's original relationships during aesthetic adjustments, or it replaces a photograph's original relationships with new ones specified by the user. In these two roles, relationship transfer reveals a new, relationship-oriented paradigm for image editing in general. This new paradigm, which has implications well beyond digital painting, is the most significant contribution of this research.

The next chapter is the paper presenting relationship transfer. The paper briefly reiterates the introductory material from this chapter for a more technical audience. It then explains relationship transfer's goals and implementation in detail. Finally, it expands on the method's applications in computational photography. The last chapter explores relationship transfer's implications for future research.

Chapter 2

Color Relationship Transfer for Digital Painting

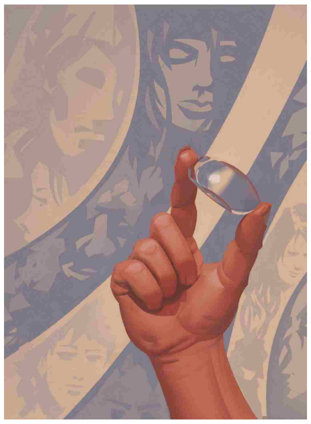
2.1 Introduction

All paintings, even stylized or fictional ones, need a degree of realism. In this context, "realism" concerns colors' values and temperatures. It does not require specific values or temperatures, but proper relationships between values and between temperatures. An artist can repaint the same scene ad infinitum with different colors, and every depiction will look realistic as long as the colors' values and temperatures relate correctly.

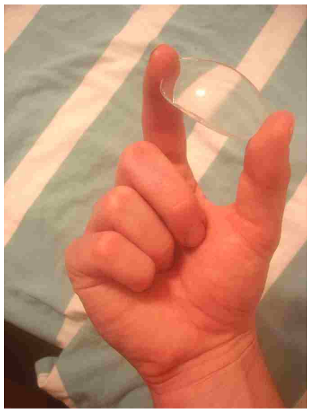
Artists know many rules of thumb about value and temperature relationships. Even so, painting believable scenes from imagination is difficult. This is why painters often use photographic reference (Figure 2.1), even when painting fictional scenes. In this context, a reference photograph does not provide aesthetic direction; that is the artist's job. A reference photograph just provides value and temperature relationships for the artist to imitate. (An artist may also use a photograph as a guide for texture or as a drawing reference, but these uses are not relevant here.)

Observing the value or temperature relationship between just two colors in a reference photograph takes a veteran eye, and matching that one relationship in a painting can be tricky. Observing and matching the relationships between n of a photograph's colors can be very hard, as the number of crucial relationships scales at least quadratically with n.

Previous work in local color transfer does not solve the problem. Local color transfer moves color distributions from regions of a reference image to corresponding regions of a source image, producing a recolored destination image. Local color transfer might implicitly

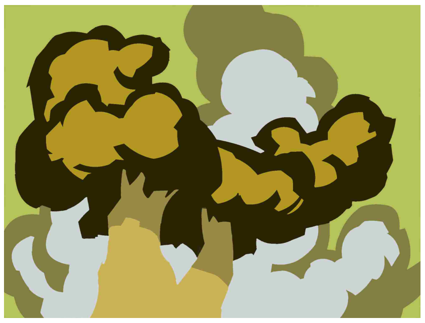


(a) Digital painting

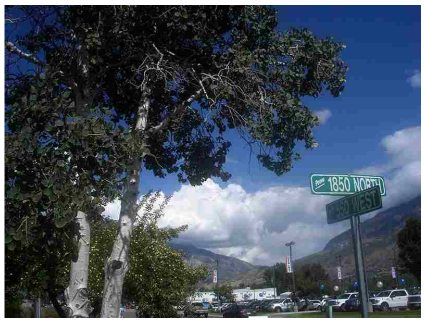


(b) Reference photograph

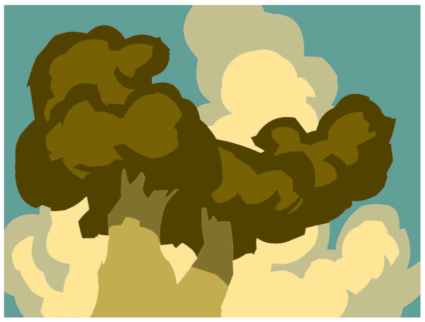
Figure 2.1: Painting from reference. A digital painting (a) made from reference (b) needs to emulate the reference's value and temperature relationships in order to look realistic. The specific colors in the painting need not, and probably will not, be the same as the colors in the reference.



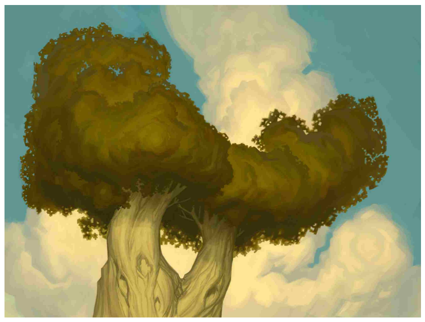
(a) Original painting (source)



(b) Reference photograph



(c) Result painting (destination)



(d) Detail added by painter

Figure 2.2: Relationship transfer. An initial digital painting (a) becomes more realistic (c) by adopting the inter-region value and temperature relationships of a reference photograph (b). The result painting preserves the value/temperature palette of the original painting. Typically, both the initial painting and the result painting are "block-ins", rough and undetailed. The painter adds detail to the result painting to complete it (d).

make a source image's regions relate to each other like a reference image's regions, but only at the expense of making the destination image take on the aesthetic properties of the reference. This is not desirable when painting from reference photography.

Relationship transfer as proposed here resembles local color transfer, but it transfers inter-region color relationships rather than transferring regions' color distributions. It modifies n regions of a digital painting so that they relate to each other in value and temperature more like n corresponding regions in a reference photograph (Figure 2.2). The recolored painting's regions meet three requirements:

1. Their colors stay within the palette of the original painting as much as possible.
2. They exhibit the same value and temperature relationships as the corresponding regions of the reference photograph.
3. Results are invariant with respect to the reference photograph's global properties: lightness, contrast, and so on.

Relationship transfer should usually be applied to "block-in" paintings. A block-in is a rough, initial painting that locks in macroscopic properties like value and temperature relationships before the addition of detail (Figure 2.3). Relationship transfer can correct detailed paintings as well as block-ins, but that is obviously less efficient. "Dots mode" is an exception to this rule (Section 2.7).

Relationship transfer has applications beyond digital painting, in computational photography. Here, the notion of "transfer" recedes, and the focus turns to relationship-preserving edits and "editing in the relationship domain". The goal of relationship transfer reverses in photographic applications. When adding realism to a painting, relationship transfer tries not to change the painting's original aesthetic properties. In photographic applications, the whole point of relationship transfer is to aid in changing a photograph's aesthetic properties. Broadly speaking, relationship transfer either preserves a photograph's original relationships during aesthetic adjustments, or it replaces a photograph's original

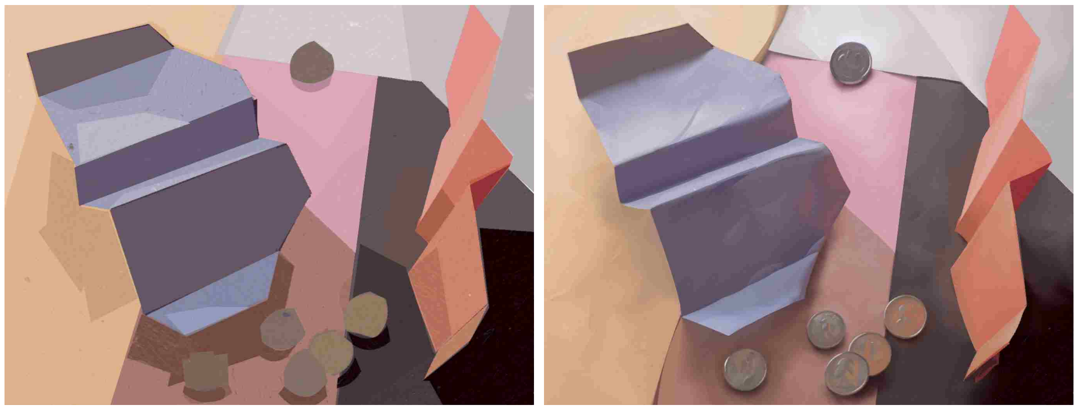


Figure 2.3: Block-in. A block-in (left) becomes a finished painting with the addition of detail (right).

relationships with new ones specified by the user. In these two roles, relationship transfer reveals a new, relationship-oriented paradigm for image editing in general. This new paradigm, which generalizes well beyond digital painting, is the most significant contribution of this research.

2.2 Related Work

Research in digital painting overwhelmingly concerns the simulation of brushes and paints [7, 8, 16, 28, 29]. Little, if any, digital painting research addresses macroscopic value and temperature relationships.

Relationship transfer falls closer to the field of color transfer, which begins with Reinhard et al. [26]. The premise of color transfer is to make a source image take on the color distribution(s) of a reference image, producing a destination image with the reference image's aesthetic properties [4, 5, 12, 14, 17, 19, 21, 22]. Local color transfer pairs regions of a source image with regions of a reference image and performs color transfer within these pairs [1, 3, 9, 20, 27, 31–33]. This resembles the modus operandi of relationship transfer, but does not explicitly address inter-region color relationships. Local color transfer just gives each region of a destination image the color distribution of a corresponding reference region.

Zhang et al. [34] present a local color transfer method that makes a photograph take on the "color contrast" of a reference painting. It finds three dominant hues in the painting, three in the photograph, and a mapping between the two groups. The transfer operation replaces each dominant hue in the photograph with the corresponding dominant hue in the painting. The idea is to transfer the color scheme of the painting to the photograph. Like other local color transfer methods, this essentially just makes pieces of a destination image take on the color distributions of corresponding pieces of a reference image. The phrase "color contrast", as used by the authors, is a misnomer. It actually means "color scheme", and does not refer to the value and temperature relationships treated by relationship transfer.

The methods of Wang et al. [30] and Cohen-Or et al. [6] are not color transfer, but still relate superficially to relationship transfer. Both methods apply a color scheme to an image, where a color scheme is a set of theme colors [30] or a "color harmonic" [6]. The theme colors method actually has little to do with relationship transfer, since it does not deal with the relationships between theme colors. The color harmonic method searches for an image's nearest canonical color scheme, and then makes the image more perfectly fit that color scheme. It shares an interest with relationship transfer in preserving the image's original colors, but there is little commonality beyond that.

Relationship transfer is closest to the work of Zhang et al. [35, 36, 37, 38], who show that the canonical regions of landscape paintings and portrait paintings (foreground, background, and so forth) have consistent relationships in saturation and value [35, 36]. For instance, in a landscape painting the background tends to be less saturated than other regions. Zhang et al. go on to apply these inter-region relationships to amateur landscape photographs or portrait photographs, giving them a painterly aesthetic [37, 38]. Their method semantically segments an amateur photograph and a reference painting into foreground, background, and so on. Each semantic part of the photograph takes on the same average saturation and value as the corresponding semantic region of the painting. It also takes on the same internal value and saturation variation as its partner in the painting, which means the method transfers intra-region relationships as well as inter-region relationships. (Relationship transfer, as proposed here, does not address intra-region relationships, as it assumes all regions to be relatively homogeneous in color.)

Zhang et al. only deal with canonical scenes having a fixed number of semantic regions, while relationship transfer handles any kind of scene with any number of regions. More importantly, their method does not actually deal with inter-region relationships explicitly. It simply makes each photograph region like its corresponding painting region in value and saturation. This does make the photograph regions relate to each other the same way as the painting regions, but is not invariant with respect to the aesthetic properties of the reference image.

Relationship transfer also resembles the application of Huang et al. [10] in its use of a mathematical metric for quantifying artistic color temperature. Relationship transfer's user interface for editing region masks borrows from Lischinski et al. [13]. The smooth destination image generation described in Section 2.4.4 also borrows from Lischinski et al.

2.3 Objective

Relationship transfer recolors an initial painting or "source image" so that n user-marked regions relate to each other in value and temperature more like n corresponding user-marked regions in a reference image. Relationship transfer deals with regions in terms of their mean values and temperatures. Given n mean values and n mean temperatures from both the source and reference images, relationship transfer first finds n new mean values and n new mean temperatures for the source image. Then it recolors the source image's pixels relative to the new mean values and temperatures.

The notation s_i represents the mean value or temperature of one of the source image's regions. The index i is in $[0, n-1]$. Similarly, r_i denotes the mean value or temperature of a reference image region. Finally, d_i refers to the mean value or temperature of a region in the recolored destination image. The vectors $\boldsymbol{s}$, $\boldsymbol{r}$, and $\boldsymbol{d}$ represent the collective values or temperatures for the source, reference, and destination images respectively:

$$\begin{aligned}
\boldsymbol{s} &= [s_0, s_1 ... s_{n-1}] \\
\boldsymbol{r} &= [r_0, r_1 ... r_{n-1}] \\
\boldsymbol{d} &= [d_0, d_1 ... d_{n-1}]
\end{aligned} \tag{2.1}$$

Relationship transfer deals with values and temperatures independently; there are individual $\boldsymbol{s}$, $\boldsymbol{r}$, and $\boldsymbol{d}$ vectors for values and for temperatures. Determining $\boldsymbol{d}$ works the same way in both cases. The following discussion is independent of whether $\boldsymbol{s}$, $\boldsymbol{r}$, and $\boldsymbol{d}$ represent values or temperatures.

The computation of the destination vector $\boldsymbol{d}$ attempts to reconcile two conflicting goals. First, $\boldsymbol{d}$ should have the same internal relationships as the reference vector $\boldsymbol{r}$. Second, $\boldsymbol{d}$ should preserve the source image's palette, as defined by $\boldsymbol{s}$. The first goal adds realism to the destination image, thus fulfilling the primary purpose of relationship transfer. The second goal makes the destination image preserve the aesthetic properties of the source image.

Three properties dictate the choice of $\boldsymbol{d}$. The first two properties pertain to honoring the relationships of $\boldsymbol{r}$ and the third pertains to preserving the palette of $\boldsymbol{s}$.

1. **Ordering Property**: If one color is brighter, darker, warmer, or cooler than another in the reference image, the same relationship should hold in the destination image:

$$\forall i, j \in [0, n-1] \mid i \neq j : r_i > r_j \implies d_i > d_j \tag{2.2}$$

2. **Offset Ratios Property**: Every ratio of value or temperature offsets should be the same in the destination image as in the reference image:

$$\forall i, j, k, l \in [0, n-1] \mid i \neq j, k \neq l : \frac{d_i - d_j}{d_k - d_l} = \frac{r_i - r_j}{r_k - r_l} \tag{2.3}$$

This property treats ratios of *differences* between values or temperatures rather than ratios of the quantities themselves. Using ratios of values or temperatures would be simpler, but this would make Property 2 variant with respect to the global properties of $\boldsymbol{r}$. The relationships in $\boldsymbol{r}$ would be quantified differently depending on the range and placement of $\boldsymbol{r}$ in value/temperature space, whereas the destination vector $\boldsymbol{d}$ should respect only the internal relationships of $\boldsymbol{r}$. Equation 2.3 quantifies an image's internal relationships consistently, invariant to linear transformations of the image's gray levels. The $i \neq j$ and $k \neq l$ rules in Equation 2.3 remove trivial relationships.

