Automatic Mood-Transferring between Color Images

Chuan-Kai Yang and Li-Kai Peng ■ *National Taiwan University of Science and Technology*

With the digital camera's invention, capturing images has become extremely easy and widespread, while image data has become more robust and easy to manipulate. Among the many possible image-processing options, users have become increasingly interested in changing an image's tone or mood by altering its colors—such as converting a tree's leaves from green to yellow to suggest a change of season.

Groundbreaking work by Reinhard and colleagues made such a conversion possible and extremely simple.[1] In their approach, users provide an input image, along with a reference image to exemplify the desired color conversion. The technique's algorithm essentially attempts to match the input image's color statistics with those of the reference image to achieve the tonal change. However, this seemingly successful approach has two outstanding problems. First, some users might find it difficult or impossible to choose a proper reference image because they lack either an aesthetic background or access to an image database. Second, it's difficult to evaluate the color-transfer quality, particularly when the input and reference images have significantly different content.

This new color-conversion method offers users an intuitive, one-click interface for style conversion and a refined histogram-matching method that preserves spatial coherence.

We propose solutions to both problems. First, we give users a simple and intuitive interface that includes a list of predefined colors representing a broad range of moods. So, rather than having to supply a reference image, users simply select a color mood with a mouse click. To address the color-transfer quality problem, we associate each color mood with a set of up to 10 images from our image database. After selecting their color mood, users choose one associated image. Our histogram matching algorithm then uses the selected image to determine the input image's color distribution. We thereby achieve more accurate and justifiable color conversion results, while also preserving spatial coherence. Here, we further describe our solutions and their results and compare them to existing approaches.

Color-mood conversion

Our approach adopts Whelan's classification scheme,[2] dividing the color spectrum into 24 categories or moods. To create a mood image database, we collect from the Web five to 10 images for each category; these serve as our reference pool. At runtime, when users input an image and a target mood, our system classifies the input image. Then, if necessary, it makes corresponding *global alignment transformations*—including translation, rotation, and scaling—to every pixel of the input image. However, because such transformations alone can't guarantee that the system will successfully convert the input image to the desired category, we employ a segmentation-based approach to make the color adjustment region by region. We perform such adjustments repeatedly until the input image matches the target mood.

Because each mood category contains more than one image, we can further enhance our conversion quality by treating each associated database image as a reference in the transformation process. Our system can then choose the best conversion result as the output image. Finally, we use the RGB color system. Although Reinhard and colleagues propose using the Lαβ for better decorrelation, we've found little significant differences among color systems. Also, Whelan uses the CMYK system in their original color-mood definitions,[2] so we've converted those definitions to their RGB counterparts for adoption in our system.

 Published by the IEEE Computer Society 0272-1716/08/$25.00 © 2008 IEEE

Color-mood classification

Figure 1 shows the 24 mood categories and their primary corresponding colors (as all moods also have secondary colors). As the figure shows, the classification scheme encompasses a broad range of moods and is thus suitable for color transferring. For example, the romantic mood, primarily represented by pink, might arouse romantic feelings, while the calm mood, denoted by a dusty blue, conveys emotions of peace and leisure (see Figure 2). More information on colors and the emotions they arouse is available elsewhere.[2]

Powerful	Soft	Traditional	Regal
Rich	Welcoming	Refreshing	Magical
Romantic	Moving	Tropical	Nostalgic
Vital	Elegant	Classic	Energetic
Earthy	Trendy	Dependable	Subdued
Friendly	Fresh	Calm	Professional

Figure 1. The 24 color moods and their primary corresponding colors.

