Image Recoloring Induced by Palette Color Associations

Gary R. Greenfield
Department of Mathematics & Computer Science
University of Richmond
Richmond, VA 23173, U.S.A.
ggreenfi@richmond.edu

Donald H. House
Visualization Laboratory
Texas A&M University
College Station, TX 77843, U.S.A
house@viz.tamu.ed

ABSTRACT

In this paper we present a non-interactive method for recoloring a destination image according to the color scheme found in a source image. The approach is motivated by trying to invert the working process employed in oil painting, and results are demonstrated by application to several well-known oil paintings. The algorithm uses several color models, but leans most heavily on the $L\alpha\beta$ color space. We first color segment each image bottom-up by iteratively merging groups of pixels into connected regions of similar color. During color segmentation, a color "texture" tree is generated and associated to each region. Next, we construct classes of regions by compensating for color duplication and color similarity within the set of averaged color values obtained from regions. We extract a color palette for each image by choosing the colors of canonical region representatives from these classes. Once this palette is constructed for each image, any inverse map from the set of destination palette colors to the set of source palette colors induces a forward map from the classes of regions in the source image to sets of classes of regions in the destination image. For each source class in the range of the inverse map we transfer color from its canonical region representative to each of the associated destination regions. Color transfer occurs at the level of pixels, and uses the color texture trees associated to the regions. Our recoloring method attempts to maintain the destination image's original value structure. This is accomplished by transferring only the α and β channels from the source. To make our method computationally tractable, we work within an image pyramid, transferring color layer by layer.

Keywords

Image recoloring, color segmentation, image pyramid.

1. INTRODUCTION

A painting's palette is responsible for how color is selected, contrasted, harmonized and blended, and it exerts a powerful influence on how the viewer interprets a painting's imagery. Consider the two Impressionist paintings at the top of Fig. 1. For *Starry Night*, Van Gogh uses stark color contrast. The cool saturated blues and greens of the sky and landscape are opposed to the warm saturated yellows and oranges of the moon and stars to create a visual intensity. Compare this with Cezanne's *Skulls*, where the painter smoothly blends warm reds and oranges of low saturation with just a few cooler brown elements to create a quiet glow.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

Figure 1. Test images and an image recoloring.
(Van Gogh and Cezanne from http://www.artchive.com)

Now, imagine being able to somehow liquefy the color of a painting in such a way that it could be poured from one painting to another thereby transferring the chromatic "orchestration," i.e. the artistic intent, of the color scheme, from one painting to the other. At the bottom of Fig. 1 we see that, paradoxically, *Starry Night*'s vibrant energy might be imbued into the somber imagery of *Skulls*.

Our goal in this paper is to investigate the problem of transferring the color scheme of a source image to a destination image and in the process learn more about what it means to "liquefy the color of a painting" and to "transfer the artistic intent of a color scheme." The solution we describe is based on the notion that one can link an image's color organization to its underlying color palette and then, by making relevant associations between colors in the source palette and the destination palette, recolor the destination image while preserving the artistic intent of the color scheme used in the source image. Potential applications include: altering the mood of a movie scene, rapidly prototyping set designs, and assisting in matching CG and live-action plates.

The idea of transferring the color content from one image to another is not new [Rei01], but previous algorithms required manual selection of swatches to steer the algorithm. Moreover, images were carefully chosen so that their compositions had about the same color proportions so that statistical methods could be used to effect the color transfer. In this paper we attempt to overcome such limitations. Our goal is to fully automate the color transfer process and thus to consider the color transfer problem in its full generality.

In developing the conceptual framework for our color transfer method we have tried to take into consideration two ideas classically trained painters are taught: 1) Paintings should be organized by value – the pattern of lights and darks. 2) To develop a painting on canvas, first block out broad loosely defined regions of color, which determine the structure of the image, and then later refine these regions by creating image detail. These ideas furnish two cornerstones for our design: 1) The value patterns of the destination image should survive color transfer from the source. 2) An image should be "deconstructed" using an image pyramid whose top layer or apex reveals only enough detail to determine a limited number of regions of color, and whose successively lower layers refine these regions.