3. **Palette Preservation Property**: Each new value or temperature should be the same as one of the original values or temperatures:

$$\forall i \in [0, n-1] : \exists j \in [0, n-1] \mid d_i = s_j \tag{2.4}$$

The indices i and j are not necessarily equal and that there is not necessarily a bijection between $\boldsymbol{d}$ and $\boldsymbol{s}$.

Properties 1 and 2 work closely together. In fact, Property 2 almost subsumes Property 1. It can be shown that if Property 2 holds for vector $\boldsymbol{d}$, then $\boldsymbol{d}$ has either the same ordering as $\boldsymbol{r}$ or the exact opposite ordering of $\boldsymbol{r}$ (Appendix A). Thus, Property 2 comes close to covering Property 1, but not close enough to make Property 1 redundant.

Still, Properties 1 and 2 act in concert. Together, they push $\boldsymbol{d}$ to honor the relationships of the reference vector $\boldsymbol{r}$. On the other side, Property 3 encourages $\boldsymbol{d}$ to preserve the palette of the source vector $\boldsymbol{s}$. This dichotomy in interests suggests a way to find $\boldsymbol{d}$: minimizing a weighted two-term function $Cost(\boldsymbol{d})$:

$$Cost(\boldsymbol{d}) = RelateCost(\boldsymbol{d}, \boldsymbol{r}) + \lambda_{palette} PaletteCost(\boldsymbol{d}, \boldsymbol{s}) \tag{2.5}$$

$RelateCost(\boldsymbol{d}, \boldsymbol{r})$ encourages $\boldsymbol{d}$ to honor Properties 1 and 2 while $PaletteCost(\boldsymbol{d}, \boldsymbol{s})$ encourages $\boldsymbol{d}$ to honor Property 3. $\lambda_{palette}$ weighs palette preservation against honoring relationships, and is not necessarily the same for values as for temperatures (a typical weight is 1).

Property 2 suggests an definition for $RelateCost(\boldsymbol{d}, \boldsymbol{r})$:

$$\sum_{i,j,k,l} \left(\frac{d_i - d_j}{d_k - d_l} - \frac{r_i - r_j}{r_k - r_l} \right)^2 \Bigg| \, i \neq j, k \neq l \tag{2.6}$$

However, the quotients in this summation are potentially unbounded. This weakens the power of $\lambda_{palette}$ to weigh $PaletteCost(\boldsymbol{d}, \boldsymbol{s})$ against $RelateCost(\boldsymbol{d}, \boldsymbol{r})$. Using symmetries, the summation could exclude terms where the $\boldsymbol{r}$ divisor was smaller than the $\boldsymbol{r}$ numerator. Yet this would not necessarily avoid terms where the $\boldsymbol{d}$ divisor was smaller than the $\boldsymbol{d}$ numerator.

Applying cross-multiplication to Equation 2.3 produces an alternate expression of Property 2:

$$(d_i - d_j)(r_k - r_l) = (d_k - d_l)(r_i - r_j) \tag{2.7}$$

This rearrangement suggests a better definition for $RelateCost(\boldsymbol{d}, \boldsymbol{r})$, one that contains no quotients and remains bounded:

$$RelateCost(\boldsymbol{d}, \boldsymbol{r}) = \sum_{i,j,k,l} \left((d_i - d_j)(r_k - r_l) - (d_k - d_l)(r_i - r_j)\right)^2 \mid i < j, k < l \tag{2.8}$$

This form of $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ also uses symmetries to reduce redundant computation. It includes only terms where $i < j$ and $k < l$, thus quartering the number of terms without changing the meaning of minimizing $RelateCost(\boldsymbol{d}, \boldsymbol{r})$.

An intuitive definition of $PaletteCost(\boldsymbol{d}, \boldsymbol{s})$ directly implements Equation 2.4 from Property 3:

$$PaletteCost(\boldsymbol{d}, \boldsymbol{s}) = \sum_{i} \min_{j} (d_i - s_j)^2 \tag{2.9}$$

$Cost(\boldsymbol{d})$ is insufficient as defined by Equations 2.8 and 2.9. It ought to enforce Properties 1–3, but only reliably covers Properties 2 and 3. Because Property 2 partially guarantees Property 1, $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ partially guarantees that $\boldsymbol{d}$ will have the same ordering as $\boldsymbol{r}$. Yet the guarantee is only partial: $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ will also allow $\boldsymbol{d}$ to have the exact opposite ordering of $\boldsymbol{r}$. $\mathit{Cost}(\boldsymbol{d})$ should always reject an out-of-order $\boldsymbol{d}$ in favor of an in-order $\boldsymbol{d}$.

The chosen solution is to redefine $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ so that it imposes a bijection between $\boldsymbol{d}$ and $\boldsymbol{s}$. This bijection pushes $\boldsymbol{d}$ to have the same ordering as $\boldsymbol{s'}$, where $\boldsymbol{s'}$ is $\boldsymbol{s}$ reordered to have the same ordering as $\boldsymbol{r}$. For example, if $\boldsymbol{r} = [r_0, r_1, r_2] = [0, 0.1, 0.2]$ and $\boldsymbol{s} = [s_0, s_1, s_2] = [0.5, 0.4, 0.6]$, then $\boldsymbol{s'} = [s'_0, s'_1, s'_2] = [0.4, 0.5, 0.6]$. The revised $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ is

$$\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s}) = \sum_i (d_i - s'_i)^2 \tag{2.10}$$

This $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ is less semantically correct than that of Equation 2.9. However, the revised $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ helps ensure that the destination vector $\boldsymbol{d}$ that minimizes $\mathit{Cost}(\boldsymbol{d})$ will have the same ordering as $\boldsymbol{r}$. Under the original definition of $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$, a highly incorrect $\boldsymbol{d}$ could optimize $\mathit{Cost}(\boldsymbol{d})$ in some scenarios. For instance, the source $\boldsymbol{s}$ might have the opposite ordering of $\boldsymbol{r}$ (a reasonable allowance, since the initial painting is expected to have poor relationships). In such a case, a $\boldsymbol{d}$ with the opposite ordering of $\boldsymbol{r}$ could optimize $\mathit{Cost}(\boldsymbol{d})$, even though it grossly violated Property 1.

2.4 Procedure

This section explains the steps of relationship transfer.

1. Define n pairings between source regions and reference regions (Section 2.4.1).
2. Use the source and reference regions to determine vectors $\boldsymbol{s}$ and $\boldsymbol{r}$ for values and separate vectors $\boldsymbol{s}$ and $\boldsymbol{r}$ for temperatures (Section 2.4.2).
3. Find the destination vector $\boldsymbol{d}$ for values and the destination vector $\boldsymbol{d}$ for temperatures such that each minimizes $\mathit{Cost}(\boldsymbol{d})$ from Equation 2.5 (Section 2.4.3).

4. Recolor each pixel in the source image based on the transition from $\boldsymbol{s}$ to $\boldsymbol{d}$ for values and temperatures (Section 2.4.4).

Figures 2.4 and 2.5 illustrate the process. Figure 2.4 shows relationship transfer applied to a block-in painting of a tree in front of some clouds. It shows the source, reference, and destination images along with their associated values and temperatures. The destination image comes from replacing $\boldsymbol{s}$ with $\boldsymbol{d}$ for values and for temperatures. The difference between the destination image and the source image mirrors the difference between the red and green graphs.

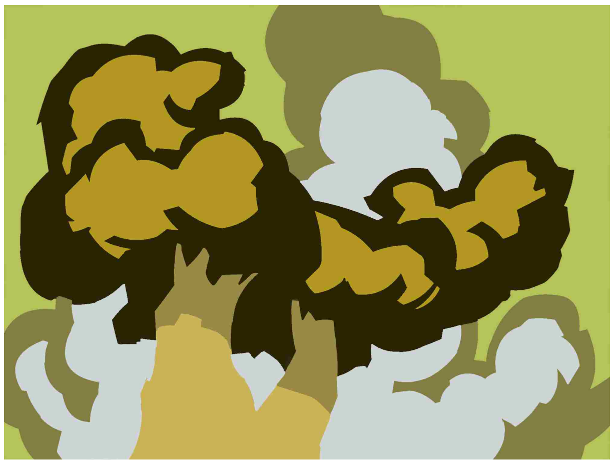
Figure 2.5 shows the regions of the source and reference images. There are seven region pairs numbered from 0 to 6. The region pairs have the following names, corresponding to an intuitive decomposition of the common scene:

0. *trunkLight*: The parts of the tree trunk not in shadow.
1. *trunkDark*: The parts of the tree trunk in shadow.
2. *leafDark*: The dark parts of the foliage.
3. *leafLight*: The light parts of the foliage.
4. *cloudLight*: The light parts of the clouds.
5. *cloudDark*: The dark parts of the clouds.
6. *sky*: The sky.

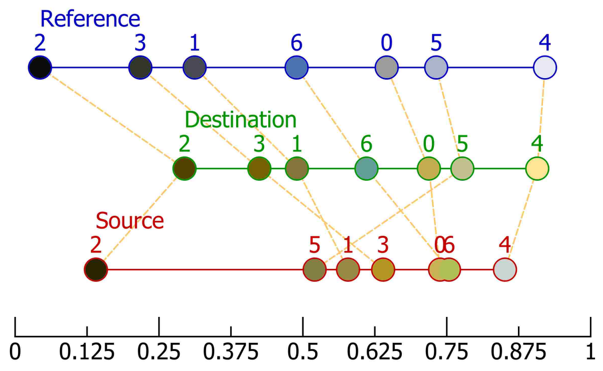
2.4.1 Region Pairs

A region is a fuzzy, discontiguous mask over its image: $\mathbb{Z}^2 \Rightarrow [0, 1]$. A region pair has a source region in the source image and a reference region in the reference image. The ith source region mask is effectively the ith destination region mask.

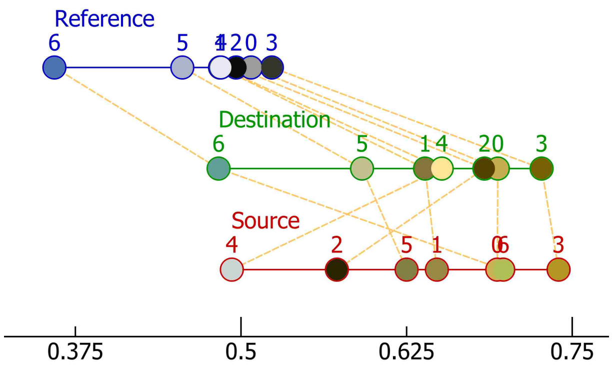
A region pair is part of a scene's semantic decomposition. A region pair might represent a grassy field, a cloud, a hubcap, the highlights on a hubcap, or the shadow sides


(a) Source

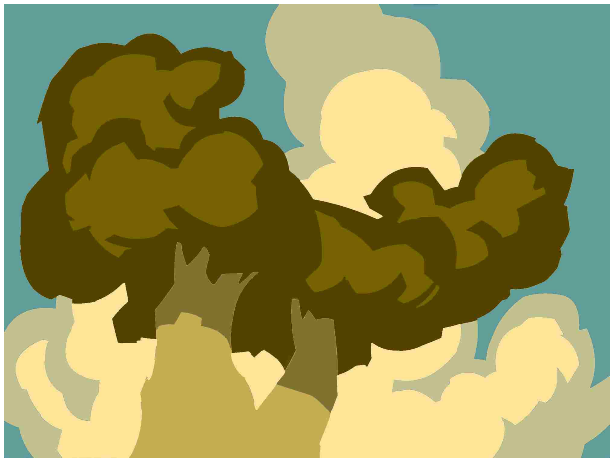
(b) Reference



(c) Values



(d) Temperatures


(e) Destination

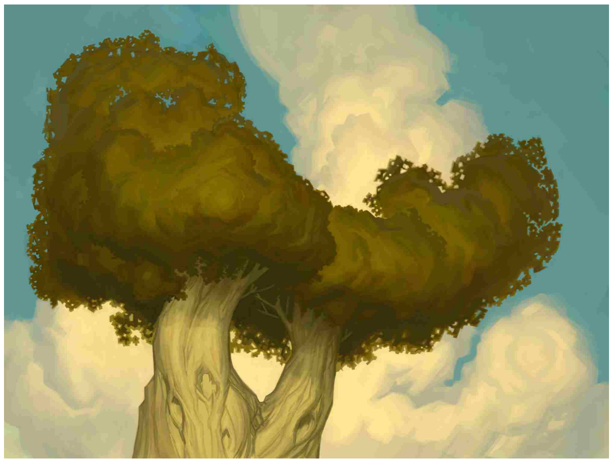
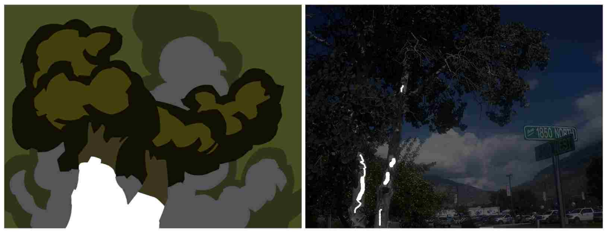
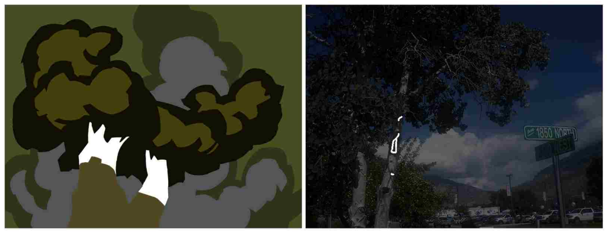
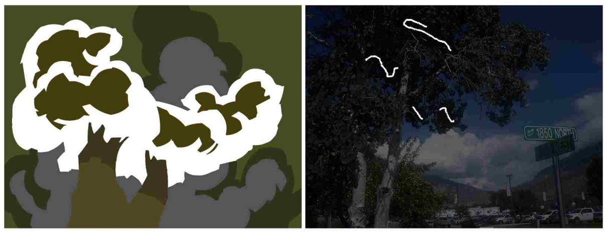
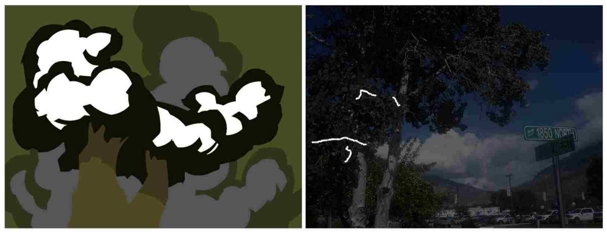

(f) Finished painting

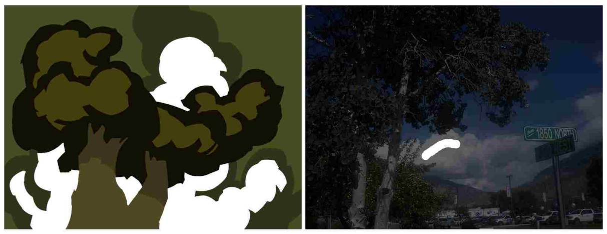
Figure 2.4: Relationship transfer example. Relationship transfer changes the values and temperatures of the block-in painting (a) so they relate more like the reference photograph's values and temperatures. The result is a more realistic-looking block-in (e). The improved block-in becomes a finished painting (f) with the addition of detail by the painter. Note that in the graphs (c,d), each dot has the color of the corresponding region in the source, reference, or destination image. Also note that indices 1 and 4 are inadvertently superimposed in the reference temperatures graph (d).

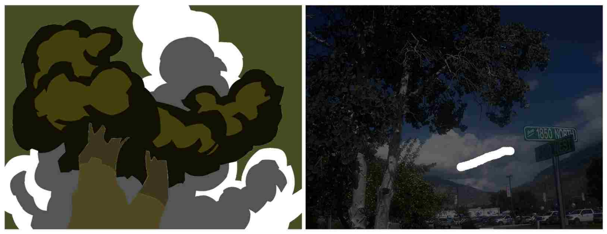

(a) Pair 0: *trunkLight*


(b) Pair 1: *trunkDark*


(c) Pair 2: *leafDark*


(d) Pair 3: *leafLight*


(e) Pair 4: *cloudLight*


(f) Pair 5: *cloudDark*

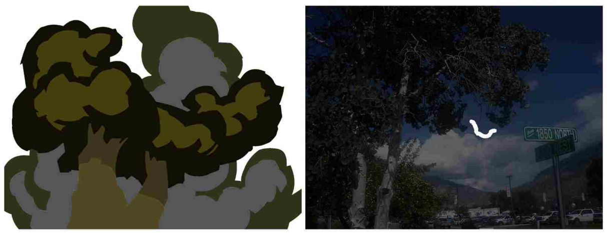
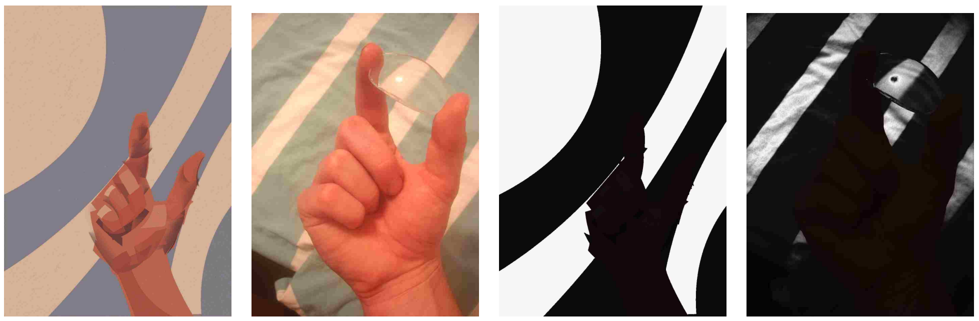

(g) Pair 6: *sky*

Figure 2.5: Region pairs of transfer example. Each region pair consists of a region in the source image and a corresponding region in the reference image. Observe that regions can be discontiguous.

of every object in a scene. The pair covers its semantic content in both the source image and the reference image. For instance, the region pair in Figure 2.6 represents a scene's white background stripes. The source region's shape is more crucial than the reference region's shape. The reference region only needs to provide a representative value and temperature for white background stripes in the reference image. The source region must similarly provide the current value and temperature of white stripes in the source image. However, it must also define the area of effect for the transition from an initial white stripes value and temperature to a new white stripes value and temperature. The lesser importance of reference regions' shapes is especially apparent in Figure 2.5.

A scene decomposition is not necessarily comprehensive; source and reference regions need not completely partition their images. Also, regions may overlap.

Because relationship transfer represents every region as a mean value and temperature, all regions should be roughly homogeneous in color. Color variation within regions is permissible, but too much variation will hurt the realism of the destination image.



(a) Source (b) Reference (c) Source region (d) Reference region

Figure 2.6: Region pair. If relationship transfer were applied to the block-in painting (a) of a hand against a blue and white background, with the photograph (b) as reference, one of the region pairs might represent the white background stripes. The source region (c) and reference region (d) would comprise the "white background" region pair. Other region pairs might be "blue background", "hand highlight", "hand shadow side", "fingernails", and so on according to the user's decomposition of the scene.

2.4.2 Determining Representative Values and Temperatures

In order to transfer relationships from the n reference regions to the n source regions, the method needs to associate each region with a representative value or temperature. The ith source region has value or temperature s_i, while the ith reference region has value or temperature r_i. A region's representative value or temperature is the weighted average of its pixels' values or temperatures. The value and temperature of a pixel c come from the L* and b* channels of c in L*a*b*. For readability, the following equations refer to L* and b* as L and b.

$$\mathit{value}(c) = c_L/100 \tag{2.11}$$

$$\mathit{temperature}(c) = (c_b + 128)/256 \tag{2.12}$$

Values and temperatures fall in $[0, 1]$.

2.4.3 Finding the New Values and Temperatures

Relationship transfer computes a set of destination quantities $\boldsymbol{d}$ for values, and another for temperatures. Each $\boldsymbol{d}$ is chosen to minimize $\mathit{Cost}(\boldsymbol{d})$ from Equation 2.5. This equation is nonlinear, non-convex, and difficult to optimize directly. However, it lends itself to approximation via an overdetermined linear system of equations. This approximation becomes the starting point for local nonlinear optimization.

The overdetermined linear system has two parts. One part solves for the optimal $\boldsymbol{d}$ with respect to $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ from Equation 2.8. The other part solves for the optimal $\boldsymbol{d}$ with respect to $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ from Equation 2.10. The two parts combine to form an overdetermined linear system that approximates the optimal $\boldsymbol{d}$ with respect to $\mathit{Cost}(\boldsymbol{d})$.

Optimizing $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ alone is easy: any linear function of $\boldsymbol{r}$ is a perfectly optimal $\boldsymbol{d}$. For any $\boldsymbol{d} = \alpha\boldsymbol{r} + \beta$, where α and β are constants, $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r}) = 0$. This is intuitive: scaling and translating a set of numbers will perfectly preserve their internal offset ratios. If $\boldsymbol{d}$ is the result of scaling and translating $\boldsymbol{r}$, then $(d_i - d_j)/(d_k - d_l)$ will equal $(r_i - r_j)/(r_k - r_l)$ for all selections of i, j, k, and l, which means that $\boldsymbol{d}$ will perfectly satisfy Property 2 and every term in $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ will be zero. Equation 2.7, the cross-multiplied form of Property 2, will hold generally:

$$(d_i - d_j)(r_k - r_l) = (d_k - d_l)(r_i - r_j) \tag{2.13}$$

Rearranging the terms produces a linear equation in terms of the unknown $\boldsymbol{d}$:

$$d_i(r_k - r_l) + d_j(r_l - r_k) + d_k(r_j - r_i) + d_l(r_i - r_j) = 0 \tag{2.14}$$

The resulting linear system $\boldsymbol{Rd} = \mathbf{0}$ is underconstrained on its own. $\boldsymbol{R}$ can be constructed a row at a time for each selection of i, j, k, and l, where $i < j$ and $k < l$ as in Equation 2.8. For each selection of indices, let $\boldsymbol{x}$ be a new row of $\boldsymbol{R}$, initially all zeros. Add $r_k - r_l$ to x_i, add $r_l - r_k$ to x_j, add $r_j - r_i$ to x_k, and add $r_i - r_j$ to x_l.

A linear system that solves for $\boldsymbol{d}$ with respect to $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ can be constructed by similarly assuming perfect palette preservation, which means setting each term in Equation 2.10 to zero. The system represents $d_i = s'_i$ for every i in $[0, n-1]$, where $\boldsymbol{s'}$ is $\boldsymbol{s}$ reordered to have the same ordering as $\boldsymbol{r}$. The resulting linear system is $\boldsymbol{Id} = \boldsymbol{s'}$. $\lambda_{\mathit{palette}}$ modulates the importance of palette preservation over honoring relationships, yielding $\lambda_{\mathit{palette}}\boldsymbol{Id} = \lambda_{\mathit{palette}}\boldsymbol{s'}$.

Combining $\boldsymbol{Rd} = \mathbf{0}$ and $\lambda_{\mathit{palette}}\boldsymbol{Id} = \lambda_{\mathit{palette}}\boldsymbol{s'}$ produces an overdetermined linear system whose least squares approximation estimates the optimal $\boldsymbol{d}$ with respect to $\mathit{Cost}(\boldsymbol{d})$:

$$\begin{bmatrix} \boldsymbol{R} \\ \lambda_{\mathit{palette}}\boldsymbol{I} \end{bmatrix} \boldsymbol{d} = \begin{bmatrix} \mathbf{0} \\ \lambda_{\mathit{palette}}\boldsymbol{s'} \end{bmatrix} \tag{2.15}$$

The left side matrix $\boldsymbol{R}$ has width n and height equal to the number of terms in $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ plus the number of terms in $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$. $\boldsymbol{d}$ is a column of height n.

For example, if n is 4, the system looks like this:

$$\begin{bmatrix} r_1 - r_2 & r_2 - r_1 + r_1 - r_0 & r_0 - r_1 & 0 \\ r_1 - r_3 & r_3 - r_1 + r_1 - r_0 & 0 & r_0 - r_1 \\ r_2 - r_3 & r_3 - r_2 & r_1 - r_0 & r_0 - r_1 \\ 0 & r_2 - r_3 & r_3 - r_2 + r_2 - r_1 & r_1 - r_2 \\ \lambda_{palette} & 0 & 0 & 0 \\ 0 & \lambda_{palette} & 0 & 0 \\ 0 & 0 & \lambda_{palette} & 0 \\ 0 & 0 & 0 & \lambda_{palette} \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \lambda_{palette}\ s'_0 \\ \lambda_{palette}\ s'_1 \\ \lambda_{palette}\ s'_2 \\ \lambda_{palette}\ s'_3 \end{bmatrix} \tag{2.16}$$

The top four rows of the system are relationship rows built from Equation 2.14. The first is for $i = 0, j = 1, k = 1, l = 2$. The second is for $i = 0, j = 1, k = 1, l = 3$. The third is for $i = 0, j = 1, k = 2, l = 3$. The fourth is for $i = 1, j = 2, k = 2, l = 3$. The bottom four rows of the system enforce palette preservation.

The top portion of the system grows at rate $\mathcal{O}(n^5)$, and the bottom portion grows at rate $\mathcal{O}(n^2)$. The growth rate $\mathcal{O}(n^5)$ is formidable. However, when the user must manually define each region pair, n will remain manageably small. Also, the matrix $\boldsymbol{R}$ is sparse, which reduces both storage and computation requirements.

After this overdetermined linear system produces an approximate optimal $\boldsymbol{d}$, that approximation becomes the starting point for direct nonlinear local minimization of Equation 2.5.[1] Experimentation with randomly generated $\boldsymbol{s}$ and $\boldsymbol{r}$ vectors shows that $Cost(\boldsymbol{d})$ usually improves by no more than eight percent, with improvement decreasing as n increases. Although individual values and temperatures can change by almost 0.1, they usually change by much less, not enough to significantly alter the appearance of the destination image. This suggests that nonlinear optimization can usually be omitted.

[1]We use the BOBYQA algorithm presented by Powell [23] and implemented in Steven Johnson's nonlinear optimization library *NLopt* [11]. The maximum initial step size is 10^{-4}. Each d_i has lower bound $0 - \epsilon$ or $\min d_j - \epsilon$, if $\min d_j$ is less than 0. Each d_i has upper bound $1 + \epsilon$, or $\max d_j + \epsilon$ if $\max d_j$ is greater than 1. The constant ϵ is 10^{-6}.

Relationship transfer makes each destination vector $\boldsymbol{d}$ relate internally more like the corresponding reference vector $\boldsymbol{r}$. This is why the green graphs in Figure 2.4 have the same ordering and roughly the same shape as the blue reference graphs rather than the red source graphs. For instance, the source values in Figure 2.4c are ordered $2, 5, 1, 3, 0, 6, 4$, whereas the destination values have the same ordering as the reference values: $2, 3, 1, 6, 0, 5, 4$.

While making these corrections, relationship transfer also keeps destination values and temperatures close to the palette of original values and temperatures. The destination temperatures in Figure 2.4d clearly demonstrate palette preservation. In Figure 2.4c, the destination values shift right relative to the source values. The shift strikes a balance between making $\boldsymbol{d}$ relate like $\boldsymbol{r}$ and keeping the values of $\boldsymbol{d}$ close to the palette of $\boldsymbol{s}$, which has a lot of its values bunched in the 0.5 to 0.875 range. The effect is a lightening of the destination image relative to the source image, most conspicuous in *leafDark* (number 2).

2.4.4 Recoloring the Source Image

Given a computed set of destination quantities $\boldsymbol{d}$ for values and another set for temperatures, the next step is to find a new value and temperature for every pixel in the source image. Value and temperature adjustments occur independently.

Given $f : \mathbb{Z}^2 \Rightarrow [0, 1]$, the source image's original values or temperatures, the goal is to determine $g : \mathbb{Z}^2 \Rightarrow [0, 1]$, the destination image's values or temperatures. In this section, m_a^b is a value in $[0, 1]$ representing the coverage of the ath source region at position b.

There are two methods for determining g:

1. *Simple*: For each source image position x, the new value or temperature is a weighted sum of the recommendations from each covering source region, or simply the original value or temperature if there are no covering regions. m_i^x is the coverage value in $[0, 1]$ of the ith source region at position x. A region covering x recommends a value or temperature that preserves the original offset from the region's original mean value or

temperature.

$$g(x) = \begin{cases} \sum_i m_i^x (f(x) - s_i + d_i) / \sum_i m_i^x & \text{or} \\ f(x) & \text{if} \sum_i m_i^x = 0 \end{cases} \tag{2.17}$$

The simple recoloring method is responsible for the transition from source image to destination image in Figure 2.4.

2. *Smooth*: This achieves the same effect as the simple method, except that value and temperature per-pixel modifications are smooth with respect to ∇c, where c is the original source image in L*a*b*. The approach comes from the smooth "influence function" generation of Lischinski et al. [13]. It solves an image-wide linear system for the additive function h such that $g(x) = f(x) + h(x)$. The system keeps h smooth by requiring $\Delta h = 0$ with diminishing importance as $||\nabla c||$ increases. h is the solution to $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ and $\boldsymbol{b}$ take the following form:

$$A_{ij} = \begin{cases} -\lambda / (||c(i) - c(j)|| + \epsilon) & j \in N_4(i) \\ \max_k m_k^i - \sum_{l \in N_4(i)} A_{il} & i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

$$b_i = (\max_k m_k^i)(\sum_k m_k^i (d_k - s_k) / \sum_k m_k^i) \tag{2.19}$$

λ weighs the importance of keeping h smooth versus honoring the recommended values or temperatures of the source regions. ϵ is 10^{-4}, a small constant that prevents division by zero.