For the purposes of illustration, throughout most of the presentation we use the low-resolution (maximum dimension of 128 pixels) *Starry Night* and *Skulls* images of Fig. 1 as source and destination images. At the end of the paper we present recolorings of other well known paintings.

2. BACKGROUND

The notion of color transfer *per se* is not widespread in the literature, although texture transfer has received considerable attention [Her01]. Reinhard et al. [Rei01] examine color transfer with the goal of color *correction* in mind. Their method is best suited for working with images (or portions of images) whose palettes are similar. Their results are quite dramatic and successful. A key contribution of their paper is the identification of the $L\alpha\beta$ color space of Ruderman et al. [Rud01] as an ideal candidate for work in image recoloring. Their recoloring method is statistical in nature: they modify destination pixels in such a way that upon transforming them to their color space they will have the same statistical *characteristics* as similarly transformed source pixels.

We work in four color spaces. Most of the work is done in $L\alpha\beta$ color space, which has the property that for a range of color images of nature scenes the L, α and β axes are highly perceptually decorrelated. The L component of the space is a reasonable measure of perceptual luminance while the α and β components measure chromatic content. We use the RGB color space for identifying extraneous colors, and we use the HSV color space to help identify color similarities within sets of colors from which palettes are extracted. Finally, out-of-gamut color correction is done with the help of the Y, or luminance, channel of the YIQ color space.

3. COLOR SEGMENTATION

Our primary objectives are to organize the color information within an image in such a way that it can be analyzed for artistic intent and that it can be quantitatively transferred from one image to another. These are mutually distinct and competing objectives. To discern artistic intent we need to make color comparisons on the basis of color metrics. To transfer color we need to capture the structure or "texture" of the color fields within an image. To handle both objectives simultaneously, we construct regions of pixels bottom-up using a region-merging algorithm that assigns binary trees to regions. Colors obtained by averaging color over all the pixels within a region are used when we need to determine artistic intent, and binary trees are used when we need to transfer color texture. Bottom-up region merging determines a color segmentation algorithm. We do not use one of the standard color segmentation algorithms [Com97, Den99] because we need to preserve color texture.

Our region-merging algorithm uses the pixel representation for a raster image first suggested by Bieri and Kohler [Bie91]. Geometrically, a *pixel* consists of a *vertex*, left and top *edges*, and a *face*. The vertex is determined by the pixel's row and column. Associated with each edge is a flag for

determining whether or not the edge is currently serving as a boundary between two regions of pixels, and a time stamp for remembering which merge event caused the edge to become converted from a boundary edge to an interior edge. Each edge knows which pixel it belongs to, and pixel color is associated with each face.

Geometrically, a *region* consists of a group of one or more pixels that are simply connected using 4-neighborhoods. A region is represented by a binary tree whose leaves are pixels, and has an identifier, an area, an average color, a height, an active flag, and a distinguished pixel (used in region merging). Initially, each pixel is made into an active region of unit area.

Any two active adjacent regions – regions sharing a common boundary edge – are candidates to be merged to form a new active region. Any sequence of region merges will color segment an image into a *forest* of binary trees. The two children of each binary tree are the regions that were merged to form that region. Pixels know which active region they belong to.

During a scanline traversal, edges of the pixel that are boundaries with adjacent pixels are placed into a priority queue. A merge event occurs when a boundary edge is removed from the priority queue and passed to a region-merging algorithm that is responsible for the bookkeeping to create a new active region from the two regions bounded by the edge. Fig. 2 shows a pixel diagram for a merge of regions N_1 and N_2 that will be triggered by boundary edge e. Our algorithm takes into account additional *shared* boundary edges such as e' and updates the priorities of non-shared boundary edges such as f, g, and h thus bringing into play regions that are respectively interior, common, and exterior to the two regions being merged. The algorithm uses merge-event time stamping to maintain edge integrity when it becomes necessary to remove more than one shared boundary edge during a merge.

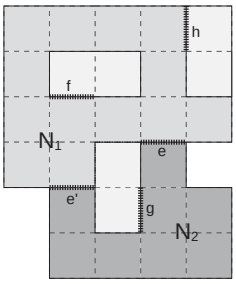

Figure 2. Merge of regions N_1 and N_2 triggered by boundary edge e.