For a pixel location i, the term $\max_k m_k^i$ is the largest coverage value in $[0, 1]$ of any k-indexed source region covering position i. For b_i, the expression $\sum_k m_k^i (d_k - s_k) / \sum_k m_k^i$ is the weighted average of the differences between source regions' original values or temperatures and their new values or temperatures. Recall that $\boldsymbol{Ax} = \boldsymbol{b}$ solves for the additive function h, not g itself.

The smooth method is much more time-consuming than the simple method. However, it is valuable when the destination image needs to be free of artifacts and the region masks are not accurate enough to accomplish this.

Once new values and temperatures are available for every pixel in the source image, producing the destination image means changing each source pixel's L* and b* channels using the inverses of Equations 2.11 and 2.12.

2.4.5 Anchoring

An anchor fixes d_i to some value or temperature. This might be the original source value or temperature s_i, some previous d_i, or an arbitrary number specified by the user. Relationship transfer makes unanchored values or temperatures relate correctly with anchored values or temperatures, as well as each other. This requires modifying the linear system of Equation 2.15:

$$\begin{bmatrix} \boldsymbol{R} \\ \lambda_{palette}\boldsymbol{I} \end{bmatrix} \boldsymbol{d} = \begin{bmatrix} \boldsymbol{0} \\ \lambda_{palette}\boldsymbol{s'} \end{bmatrix} \tag{2.20}$$

Anchoring alters both parts of the linear system. The $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ part changes from $\boldsymbol{Rd} = \boldsymbol{0}$ to $\boldsymbol{Ry} = \boldsymbol{b}$. $\boldsymbol{y}$ is a column of height m rather than n, containing only the unanchored parts of $\boldsymbol{d}$. For instance, if $n = 4$ and d_2 is anchored, then $m = 3$ and

$$\boldsymbol{y} = \begin{bmatrix} d_0 \\ d_1 \\ d_3 \end{bmatrix} \tag{2.21}$$

The right side is now $\boldsymbol{b}$ rather than a column of zeros. The height of the column and the height of $\boldsymbol{R}$ remain unchanged, since the system still handles the same number of relationships. Building $\boldsymbol{R}$ proceeds almost as before, adding a row $\boldsymbol{x}$ to $\boldsymbol{R}$ and now also an entry to $\boldsymbol{b}$

for each selection of i, j, k, and l. As before, the new row $\boldsymbol{x}$ and corresponding entry in the column $\boldsymbol{b}$ come from Equation 2.14:

$$d_i(r_k - r_l) + d_j(r_l - r_k) + d_k(r_j - r_i) + d_l(r_i - r_j) = 0 \tag{2.22}$$

Suppose that d_i is anchored but d_j, d_k, and d_l are not. The constant term $d_i(r_k - r_l)$ moves to the right hand side, yielding

$$d_j(r_l - r_k) + d_k(r_j - r_i) + d_l(r_i - r_j) = -d_i(r_k - r_l) \tag{2.23}$$

In this particular instance, building $\boldsymbol{x}$ and the newest entry in $\boldsymbol{b}$ proceeds thus: Set $\boldsymbol{x}$ to all zeros and the newest entry in $\boldsymbol{b}$ (call it b) to 0. Add $r_l - r_k$ to $x_{j'}$, add $r_j - r_i$ to $x_{k'}$, and add $r_i - r_j$ to $x_{l'}$. Finally, subtract $d_i(r_k - r_l)$ from b. For an index i such that d_i is not anchored, i' is the ordinal position of d_i among the unanchored values or temperatures of $\boldsymbol{d}$. For instance, if only d_0 is anchored, $i' = 0$ when $i = 1$, $i' = 1$ when $i = 2$, and so on. When adding to the entries in $\boldsymbol{x}$, the indices i', j', k', and l' must be used instead of i, j, k, and l, since $\boldsymbol{x}$ has length m rather than n. The system only solves for the m unanchored values or temperatures in $\boldsymbol{d}$.

In general, then, building $\boldsymbol{x}$ and b is as follows: First set $\boldsymbol{x}$ to all zeros and b to zero. For i, add the quantity $r_k - r_l$ to $x_{i'}$ if d_i is unanchored or subtract d_i times that quantity from b if d_i is anchored. Do the same for j, k, and l with quantities $r_l - r_k$, $r_j - r_i$, and $r_i - r_j$, respectively.

The $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ part of the linear system similarly changes from $\lambda_{palette}\boldsymbol{Id} = \lambda_{palette}\boldsymbol{s'}$ to $\lambda_{palette}\boldsymbol{Iy} = \lambda_{palette}\boldsymbol{s'}$. Here, $\boldsymbol{y}$ is the same as in $\boldsymbol{Ry} = \boldsymbol{b}$. $\boldsymbol{s'}$ is a column of height m containing only the unanchored parts of $\boldsymbol{s}$ reordered to have the same ordering as the corresponding parts of $\boldsymbol{r}$. The system represents $d_i = s'_i$ for each i where d_i is not anchored.

For instance, if $n = 4$ and d_2 is anchored, the $PaletteCost(\boldsymbol{d}, \boldsymbol{s})$ part of the linear system is

$$\lambda_{palette} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_3 \end{bmatrix} = \lambda_{palette} \begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \end{bmatrix} \tag{2.24}$$

Notice that d_3, the third unanchored element in $\boldsymbol{d}$, corresponds with s'_2, the third element in $\boldsymbol{s'}$.

Combined, the two parts of the linear system form

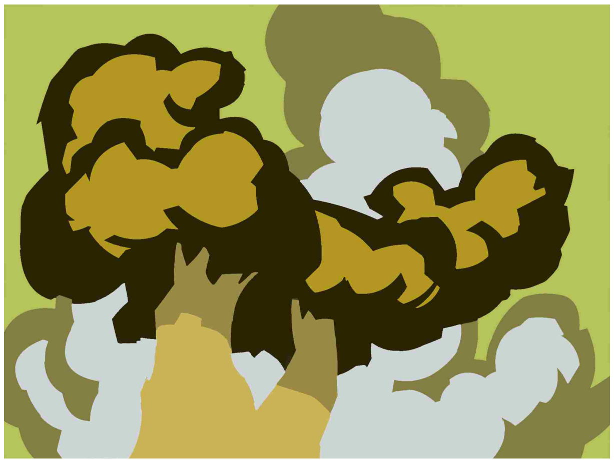
$$\begin{bmatrix} \boldsymbol{R} \\ \lambda_{palette}\boldsymbol{I} \end{bmatrix} \boldsymbol{y} = \begin{bmatrix} \boldsymbol{b} \\ \lambda_{palette}\boldsymbol{s'} \end{bmatrix} \tag{2.25}$$

After solving for $\boldsymbol{y}$, $d_i = y_{i'}$ for every i' in $[0, m-1]$.

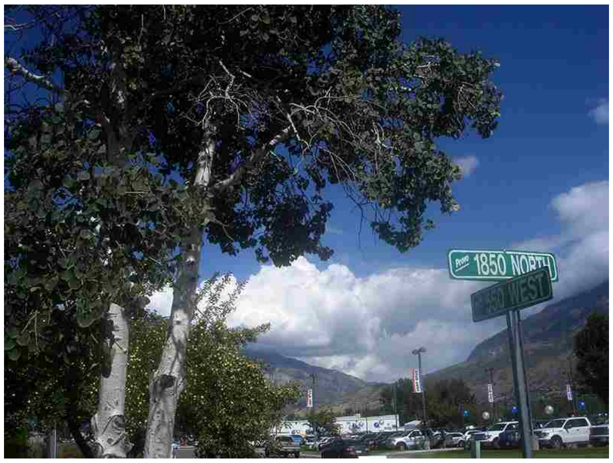
For example, when $n = 4$ and the third source value or temperature d_2 is anchored, the system looks like this:

$$\begin{bmatrix} r_1 - r_2 & r_2 - r_1 + r_1 - r_0 & 0 \\ r_1 - r_3 & r_3 - r_1 + r_1 - r_0 & r_0 - r_1 \\ r_2 - r_3 & r_3 - r_2 & r_0 - r_1 \\ 0 & r_2 - r_3 & r_1 - r_2 \\ \lambda_{palette} & 0 & 0 \\ 0 & \lambda_{palette} & 0 \\ 0 & 0 & \lambda_{palette} \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_3 \end{bmatrix} = \begin{bmatrix} -d_2(r_0 - r_1) \\ 0 \\ -d_2(r_1 - r_0) \\ -d_2(r_3 - r_2) - d_2(r_2 - r_1) \\ \lambda_{palette}\ s'_0 \\ \lambda_{palette}\ s'_1 \\ \lambda_{palette}\ s'_2 \end{bmatrix} \tag{2.26}$$

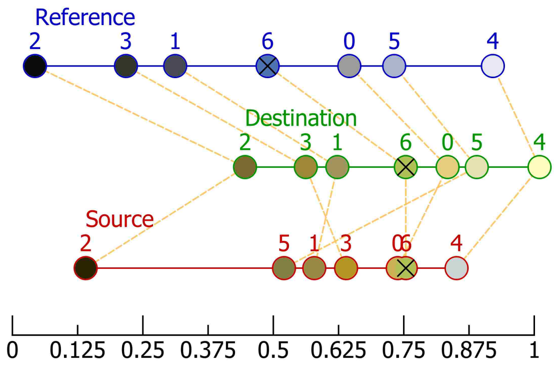
Figure 2.7 demonstrates anchoring. Anchors force the sky ($i = 6$) to preserve its original color; $d_6 = s_6$ in both value and temperature. Of course, just leaving the sky out of the transfer operation (not having any region cover it) would achieve the same effect. However, the sky would then not relate correctly with the rest of the image.



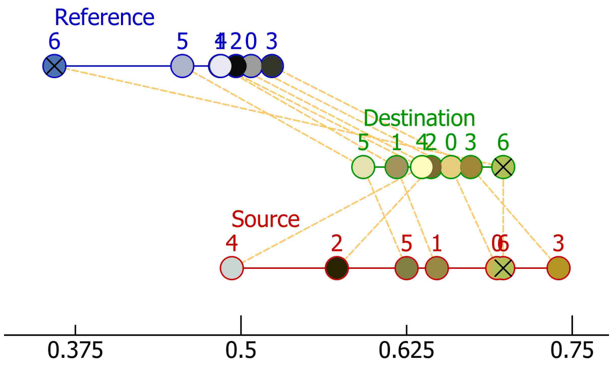
(a) Source



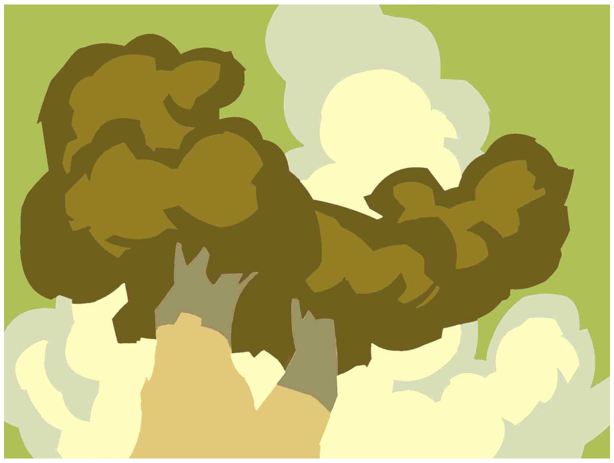
(b) Reference



(c) Values



(d) Temperatures



(e) Destination

Figure 2.7: Anchoring. The sky is region pair 6. Anchors make the sky in the destination image preserve its initial color in the source image; $d_6 = s_6$ in both value and temperature. This anchoring makes the sky in the destination image (e) have the same color as the sky in the source (a). The 'X's in the graphs indicate anchors (c,d).

Anchoring is useful for interactively refining the results of relationship transfer. However, it can limit the ability of relationship transfer to produce a correct $\boldsymbol{d}$. In Figure 2.7, for instance, the destination temperatures fail to have the same ordering as the reference temperatures. As per the reference, *sky* should be the coolest region, whereas it is the second warmest in the destination. The temperature anchor in this example causes $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ to come into too much conflict with $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$.

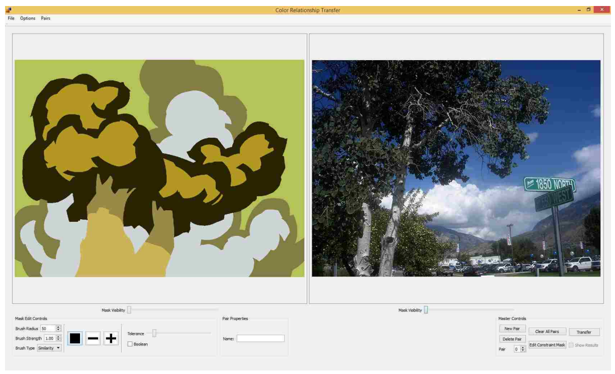
Anchoring offers a greater degree of user control at the expense of enforcing Properties 1–3. In some cases, anchoring can completely override Property 1, such as when the user creates two anchors that force d_i and d_j to have opposite ordering of r_i and r_j. In such a scenario, the optimal $\boldsymbol{d}$ might have the reverse ordering of $\boldsymbol{r}$.

Anchoring is not about realism; it is about refining aesthetics. If all values or temperatures are anchored, relationship transfer reduces to a local value and temperature editing operation. This can be useful as a final tuning step, when the user just wants to modify the value or temperature of a specific region without having the other regions' values or temperatures shift in response.

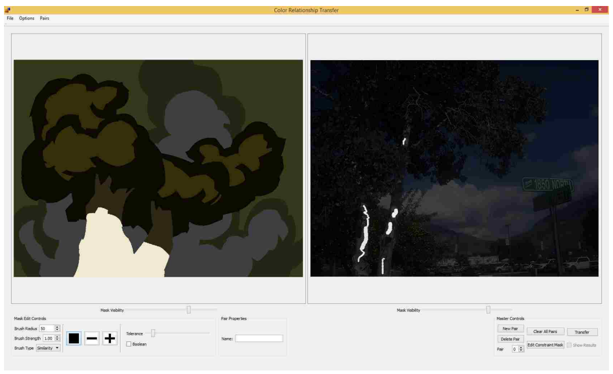
2.5 User Interface

Figure 2.8 shows the user interface for our implementation of relationship transfer. The source image appears on the left, and the reference image on the right. During transfer, the destination image appears in place of the source image.

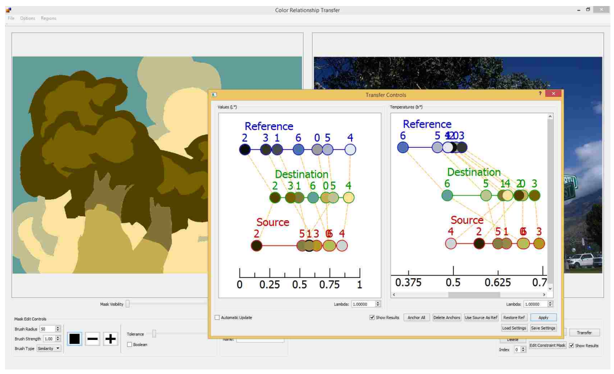
Using tools similar to those of Lischinski et al. [13], the user marks and refines region masks over the source and reference images. There are special, global tools for detailed source images, where region masks need to be very accurate. One of these is the "cooperative geodesics" tool, which performs a fuzzy n-way segmentation based on a multi-label extension to the fuzzy binary segmentation of Bai and Sapiro [2].