We attempt to preserve the artist's broad color structure by merging neighboring regions with the same perceived chromatic content. The merge priority of an edge is defined to be the inverse of the sum of the squares of the color component differences across the edge, computed in $L\alpha\beta$ space.

4. THE IMAGE PYRAMID

We organize each digital image into a pyramid [Hee95] by successively down sampling so that the *base*, at layer zero, has the image at full resolution, while the *apex* has the image at the lowest resolution we wish to use. Our down sampling method is nonstandard. From a block of four pixels we select the pixel that is closest to the average color of the block in order to maintain true color and help to prevent the painterly style from being degraded by color averaging.

Fig. 3 illustrates how we use the image pyramid. The apex forms one "logical" active region. To descend through an image pyramid, we select an active region in the current layer (indicated by hatching in the figure), and we *mask* those pixels in the layer directly beneath (indicated by shading in the figure) which can be projected back up to pixels in the region selected. Now, we color segment only the portion that has just been masked, and we repeat the process on down to the base.

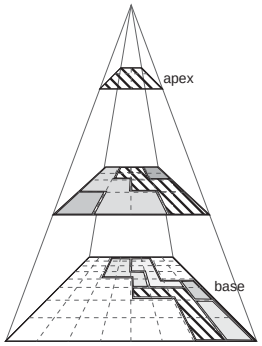

Figure 3. Synthesis within an image pyramid.

If we start in the layer beneath the apex of the image pyramid, and color segment until we have a reasonably small number of regions, then we can usually achieve good approximations to the color organization for the image. There are broad expansive areas of the image to work with which are broken up by areas containing highlights, shadows, and transitional colors.

One problem that arises is that there can be too many small regions. Some of these small regions consist of single never-merged accent *pixels*, whose boundary edges are of such low merge priority that continued merging will preserve these "rogue" regions at the expense of the structural integrity of the composition as a whole. The decision about how to handle rogue regions affects both the formation of image palettes and the actual color transfer that takes place at the level of pixels. To help prevent image

corruption, we interrupt the region-merging algorithm before the major structural elements are lost and work the rogue regions into the merged framework by *absorption* until the desired granularity is obtained. More precisely, once a merge-priority threshold is reached, we override the algorithm for selecting edges that trigger merge events so that rather than selecting edges of maximum priority, it reverts to a scanline algorithm to find an edge associated to a pixel in a region of minimal area.

We faced a tradeoff when deciding how to select companion regions to use for absorption. Our experiments showed that absorbing the smallest extant regions with their *largest* nearest neighbors gave the best color distributions, while absorbing them with their *smallest* nearest neighbors gave better image decompositions. Even though our goal is color transfer, we adopted the smallest-nearest-neighbor rule so that image recoloring would have fewer image artifacts.

In order for regions in layers that are lower in the pyramid to be able to reveal more detail when they are color segmented we *decrease* proportionally the threshold we use to interrupt priority merging. Because absorption impacts color transfer at the pixel level, we do not absorb rogue regions in subsequent layers.

5. PALETTE CONSTRUCTION

Our first step in palette creation is to compensate for color duplication within the set of *segmented* colors, i.e. the set of averaged colors from active regions. Because regions are 4-connected, there will usually be a number of non-adjacent regions that will have perceptually indistinguishable colors. We partition the set of segmented colors into subsets of colors and select one color representative from each subset. These representatives form the set of *identified* colors. Working in RGB space, we define two colors to be perceptually identical provided their Euclidean distance does not exceed a specified threshold. We use RGB space for this purpose because $L\alpha\beta$ space is logarithmic and makes too fine a distinction between dark colors and too coarse a distinction between light colors. The color chosen as the canonical representative for a subset is the one that is associated to the region having the largest area. Following segmentation, *Starry Night* has 159 colors and *Skulls* has 119 colors. Following identification, these are reduced to 22 and 13.