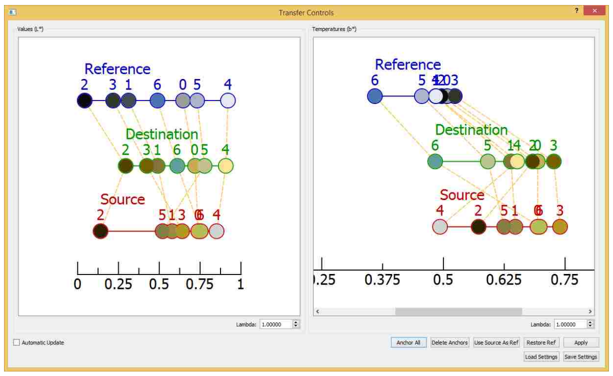
(a) Source and reference



(b) Region pair editing (*trunkLight* selected)



(c) Applying relationship transfer



(d) “Transfer Controls” dialog

Figure 2.8: User interface.

During transfer, a “Transfer Controls” dialog shows $\boldsymbol{s}$, $\boldsymbol{r}$, and $\boldsymbol{d}$ for values and temperatures. This dialog presents various controls for tuning the transfer operation. As the user interacts with the dialog, the destination image changes in the background.

To anchor or unanchor a value or temperature, the user right clicks on it in its $\boldsymbol{s}$, $\boldsymbol{r}$, or $\boldsymbol{d}$ graph. By dragging an anchored point in the $\boldsymbol{d}$ graph, the user can set an anchored value or temperature to any number in $[0, 1]$. As the user drags the anchored value or temperature, relationship transfer updates the other values or temperatures, as well as the destination image. The user can also click and drag in the reference graphs. This accomplishes “editing in the relationship domain”, which is described in Section 2.8.2.

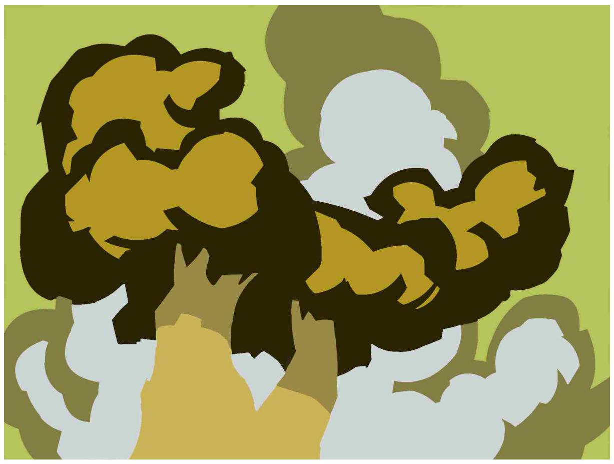
2.6 Using All Three Channels

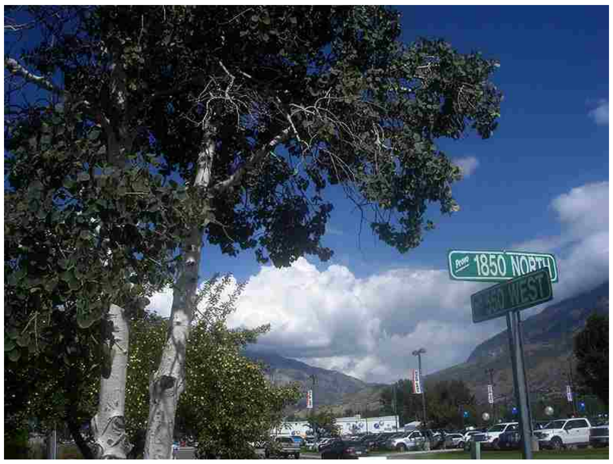
Editing values and temperatures only affects an image’s L* and b* channels respectively. The work described so far does not touch an image’s a* channel. Of course, relationship transfer can deal with a* the same way it deals with the other two channels. This is a departure from custom, since the a* channel does not correspond to any traditional idea in painting. For consistency, this research terms the a* channel the “anti-temperature” channel.

The user interface allows toggling all three channels independently. Figure 2.9 revisits the transfer scenario of Figures 2.4 and 2.5, showing results for different selections of channels.

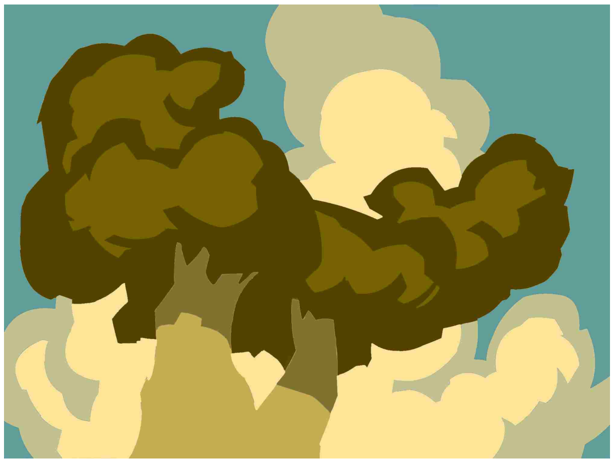
2.7 “Dots Mode”

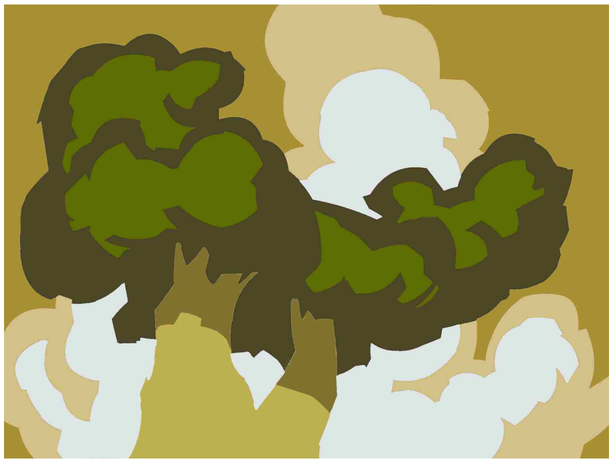
Dots mode is an alternative way for relationship transfer to help an artist with a painting. So far in this paper, relationship transfer has fixed a source painting by recoloring its regions to produce a destination painting. In principle, however, relationship transfer could just produce n destination colors and let the painter perform the recoloring step manually. The painter would not need to mark complete source regions, just representative color samples. In dots mode, the painter rapidly defines regions as variably sized dots using a simple click and drag interface. Relationship transfer generates a semitransparent image containing only


(a) Source image



(b) Reference image


(c) Value and temperature destination


(d) Value and anti-temperature destination

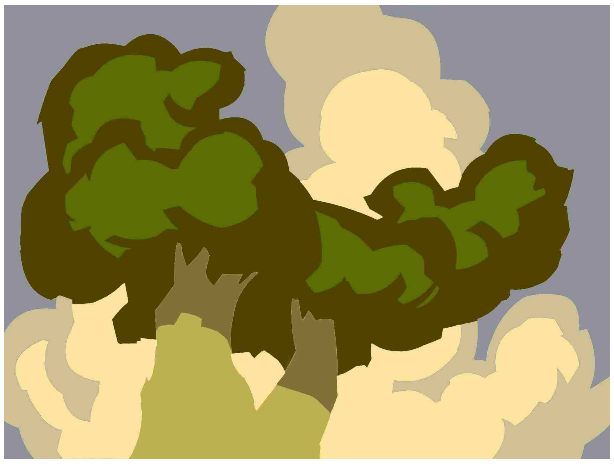

(e) Value, temperature, and anti-temperature destination

Figure 2.9: Channel selection. The user can choose which channels to treat out of value (L*), temperature (b*), and anti-temperature (a*). When transferring relationships to a painting, it is most traditional to treat only values and temperatures.

the destination dots. The painter loads the source painting along with the dots image in the painting program of choice. Then the painter can recolor the source painting at leisure, alternating between sampling the dots image and adding brushstrokes to the painting.

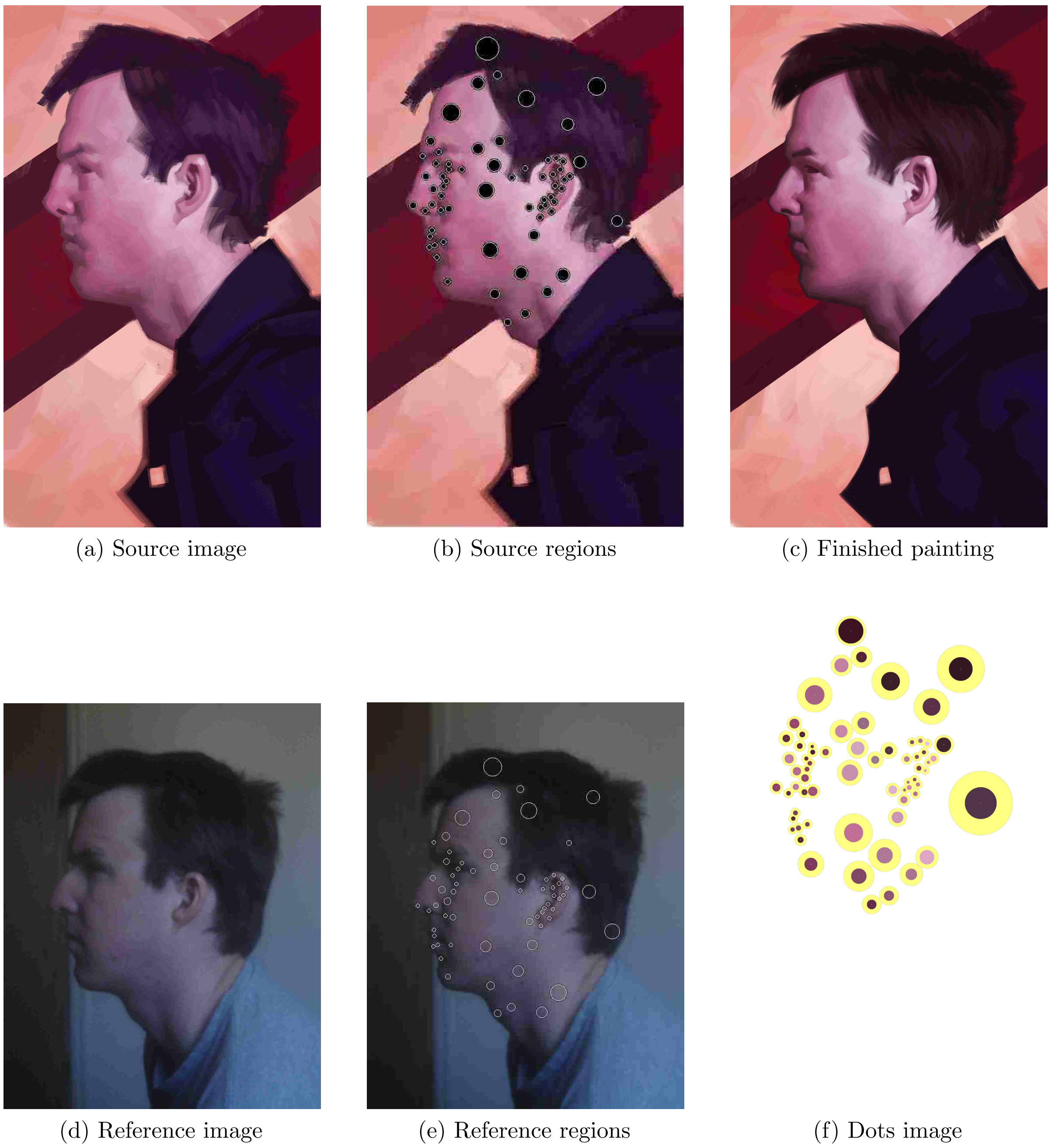
Figure 2.10 illustrates the process. The finished painting in this example is not the output of relationship transfer, but the product of manual brushwork. However, the finished painting does reflect the guidance given by relationship transfer via the dots image. The most noticeable impact is in the hair, which warms up significantly. Not all the color changes are due to relationship transfer; some come from the artist. The darkening of the shadows and overall increase in value contrast are the artist's own doing. The coat in the painting is not associated with the shirt in the reference photograph; relationship transfer only handles the internal relationships of the head and neck in this example.

Dots mode is ideal when the painting to fix does not partition easily. Unlike traditional relationship transfer, dots mode easily handles detailed paintings. This lets the artist benefit from relationship transfer when painting complex, difficult-to-segment subjects, particularly people. In dots mode, a region is extremely easy to define: the user need only click to define a center point and drag to define a radius. The only data required to represent a region are this center point and radius rather than an entire image mask. Thus, the user can deal with far more region pairs than would otherwise be tolerable or possible.

In fact, the only real problem with dots mode is that it encourages the user to drive n too high. Since the linear system that approximates each $\boldsymbol{d}$ vector grows at $\mathcal{O}(n^5)$, this can lead to heavy memory usage and long computation times.

2.8 Photograph Editing

Aside from adding realism to block-in paintings, relationship transfer has interesting photograph editing applications. Here, the goal is not adding realism to a source image, but changing its aesthetics. This is a deviation from the work shown so far, which tries leave a source image's aesthetic properties unchanged.


(a) Source image (b) Source regions (c) Finished painting

(d) Reference image (e) Reference regions (f) Dots image

Figure 2.10: “Dots mode”. The user defines regions as variably sized dots (b,e). Relationship transfer does not output a recolored painting, but a dots image (f) containing new colors for the painting. The user can sample from the dots image while bringing the painting to completion (c). In this example, the effects of relationship transfer are subtle. Most noticeable is a warming up of the hair.

In photographic applications, there is no separate reference image. The reference values and temperatures are either the source image's original values and temperatures, modified versions of these, or arbitrary numbers.

2.8.1 Photograph Anchoring

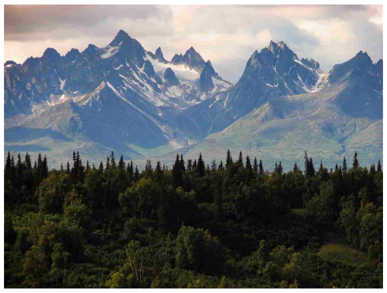
If the user does nothing more than make a source image its own reference, relationship transfer has virtually no effect. However, if the user also anchors destination values or temperatures to arbitrary numbers, the whole source image changes to preserve its original relationships. This allows a user to answer questions like, "What would a realistic depiction of this scene look like if the sky had temperature x and the trees had temperature y?"

Figure 2.11 shows three different temperature anchorings applied to a picture of an Alaskan landscape. In all three examples, $\boldsymbol{r} = \boldsymbol{s}$; the blue reference temperature graphs are identical to the red source temperature graphs. The three destination images try to preserve the source image's original relationships despite the anchorings applied.

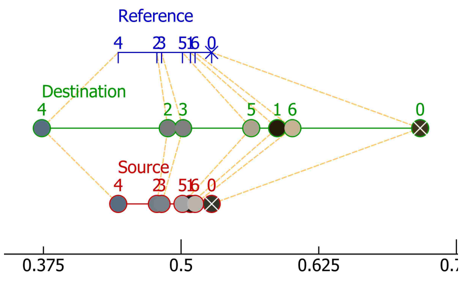
The first example anchors temperature d_0 to be significantly warmer than s_0. Region 0 corresponds to the lit side of the foreground trees, so the isolated effect of this anchoring is to warm up the lit side of the foreground trees. The best way for relationship transfer to meet this constraint while preserving original relationships and adhering to the original temperature palette is to stretch out the destination graph. Whereas region 0, the original warmest region, becomes warmer, region 4, the original coolest region, becomes cooler. Region 4 covers the blue background shadows.

The second example is the same as the first, but with an additional anchor of d_4 to a warmer value than s_4. Now that both the original warmest and original coolest regions are anchored to warmer temperatures, the entire image warms up.

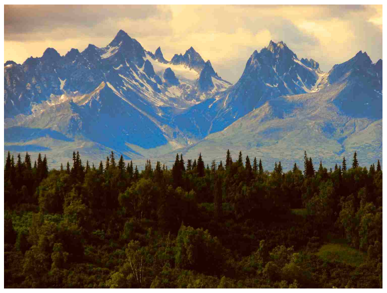
The third example is more aggressive. It anchors both d_4 and d_1, the temperature of the foreground shadows, such that d_1 is now cooler than d_4. In the original image, the foreground shadows are warmer than the background shadows, so the two anchors violate



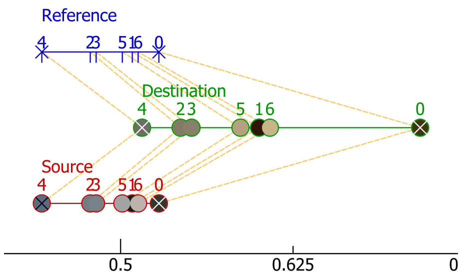
(a) Source



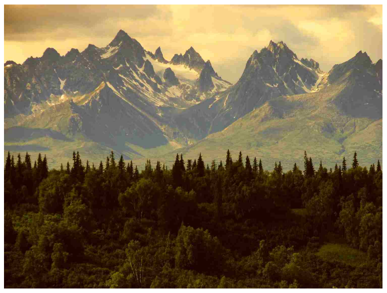
(b) Example 1 temperatures



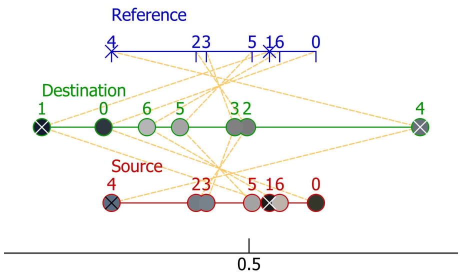
(c) Destination 1



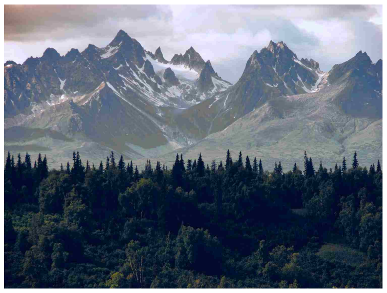
(d) Example 2 temperatures



(e) Destination 2



(f) Example 3 temperatures



(g) Destination 3

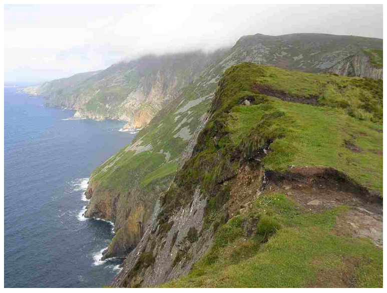
Figure 2.11: Photograph anchoring. Three different temperature anchorings (b,d,f) produce three different destination images (c,e,g). In all of these, the reference values and temperatures are equal to the source values and temperatures ($\boldsymbol{r} = \boldsymbol{s}$). This means that all the destination images try to preserve the relationships of the original image (a). In effect, the source image is its own reference image. Image courtesy of Skitterphoto (skitterphoto.com).

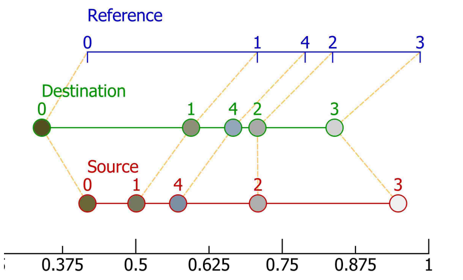
the ordering of the reference. The violation pushes $\boldsymbol{d}$ to have the reverse ordering of $\boldsymbol{r}$, in opposition to Property 1. As a rule, if the user anchors both d_i and d_j, then d_i and d_j should keep the same ordering as r_i and r_j.

The value of photograph anchoring is that unanchored values or temperatures automatically update to preserve a photograph's original relationships. Achieving this effect without relationship transfer would require the user to mentally track all a photograph's relationships while making local value or temperature adjustments.

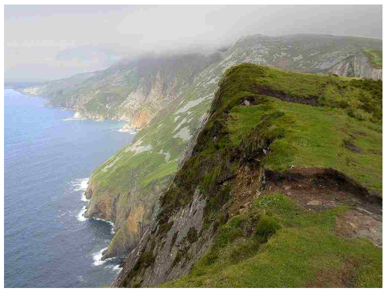
2.8.2 Editing in the Relationship Domain

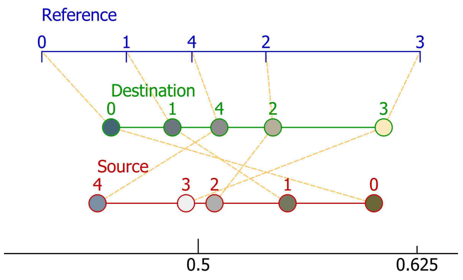
The "Transfer Controls" dialog lets the user define arbitrary reference values and temperatures by clicking and dragging in the reference graphs. The effect is to replace an image's relationships with new ones.


(a) Source



(b) Example 1 values


(c) Destination 1



(d) Example 2 temperatures

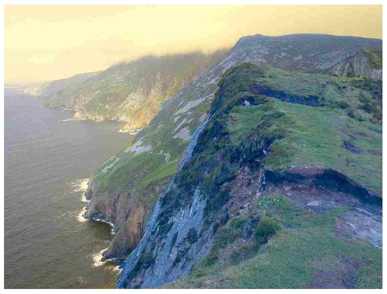

(e) Destination 2

Figure 2.12: Editing in the relationship domain. By constructing an arbitrary $\boldsymbol{r}$ (blue graph) for values or temperatures, a user can directly edit a photograph's relationships. The first example edits value relationships to make the foreground cliffs stand out. The second example edits temperature relationships so that objects get warmer rather than cooler as they recede into the background. Image courtesy of pdphoto.org.

Figure 2.12 shows two examples of editing in the relationship domain, both applied to a seaside landscape. There are 5 regions:

0. Foreground cliff
1. Midground cliffs
2. Background cliffs
3. Clouds
4. Water

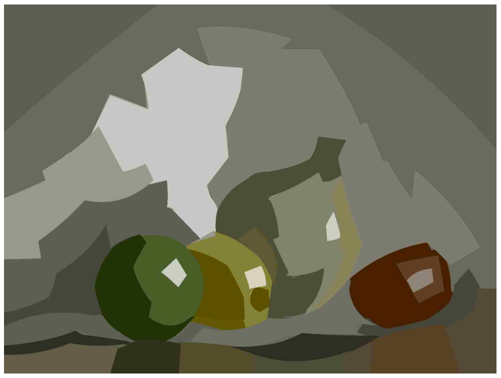
The first example edits value relationships, while the second edits temperature relationships. In both examples, the reference relationships are the user's invention. They are not the relationships of a separate reference image, nor of the source image.

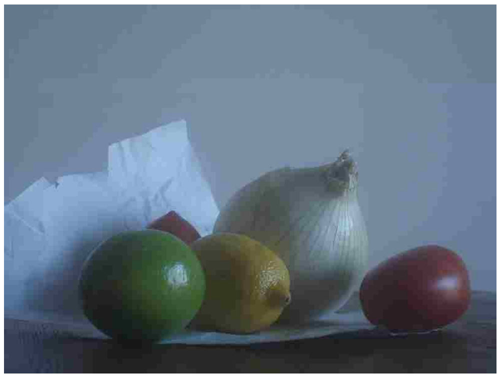
Editing in the relationship domain varies from adjusting the relative spacing between two values or temperatures to giving an image an entirely new set of relationships. The first example adds depth to the scene by making the interval between the values of region 0 and region 1 wider than the other intervals in the reference graph. This makes the foreground cliff stand out from its background, since the value difference between it and the cliffs behind is now more pronounced than the other value differences in the scene. The second example is more radical. It reverses most of the scene's temperature orderings so that objects get warmer rather than cooler as they recede into the background.

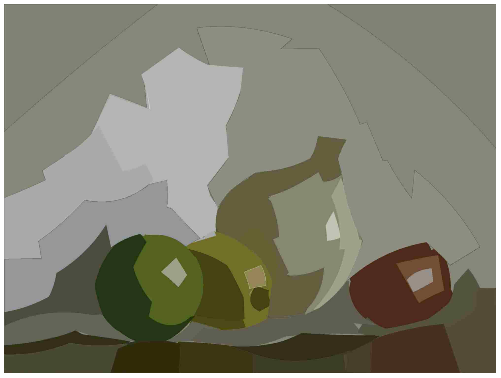
2.9 Additional Results

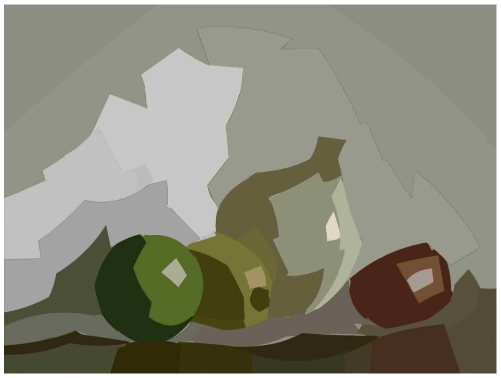
2.9.1 Paintings

Figure 2.13, like the foregoing Figure 2.7, shows how anchors can refine the results of relationship transfer. In Figure 2.7, a value anchor and a temperature anchor make the sky in a painting retain its original color. In Figure 2.13, the user applies a single value anchor to a piece of paper in a still life block-in to make the paper brighter. Unlike in Figure 2.7, the


(a) Source


(b) Reference


(c) Bad destination


(d) Fixed destination

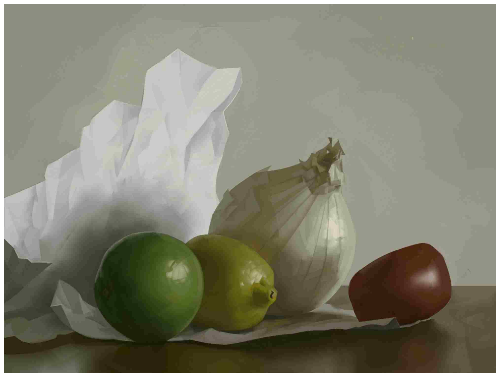
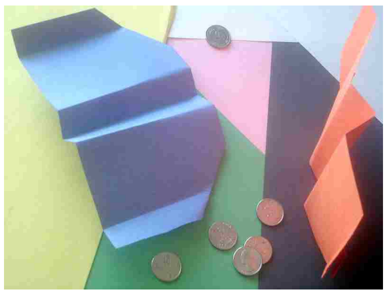
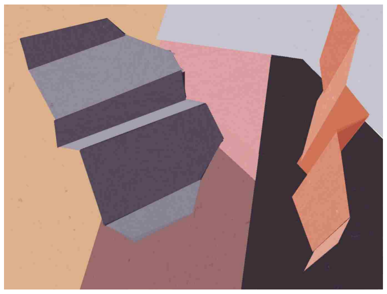
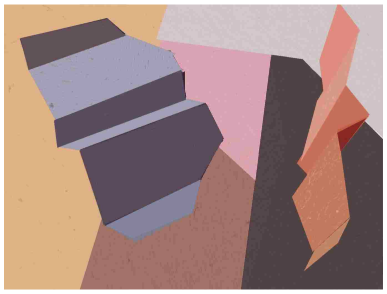
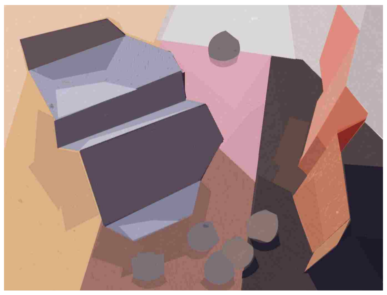

(e) Final painting

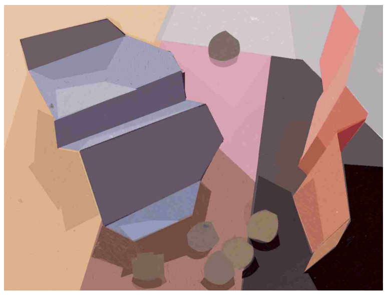
Figure 2.13: Refining results. In this example, relationship transfer recolors a still life block-in (a). In the initial destination image (c), the paper is too dark and does not stand out enough from its background. By anchoring the paper's value and dragging it higher in the value graph, the user manually lightens the paper. The other regions adjust automatically, producing a better destination image (d). The user manually adds detail to produce the final painting (e).


(a) Reference


(b) First before


(c) First after


(d) Second before


(e) Second after

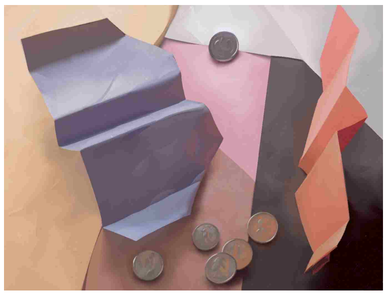

(f) Final painting

Figure 2.14: Multiple relationship transfers. An artist can apply relationship transfer multiple times during a painting's lifetime. In this still life example, the first application occurs at a coarse block-in stage (b,c). Here, relationship transfer makes subtle value corrections, most noticeable in the black paper. The second application occurs at a later, slightly more detailed stage (d,e). The second application warms the temperature of the coins and darkens the shadow on the black paper. Both applications contribute to the realism of the final, detailed painting (f).