Since we are attempting to model the painter's palette, we want a bare minimum number of colors, thus we must account for the fact that within the set of identified colors there may be a number of shades of a given color. We need to partition the set of identified colors into subsets such that all colors within a subset are similar up to shading. Unfortunately, this is a very difficult task, as no single color space accurately captures the notion of shades of a color. We again adopted a bottom-up approach. Sorting the identified colors by their saturation components in HSV space, we let each (unused) color serve as the representative for a cluster of colors that are similar with respect to shading as determined in some color space. Iterating this clustering algorithm first in HSV space, where similarity is defined using a "tapered wedge" neighborhood, then in $L\alpha\beta$ space, where similarity is defined using a "slab" neighborhood, allows us to "converge" to a set of color representatives for clusters of colors within the identified set of colors. Clustering in HSV space has advantages for lighter colors, while $L\alpha\beta$ has advantages for darker colors. The final color *palette* consists of the set containing the most saturated color in each cluster. For our test images, shade clustering reduces the number of colors for *Starry Night* from 22 to a palette of 10 and in *Skulls* from 13 to 8.

For most images it does not make sense to try and recolor true black or true white, since they are mixing colors used to make shades from palette colors. Thus, we do not consider true black and true white as belonging to our palette, when we are trying to make decisions about color transfer. This has no effect on the *Skulls* palette, but does remove the darkest color from consideration in the *Starry Ni*ght palette.

At the top level of the pyramid, following "shade extraction" our algorithm establishes an initial set of "master" palette color associations. Thanks to master associations, when layers lower in the pyramid are considered, although we must continue to compensate for color duplication following segmentation, it is no longer necessary to extract shades, because at this point we are trying to *refine* the master associations. Instead, we *discard* some colors from the set of identified colors of the source image in an effort to prevent spurious colors that arise during segmenting from corrupting color transfer from source to destination. Such spurious colors can result from averaging near feature boundaries, averaging over non-segmentable textures, or averaging over upsampling artifacts. After the issue of color duplication has been dealt with, we discard colors by sorting them on the basis of segmented area and then retaining only enough colors to account for 90% of th e portion that has just been segmented.

In the layer immediately below the layer that was segmented to extract the image's palette, whenever a projected region is to be segmented, the region is redefined to include the projected regions from all the regions in the class of its parent. This ensures that all the shades from the source color and all the shades

from the destination color get considered simultaneously.

To summarize, an image palette color is a region color that is responsible for a set of shades, and a shade is a region color that is responsible for a subset of regions with perceptually indistinguishable colors. Thus, image palette colors partition the set of segmented colors into classes. The particular region in each class whose color is identical with the palette color serves as the canonical representative for the class to which it belongs.

Fig. 4 shows the palettes and the segmented images recolored with respect to the palette color of the class they belong to. These palettes were obtained by using a four layer pyramid with the apex having maximum pixel dimensions *16 x 16*. Colors from the palette are sorted in descending order according to the total area they are responsible for. A histogram of these areas lies above each palette.

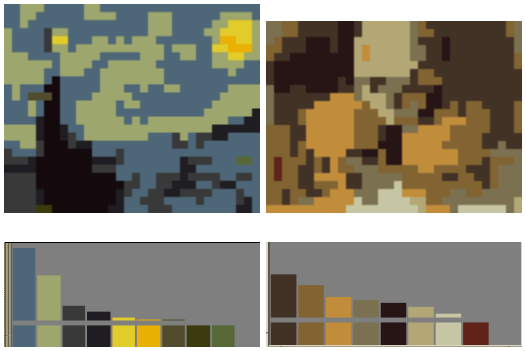

Figure 4. Palettes for the low resolution test images.

6. PALETTE COLOR ASSOCIATIONS

Although making "intelligent" color associations between source palette colors and destination palette colors is not the primary focus of the current paper, we propose a preliminary approach. We call this approach naïve, as it is not informed by deep knowledge of palette structure or artistic intent. Numerous tests suggested to us that it can be beneficial to associate the color responsible for the largest area in the source with that from the destination. This leads to the color transfer heuristic: form an anchor pairing between source and destination based on largest areas and then make further pairings based on how colors deviate from these anchor colors. Our implementation sorts the palette colors from largest to smallest by area. We then translate all of the colors in a palette so that the color with the largest summed area is the origin of a local palette color space. This serves to measure the remaining colors *relative* to the color with the largest summed area. Since translation preserves relative values of *L*, value rankings of the colors do not change. In the spirit of Reinhardt et al. [Rei01], we turn to elementary statistics to establish a meaningful way to make comparisons across color sets. Within each palette, the mean and standard deviation of each component is calculated, and all components are replaced by their z-scores, i.e. the number of standard deviations from the mean. We then translate a second time so that the anchor color remains at the origin.

Our algorithm for associating source palette colors to destination palette colors forces the colors representing the largest summed areas of the two images to be paired. To *encourage* the colors representing the second largest areas to be paired, we rotate each normalized palette around its *L*-axis so that the β components of the normalized colors corresponding to the second largest summed areas are zero. Palette color associations and their induced source to destination region pairings now arise as follows. Associate to each normalized, rotated color in the destination palette the color in the source palette that is closest to under Euclidean distance measure. For each of these palette color pairings, form source to destination region pairs by linking the source palette color's canonical region with all of the regions in the class of the destination palette's color. Fig. 5 shows the palette color pairing this strategy produced for our test images following color segmentation of the *32* x *32* resolution layer. Source colors are in the top narrow row, and corrected source colors (see next section) are in the lower narrow row. Destination color is in the bottom thicker row. Since the palette color associations in Fig. 5 respect the area sort that was imposed upon the destination palette, we are able to observe that some source colors (e.g. gray) are used only sparingly and some (e.g. gold) not at all.

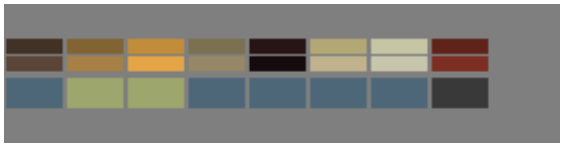

Figure 5. Palette color associations for the test images.

7. PIXEL LEVEL COLOR TRANSFER

The final step of our color transfer method requires us to use the binary trees from a paired source region and destination region to transfer color from source pixels to destination pixels. If the trees were isomorphic, then we could traverse them in parallel and achieve a one-to-one pixel matching. However, the two trees will almost certainly be topologically different. Instead, we transfer color between paired regions by simultaneously traversing their binary trees in such a way as to approximate a parallel traverse between two identical full trees.

There is insufficient space for the full details, but our traversal algorithm has the following properties: 1) if presented with identical trees it pairs identical pixels, 2) if a source pixel is transferred to a destination region, then this pixel's color is distributed throughout the entire region, 3) if a source region is transferred to a destination pixel, we choose the color closest to the region's average color, and 4) when neither the source region nor destination region is a leaf, we recursively descend through the subtrees until a single source or destination pixel is reached.

The images in the top row of Fig. 6 show the advantage we gain from subtree analysis. The image at the upper left shows the transfer of the average $L\alpha\beta$ color from the paired source *region*, while the image at the upper right shows the transfer of the $L\alpha\beta$ color from the paired source *pixel* for our test example. The pixel transfer approach clearly preserves more of the visually rich color texture of the source region.

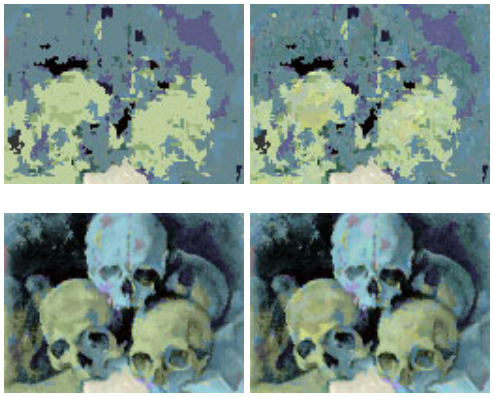

Figure 6. Comparison of color transfer by regions (left column) versus pixels (right column) and by $L\alpha\beta$ (top row) versus $\alpha\beta$ (bottom row).