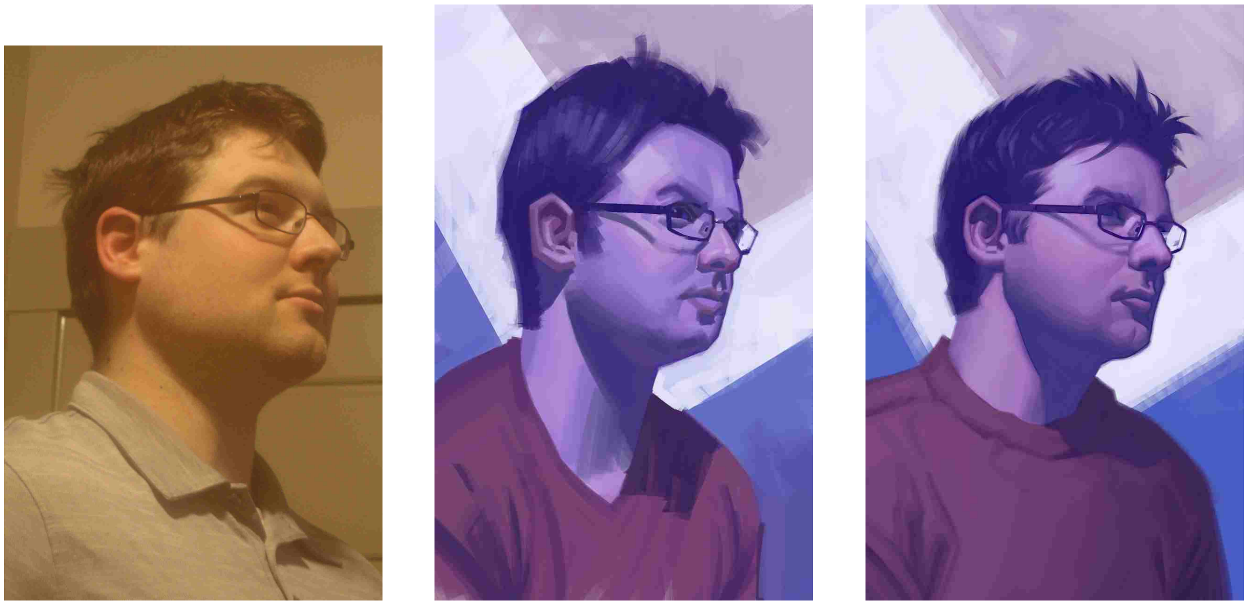

(a) Reference (b) Source (c) Final painting

Figure 2.15: Dots mode flexibility. The color alterations in the subject's face between the source painting (b) and the finished painting (c) come from relationship transfer, while the spatial corrections come from the artist. This shows the flexibility of dots mode: an artist can get help with color choices while still working out a painting's structural details.

user does not force the paper to have its original value from the source image. The user just anchors and drags the paper's value to the right, making it arbitrarily lighter. As the user drags the value higher, the destination image automatically recolors in response.

Another way to get more out of relationship transfer is to apply it multiple times during a painting's development. The example in Figure 2.14 invokes relationship transfer twice: once at a coarse stage and again at a finer stage.

Figure 2.15 shows another portrait painting aided by relationship transfer's dots mode. In this example, relationship transfer provides corrected colors while the painting still has numerous spatial errors. This highlights the flexibility of dots mode in an artist's workflow.

2.9.2 Full-Color Anchoring

Section 2.6 discusses performing relationship transfer on one or more of the available color channels: value (L*), temperature (b*), and "anti-temperature" (a*). When all three channels

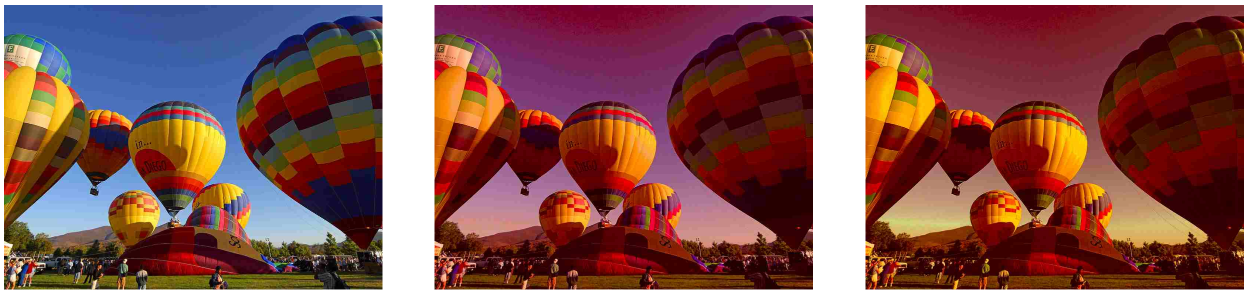

(a) Source (b) First anchor (c) Second anchor

Figure 2.16: Two full-color anchors. With all color channels active, the user can anchor parts of the source image (a) to new colors specified as RGB or HSV triplets. Here, the user applies two full-color anchors together. The first forces the top of the sky to be bright red (b). The second forces the bottom of the sky to be a golden color (c). As the user applies these anchors, the rest of the image automatically adjusts to preserve the source image's original relationships in the L*, a*, and b* channels. Image courtesy of pdphoto.org.

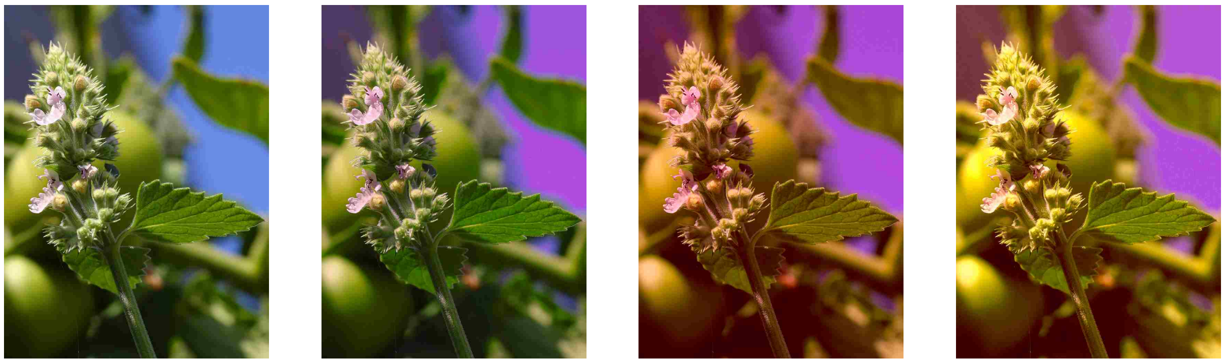

(a) Source (b) First anchor (c) Second anchor (d) Third anchor

Figure 2.17: Three full-color anchors. As in Figure 2.16, the user applies multiple, cooperating full-color anchors to a photograph (a). The first anchor forces the light part of the background sky to be purple (b). The second makes the shadows red (c). The third turns the highlights in the background a strong yellow (d). Image courtesy of pdphoto.org.

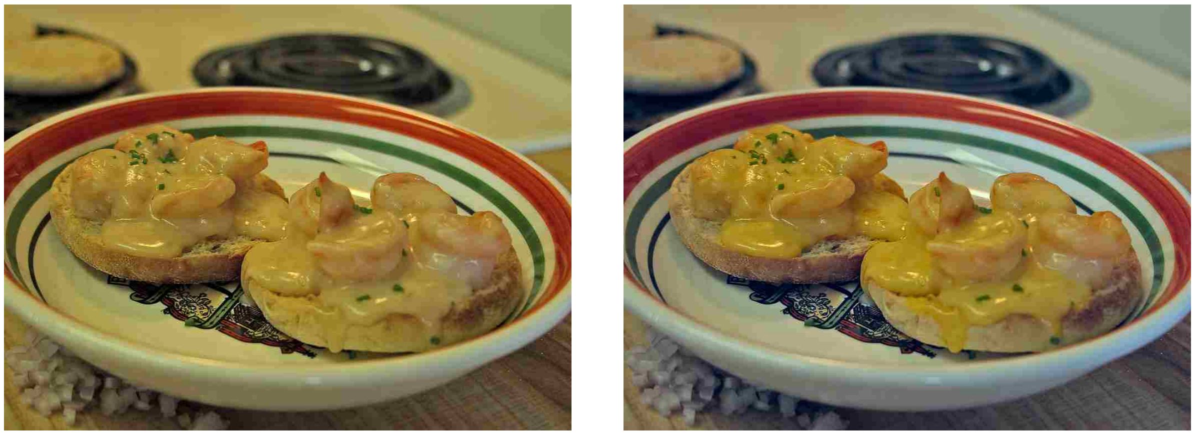
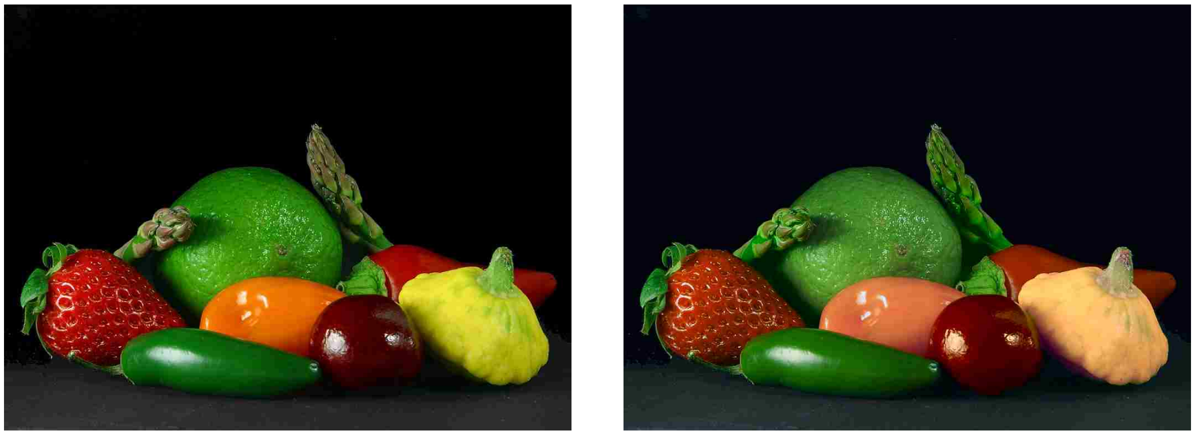

(a) Source (b) Destination

Figure 2.18: Editing in the relationship domain: enhancing appeal. The user assigns arbitrary new temperature relationships to the initial image (a) to create a more appealing result (b). The new temperature relationships dictate that the temperatures within the food should be far apart relative to the other temperatures in the scene. The effect is to make the food take on a high temperature contrast relative to the rest of the image. In addition, the user applies a cooling anchor to the temperature of the plate, increasing the temperature range of the destination image and making the new relationships more apparent. Image courtesy of pdphoto.org.

are active, a useful feature becomes available: the user can anchor parts of an image to desired colors. The user inputs a full-color anchor as an RGB or HSV triplet. This triplet is converted to three anchors: one for each channel. Full-color anchoring is a very intuitive way to adjust a photograph's aesthetics while preserving its original relationships. The technique is especially powerful when multiple full-color anchors apply at the same time, as in Figures 2.16 and 2.17.

2.9.3 Editing in the Relationship Domain

Figures 2.18 and 2.19 show two ways for a user to aesthetically modify a photograph by assigning it customized new relationships. The first example assigns new temperature relationships to a photograph of some food to make the food have higher temperature contrast than the rest of the image. The second example assigns new temperature and anti-temperature relationships to a photograph of some produce to force the image's colors to collapse into two groups.



(a) Source (b) Destination

Figure 2.19: Editing in the relationship domain: artistic simplification. The initial photograph (a) is colorfully complex. To subdue the image, the user polarizes its colors into two groups. The user assigns new relationships in both the temperature and anti-temperature channels. In both channels, the new relationships have the green colors collapse into one group and the other colors collapse into a second group. The destination image (b) shows the resulting color dichotomy. Image courtesy of pdphoto.org.

In the first example, the user not only applies new temperature relationships, but also applies a cooling anchor to the temperature of the plate. The anchor boosts the temperature range of the destination image, making the new temperature relationships more obvious. This shows how anchoring and editing in the relationship domain can work together.

2.10 Conclusions and Future Work

Relationship transfer meets its original goal: to transfer relationships from reference photographs to digital paintings. This suggests that relationship transfer might be successful in other painting applications. For instance, it could enable art teachers to objectively critique still life exercises done in oil or acrylics. This would involve comparing photographs of students' work with students' reference photography. There may also be more advanced ways for relationship transfer to aid in digital painting. For example, relationship transfer could integrate into a painting program like Adobe's *Photoshop* or Corel's *Painter* to provide a "history sensitive" paint brush. This brush would automatically tweak its user-input color to make the next brushstroke relate more correctly with previous brushstrokes.

Yet the real value of this research is the more general notion of editing an image in terms of value and temperature relationships. The idea has major implications for computational photography in particular. This paper does show an initial foray into photography, but the examples are limited by crude, research-quality region editing tools. A more sophisticated but easy-to-use mask editing suite would better demonstrate photograph anchoring and editing photographs in the relationship domain. Relationship transfer might even utilize automatic image decomposition, such as that of Price and Barrett [24] or of Reese and Barrett [25]. At an interesting extreme, each pixel in a photograph could be its own region. This would fully realize the concept of anchoring, with an image preserving its inter-pixel relationships in response to totally arbitrary local edits.

Whether or not relationship transfer receives future attention, it clearly has potential well beyond transferring relationships from a reference photograph to a digital painting.

Chapter 3

Conclusions and Future Work

Relationship transfer does well at transferring relationships from reference photographs to paintings, and initial photography work is promising. However, the technology has room to grow. With more development, there is potential for more art-related applications, as well as interesting photography applications beyond those demonstrated here. In the end, though, the real contribution of relationship transfer is not a specific application. Rather, it is the idea of editing an image with respect to its internal relationships.

3.1 Weaknesses

Relationship transfer's major current weakness is $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ from Equation 2.8:

$$\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r}) = \sum_{i,j,k,l} ((d_i - d_j)(r_k - r_l) - (d_k - d_l)(r_i - r_j))^2 \mid i < j, k < l \qquad (3.1)$$

$\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ on its own should enforce both relationship properties: Properties 1 and 2. However, it does not strictly enforce Property 1. Since $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ will allow a destination vector $\boldsymbol{d}$ to have the opposite ordering of $\boldsymbol{r}$, $\mathit{Cost}(\boldsymbol{d})$ relies implicitly on $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$ to enforce Property 1.

This can be a problem when transferring relationships from a photograph to a painting in the presence of multiple anchors. Just two anchors can prevent $\mathit{Cost}(\boldsymbol{d})$ from yielding a correct $\boldsymbol{d}$ if they are out of order with respect to $\boldsymbol{r}$. In such a scenario, $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ will favor a $\boldsymbol{d}$ that reverses the ordering of $\boldsymbol{r}$, which puts $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ into unusual conflict with $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$. At best, $\lambda_{palette}$ will be high enough that $\boldsymbol{d}$ preserves the ordering of $\boldsymbol{r}$, but with compressed range. At worst, $\boldsymbol{d}$ will have the reverse ordering of $\boldsymbol{r}$.

The issue is especially problematic for photograph editing, where palette preservation is relatively unimportant. In photograph anchoring, for example, a user might like to reduce $\lambda_{palette}$ to zero, apply two value anchors, and see a destination image that reproduces original relationships as much as possible. In this scenario, the result $\boldsymbol{d}$ will have the reverse ordering of $\boldsymbol{r}$ if the two anchors are out of order with respect to $\boldsymbol{r}$.

Future work in relationship transfer that bases in Properties 1 and 2 should use a different definition of $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$. It should not be necessary for some notion of palette preservation to help enforce relationship properties, since relationships and palette preservation are essentially orthogonal concepts.