8. COLOR CORRECTION

Since we want to transfer chroma not value, we transfer only the α and β channels from source to destination. Fig. 6 shows how important this is. Without value preservation (top row), the structure of the image is lost, whereas with preservation (bottom row) it is retained. However, two problems arise: 1) color corruption due to large discrepancies in the L channels i.e. excessive lightening or darkening of a color and 2) out-of-gamut colors due to significant differences in hues. When the L component of a color is low, its chroma should not matter. Since many digital images have high saturations in their dark colors, transferring the chroma from dark colors to destinations with moderately higher value components often yields unnaturally saturated colors. As a partial solution to this problem, when we transfer the α and β channels from the source image to the destination image, we attenuate them when the destination pixel is bright and has a much higher value than the source pixel.

Even though desaturation is not invoked when highly saturated colors are transferred to destination colors of lower value, out-of-gamut colors may still result because the "strength" of the chroma being transferred may artificially inflate one or more of the RGB components when we transform from $L\alpha\beta$ space. Regardless of how out-of-gamut colors arise, after we transform all of the destination image pixels back to RGB space we must often make a global correction of the RGB image so that all colors are in gamut. This is accomplished by identifying the 96-th percentile for the set of R, G, and B components collected from all image pixels and then scaling the entire image by this value, if necessary. We found that scaling by the global maximum darkens the image too much, so use of this smaller value is a compromise that seems to work well in practice. Pixels where one or more R, G, and B components were above the 96-th percentile are still out-of-gamut, so we do a local correction as follows: the maximal channel value is clamped and the values of the remaining channels are raised in an effort to try to preserve the pixel's luminance, by holding the Y component (of YIQ space) constant.

Desaturating colors as they are transferred from source pixel to destination pixel and then invoking global color corrections can produce images with excessive amounts of gray. This problem is most acute when (pure) whites are involved. Trying to add chroma to whites in $L\alpha\beta$ space causes colors to soar out of gamut in RGB space. Further complicating matters is the fact that when trying to transfer saturated colors to dark areas, global correction can create "hot spots" caused by clamping only a few pixels within a region. Issues such as these revealed to us how difficult the *general* problem of color transfer really is.

In making images for this paper we added one more feature to try and head off out-of-gamut problems. When artists want to incorporate a disparate color into their palette at a late stage of image composition, they must often readjust the value structure of the entire painting. Therefore, prior to color transfer we tried preconditioning our destination images by implementing an algorithm to modify L values of the pixels in the destination image so that their resulting histogram would match the histogram of L values in the source image. Fig. 5 showed the colors in the destination palette on top split in half. The top half is the color as it appears in the image and the bottom half is the color as it appears after conditioning. The fact that a destination image might need conditioning in order to be "value compatible"

Figure 7. Top: Test Fragonard and Kandinsky. Lower left: Fragonard value-conditioned from Kandinsky, right: Kandinsky from Fragonard.

(Fragonard **&** Kandinsky from http://www.artframed.com)

with a source image demonstrates why it is virtually impossible for certain recolorings to be done. For example, Fig. 7 shows what happened when we tried to condition a Fragonard using a Kandinsky and vice versa. The Fragonard became useless after this step, as its value structure was destroyed by the process. The Kandinsky held up somewhat better.

9. EXAMPLE RECOLORINGS

When producing high resolution image recolorings, to prevent "blockiness" from propagating during pyramid descent, we turn off absorption after the initial layer is color segmented, and as each new layer within the destination pyramid is encountered, we examine its pixels one by one in order to resolve any questionable parent-region assignments for pixels that are children of *boundary* pixels in the parent layer. A high resolution recoloring for our test images is shown in Fig. 8. A high resolution recoloring of Kandinsky by Fragonard is shown in Fig. 9.

The top row of Fig. 10 shows additional test images by Franz Marc (*Yellow Cow*) and Emil Nolde (*Mask Still Life III*) that we used to evaluate the generality and the limitations of our approach. They were selected for their range of palette colors. Figs. 11 and 12 show bi-directional recolorings. The bottom row of Fig. 10 shows the top-level palette color associations for each of these recolorings.

Figure 8. Recoloring of *Skulls* by *Starry Night.*

Figure 9. Recoloring of Kandisky by Fragonard.