3.2 Future Work

Relationship transfer does well at transferring relationships from a reference photograph to a digital painting. This is not to say that the work is done where art is concerned. There might be better ways to quantify value and temperature, or to characterize crucial relationships. There could also be new painting applications. Yet the real value of this research, and the most likely focus of subsequent work, is not about painting. It is not even necessarily about transferring relationships between images. Rather, it is the general notion of editing an image in terms of, or with respect to, its internal relationships. This idea has major future implications for computational photography in particular.

Section 2.3 makes strong assumptions about which intra-image value and temperature relationships are most crucial. Properties 1 and 2 come from intuition, not solid literature. Study of graphic design, fine art, and perceptual psychology could produce a better set of fundamental relationship properties.

Similarly, there is nothing authoritative about using the L* and b* channels to quantify value and temperature. Bruce MacEvoy explains that there are other ways to formulate color temperature [18], and there are clearly more ways than one to represent value.

Along with rethinking relationship transfer's formulation, future work might explore new art-related applications. For instance, relationship transfer has potential as a diagnostic tool in art education. Here, the goal would not be to automatically fix a painting, but to teach a student how to fix a painting. College-level art courses frequently have students make still lifes (digital or traditional) from a reference photograph. Teachers must compare students' attempts with their reference and make objective criticisms. This is difficult for a human to do on the spot. Relationship transfer could step in here to analyze the relationship problems in student work.

The primary needed innovation would be a human-readable explanation of the journey from the source $\boldsymbol{s}$ to the destination $\boldsymbol{d}$ for values and temperatures. The graphs in the "Transfer Controls" dialog already show what a student would need to do to fix a painting. However, they are difficult to read, especially for a nontechnical audience. A more useful output could be a series of instructions, such as "First make the tree warmer. Then make the sky darker until it has about the same value as the rocks."

Despite the potential utility of such a tool, the most likely next applications of relationship transfer are in computational photography. This research only goes a short way in that direction because its mask editing tools are crude and unoptimized. Decomposing a photograph with them is tedious, which is why the photography examples shown are so simple. A trivial next step for relationship transfer would be coupling it with faster, more photograph-friendly mask selection tools.

A longer term goal would be to challenge a hitherto tacit assumption: that the regions of a scene's semantic decomposition should be homogeneous in color. This assumption rarely holds in photographs. A user may want to identify a car in a picture as a single semantic element, but this will typically violate the homogeneity assumption, and results will look incorrect unless the user chooses a denser semantic decomposition that treats the car's wheels, doors, windshield, etc. as separate components. One solution would be a semi-automatic decomposition mechanism where a user selects n super-regions, each of which automatically splits into multiple sub-regions. Another solution would be to simply abandon the homogeneity assumption. This would blur the current temporal separation between determining new values and temperatures and creating the destination image.

Other future work might abandon the idea of "transfer" altogether and focus only on relationship-preserving image editing. In this context, the user does not necessarily need to manually decompose the image into regions. The decomposition could be automatic, using methods like those of Price and Barrett [24] or of Reese and Barrett [25]. At an extreme, each pixel could be its own region. This would mean preserving the relationships between every two pixels in an image. Such an intensely computational process would require clever optimization.

Whether or not any work follows this research, relationship transfer definitely has potential beyond transferring relationships from a photograph to a digital painting.

Appendix A

Minimizing *RelateCost*$(\boldsymbol{d}, \boldsymbol{r})$ Alone

A.1 Goal

The goal is to prove that the vector $\boldsymbol{d}$ which minimizes $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ must have the exact same or exact opposite ordering of the fixed reference vector $\boldsymbol{r}$. The proof considers $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ alone; it does not involve $\mathit{PaletteCost}(\boldsymbol{d}, \boldsymbol{s})$. $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ comes from Equation 2.8:

$$\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r}) = \sum_{i,j,k,l} ((d_i - d_j)(r_k - r_l) - (d_k - d_l)(r_i - r_j))^2 \mid i < j, k < l \tag{A.1}$$

$\boldsymbol{d}$ has the same ordering as $\boldsymbol{r}$ when $d_i - d_j$ has the same sign as $r_i - r_j$ for every i and j, where $i \neq j$. $\boldsymbol{d}$ has the opposite ordering of $\boldsymbol{r}$ when $d_i - d_j$ has the opposite sign of $r_i - r_j$ for every i and j, where $i \neq j$.

An initial assumption states that neither $\boldsymbol{d}$ nor $\boldsymbol{r}$ have duplicate values. For every i and j such that $i \neq j$, $d_i \neq d_j$ and $r_i \neq r_j$. This offers an easy initial proof. The assumption does not hold in practice, of course, but it does not need to, according to the definition of "ordering".

A.2 No-Duplicates Proof

$\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ is a sum of squares, one square for each selection of indices i, j, k, and l such that $i < j$ and $k < l$. A sum of squares cannot have a value below zero. So the best possible value for $\mathit{RelateCost}(\boldsymbol{d}, \boldsymbol{r})$ is zero.

Trivial algebra shows that any linear transformation of $\boldsymbol{r}$ produces an optimal $\boldsymbol{d}$. For any constants α and β, $\boldsymbol{d} = \alpha\boldsymbol{r} + \beta$ will set $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ to zero. Thus, there is a hyperplane of optimal $\boldsymbol{d}$ selections. It is unnecessary to say that any optimal $\boldsymbol{d}$ must fall in this plane. What matters is that because there exists at least one $\boldsymbol{d}$ that sets $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ to zero, any $\boldsymbol{d}$ which does not set $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ to zero is suboptimal.

If $\boldsymbol{d}$ does not have the same ordering as $\boldsymbol{r}$ or the opposite ordering of $\boldsymbol{r}$, then there must be some selection of indices i, j, k, and l such that d_i and d_j are in the same order as r_i and r_j but d_k and d_l are in the opposite order of r_k and r_l. For this selection of indices, the term added to $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ is

$$((d_i - d_j)(r_k - r_l) - (d_k - d_l)(r_i - r_j))^2 \tag{A.2}$$

$d_i - d_j$ has the same sign as $r_i - r_j$ and $d_k - d_l$ has the opposite sign of $r_k - r_l$. This means that $(d_i - d_j)(r_k - r_l)$ has the opposite sign of $(d_k - d_l)(r_i - r_j)$. Neither of these is equal to zero since there are no duplicates in $\boldsymbol{d}$ or $\boldsymbol{r}$. This means

$$((d_i - d_j)(r_k - r_l) - (d_k - d_l)(r_i - r_j))^2 > 0 \tag{A.3}$$

$RelateCost(\boldsymbol{d}, \boldsymbol{r})$ is therefore greater than zero. Therefore, $\boldsymbol{d}$ is suboptimal.

A.3 Duplicates Proof

An extension to the previous proof shows that an optimal $\boldsymbol{d}$ must have the same or opposite ordering of $\boldsymbol{r}$ even when both $\boldsymbol{d}$ and $\boldsymbol{r}$ are allowed to contain duplicate numbers. The extension to the proof is more intuitive than formal.

An acceptable definition of "ordering" states that a and b have the same ordering *and* the opposite ordering of c and d if either $a = b$ or $c = d$. In general, if n points on a line graph start out with some arbitrary ordering and then collapse to a single point, that single point presumably contains the n initial points in their original order, as well as any other possible order.

Any m numbers ($m <= n$) in $\boldsymbol{r}$ or $\boldsymbol{d}$ that have the same value also have any possible ordering with respect to each other. More to the point, they have the convenient ordering relative to the assessment of $\boldsymbol{d}$ as having the same ordering or the opposite ordering of $\boldsymbol{r}$.

- If $r_i = r_j$, then d_i and d_j have both the same ordering and the opposite ordering of r_i and r_j.
- If $d_i = d_j$, then d_i and d_j have both the same ordering and the opposite ordering of r_i and r_j.

This means that $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ has many trivial optima. For instance, any vector of n identical values will yield $RelateCost(\boldsymbol{d}, \boldsymbol{r}) = 0$. Which optimal $\boldsymbol{d}$ gets chosen is not pertinent here. The point is just to show that any $\boldsymbol{d}$ that does not have the same ordering or the opposite ordering of $\boldsymbol{r}$ is suboptimal with respect to $RelateCost(\boldsymbol{d}, \boldsymbol{r})$ alone.

References

[1] Xiaobo An and Fabio Pellacini. User-controllable color transfer. In *Computer Graphics Forum*, volume 29, pages 263–271. Wiley Online Library, 2010.

[2] Xue Bai and Guillermo Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE 11th International Conference on Computer Vision (ICCV 2007)*, pages 1–8. IEEE, 2007.

[3] Nicolas Bonneel, Kalyan Sunkavalli, Sylvain Paris, and Hanspeter Pfister. Example-based video color grading. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)*, 32(4):2, 2013.

[4] Youngha Chang, Suguru Saito, and Masayuki Nakajima. Color transformation based on the basic color categories of a painting. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, pages 157–157. ACM, 2002.

[5] Youngha Chang, Keiji Uchikawa, and Suguru Saito. Example-based color stylization based on categorical perception. In *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, pages 91–98. ACM, 2004.

[6] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. Color harmonization. *ACM Transactions on Graphics (TOG)*, 25(3):624–630, 2006.

[7] Stephen J DiVerdi, Sunil Hadap, and Aravind Krishnaswamy. System and method for simulation of paint deposition using a pickup and reservoir model, June 11 2013. US Patent 8,462,173.

[8] Stephen J DiVerdi, Aravind Krishnaswamy, Jerry G Harris, Sunil Hadap, and Walter Michael Shaw. System and method for simulating paint brush strokes using configurable wetness, drying, and mixing parameters, December 3 2013. US Patent 8,599,213.

[9] Gary R Greenfield and Donald H House. A palette-driven approach to image color transfer. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 91–99. Eurographics Association, 2005.

[10] Hua Huang, Yu Zang, and Chen-Feng Li. Example-based painting guided by color features. *The Visual Computer*, 26(6-8):933–942, 2010.

[11] Steven G. Johnson. The NLopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`.

[12] Meng-Tsan Li, Ming-Long Huang, and Chung-Ming Wang. Example-based color alternation for images. In *2nd International Conference on Computer Engineering and Technology (ICCET 2010)*, volume 7, pages V7–316. IEEE, 2010.

[13] Dani Lischinski, Zeev Farbman, Matt Uyttendaele, and Richard Szeliski. Interactive local adjustment of tonal values. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 646–653. ACM, 2006.

[14] Shiguang Liu, Hanqiu Sun, and Xiang Zhang. Selective color transferring via ellipsoid color mixture map. *Journal of Visual Communication and Image Representation*, 23(1): 173–181, 2012.

[15] Andrew Loomis. *Creative illustration*. Titan Books, London, 2012. ISBN 978-1845769284.

[16] Jingwan Lu, Connelly Barnes, Stephen DiVerdi, and Adam Finkelstein. Realbrush: Painting with examples of physical media. *ACM Transactions on Graphics (TOG)*, 32 (4):117, 2013.

[17] Qing Luan, Fang Wen, and Ying-Qing Xu. Color transfer brush. In *Pacific Conference on Computer Graphics and Applications*, pages 465–468, 2007.

[18] Bruce MacEvoy. Color temperature. `http://www.handprint.com/HP/WCL/color12.html`, 2009.

[19] L Neumann, M Sbert, B Gooch, and W Purgathofer. Color search and replace. In *Computational Aesthetics 2005: Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging, Girona, Spain*, page 101. AK Peters, 2005.

[20] F Pitié and A Kokaram. The linear monge-kantorovitch linear colour mapping for example-based colour transfer. In *Visual Media Production, 4th European Conference on Visual Media Production, London, UK*, pages 1–9, 2007.

[21] Francois Pitie, Anil C Kokaram, and Rozenn Dahyot. N-dimensional probability density function transfer and its application to color transfer. In *Tenth IEEE International Conference on Computer Vision (ICCV 2005)*, volume 2, pages 1434–1439. IEEE, 2005.

[22] Tania Pouli and Erik Reinhard. Progressive color transfer for images of arbitrary dynamic range. *Computers & Graphics*, 35(1):67–80, 2011.

[23] Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009.

[24] Brian Price and William Barrett. Object-based vectorization for interactive image editing. *The Visual Computer*, 22(9-11):661–670, 2006.

[25] LJ Reese and WA Barrett. Image editing with intelligent paint. In *Proceedings of Eurographics*, volume 21, pages 714–724, 2002.

[26] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.

[27] Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 747–754. IEEE, 2005.

[28] Tom Van Laerhoven, Jori Liesenborgs, and Frank Van Reeth. Real-time watercolor painting on a distributed paper model. In *Computer Graphics International*, pages 640–643. IEEE, 2004.

[29] Peter Vandoren, Tom Van Laerhoven, Luc Claesen, Johannes Taelman, Chris Raymaekers, and Frank Van Reeth. Intupaint: Bridging the gap between physical and digital painting. In *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems, TABLETOP*, pages 65–72. IEEE, 2008.

[30] Baoyuan Wang, Yizhou Yu, Tien-Tsin Wong, Chun Chen, and Ying-Qing Xu. Data-driven image color theme enhancement. *ACM Transactions on Graphics (TOG)*, 29(6): 146, 2010.

[31] Chung-Lin Wen, Chang-Hsi Hsieh, Bing-Yu Chen, and Ming Ouhyoung. Example-based multiple local color transfer by strokes. In *Computer Graphics Forum*, volume 27, pages 1765–1772. Wiley Online Library, 2008.

[32] Fuzhang Wu, Weiming Dong, Yan Kong, Xing Mei, Jean-Claude Paul, and Xiaopeng Zhang. Content-based colour transfer. In *Computer Graphics Forum*, volume 32, pages 190–203. Wiley Online Library, 2013.

[33] Jae-Doug Yoo, Min-Ki Park, Ji-Ho Cho, and Kwan H Lee. Local color transfer between images using dominant colors. *Journal of Electronic Imaging*, 22(3):033003, 2013.

[34] Xiaoyan Zhang, Martin Constable, and Ying He. On the transfer of painting style to photographic images through attention to colour contrast. In *Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, pages 414–421. IEEE, 2010.

[35] Xiaoyan Zhang, Kap Luk Chan, and Martin Constable. Depth-based reference portrait painting selection for example-based rendering. In *IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS)*, pages 105–110. IEEE, 2011.

[36] Xiaoyan Zhang, Martin Constable, and Kap Luk Chan. Aesthetic enhancement of landscape photographs as informed by paintings across depth layers. In *18th IEEE International Conference on Image Processing (ICIP)*, pages 1113–1116. IEEE, 2011.

[37] Xiaoyan Zhang, Martin Constable, and Kap Luk Chan. Example-based contrast enhancement for portrait photograph. In *21st International Conference on Pattern Recognition (ICPR)*, pages 943–946. IEEE, 2012.

[38] Xiaoyan Zhang, Kap Luk Chan, and Martin Constable. Atmospheric perspective effect enhancement of landscape photographs through depth-aware contrast manipulation. *IEEE Transactions on Multimedia*, 16(3):653–667, 2014.