10. CONCLUSIONS

The goals of this paper were limited – we attempted to establish a framework to investigate the problem of transferring the chromatic content from one image to another in such a way that its artistic intent was preserved. The images presented are demonstrations of the potential of this framework, and provide a benchmark by which to measure future work. In that sense, we feel that we have succeeded in our original quest.

Nevertheless, we were never wholly successful at making a bi-directional recoloring of *Starry Night* with *Skulls*, primarily because the narrow range of earth tones in *Skulls* meant that nearly all the colors in

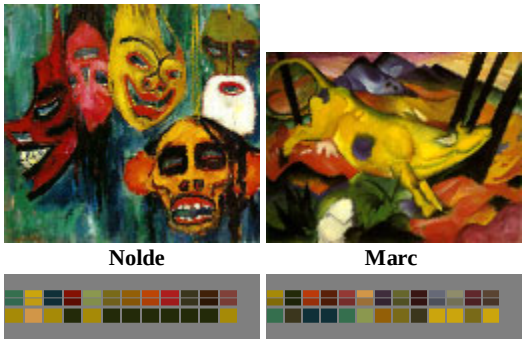

Figure 10. Top row: Nolde and Marc images. Bottom row: Palette color associations. (Nolde and Marc from **http://www.artframed.com**)

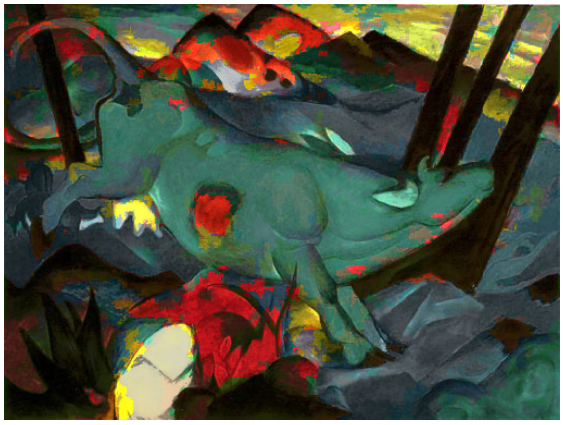

Figure 11. Recoloring of Marc by Nolde.

Figure 12. Recoloring of Nolde by Marc.

its palette needed to be transferred before the *Starry Night* recoloring made sense. Given the chromatic range of *Starry Night* this is not an easy task. In general, we found that the more colors that *needed* to be transferred from the source to the destination, the more likely it was that errors in artistic intent would occur.

Thus, the most interesting direction for future work will be to investigate how to make decisions about color pairing across palettes. It seems clear that such an approach must be informed by an understanding of the artistic intent of each palette, and must attempt to preserve that intent across the color transfer.

To summarize, we feel that the key problems that remain are: 1) deciding which colors within a palette are the most important ones to use for associations, 2) establishing "rule sets" for making color associations between palettes, 3) refining color associations while descending through an image pyramid (this causes new colors to come into play, which are not easily reconciled in accordance with top-level color associations), and 4) making color transfer decisions that keep colors in gamut.

11. REFERENCES

[Bie91] H. Bieri and A. Kohler. Computing the area, the circumference, and the genus of a binary digital image. Graphics *Gems II*, J. Arvo (editor), Academic Press, 107--111, 1991.

[Com97] Comaniciu, D. and P. Meer, Robust analysis of feature spaces: color image segmentation, *Proceedings of CVPR '97*, 1997, 750-755.

[Den99] Y. Deng, B. Manjunath and H. Shin, Color image segmentation, Proceedings of CVPR '99, 1999.

[Hee95] D. Heeger and J. Bergen. Pyramid-based texture analysis/synthesis. ACM Computer Graphics (Proc. of SIGGRAPH '95), 229-238, 1995.

[Her01] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. ACM Computer Graphics (Proc. of SIGGRAPH '01), 327--340. 2001.

[Rei01] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. IEEE Computer Graphics and Applications, 21(5):34--41, 2001.

[Rud01] D.L. Ruderman, T.W. Cronin, and C.C. Chiao. Statistics of cone responses to natural images: implications for visual coding. J. Optical Soc. of America, 15(8):2036--2045, 2001.