Image classification

To build a given input image's associated color histogram, we first examine all of its pixels. Our color histogram has 24 color moods on its *x*-axis; each color's accumulated frequency is piled along the *y*-axis. To construct the histogram, we compare each pixel of the input image with all 24 moods to find the closest mood, then attribute it to that color mood's bin in the color histogram. Each color mood has a corresponding RGB value, represented as a point in the RGB color space. We therefore define the difference between a pixel and specific color mood as their Euclidean distance in the RGB color space. That is,

$$\sqrt{(R_1-R_2)^2+(G_1-G_2)^2+(B_1-B_2)^2} \quad (1)$$

where (R_1, G_1, B_1) and (R_2, G_2, B_2) are two points in the RGB color space. Figure 3 shows some example histograms. To classify an image after its histogram is computed, we could determine its category

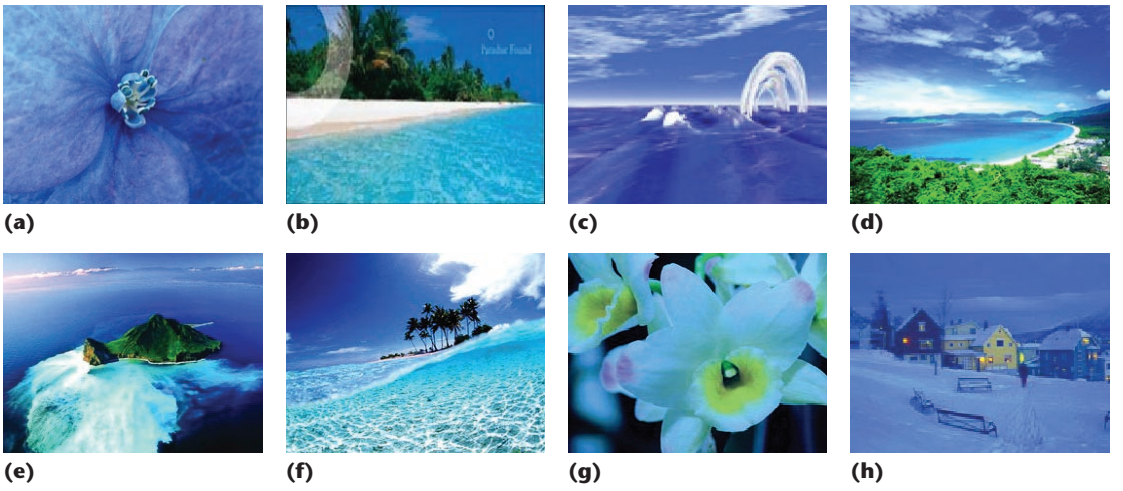

Figure 2. Database references images for the "calm" mood category.

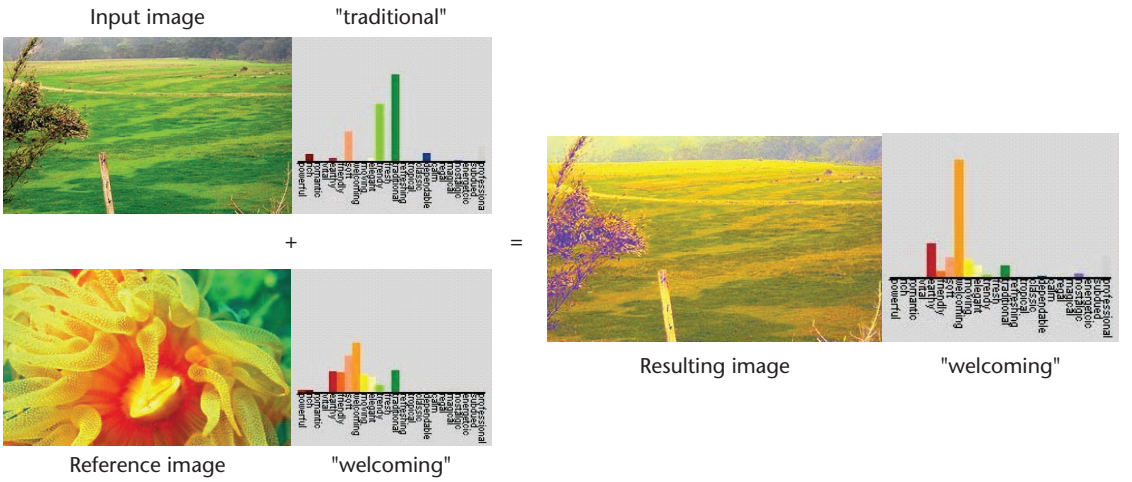

Figure 3. Color-mood transfer. Using the principle-component-analysis technique, the system finds the RGB space's main trend or distribution axis for both images, and then makes corresponding transformations to fit the input image's distribution into that of the reference image.

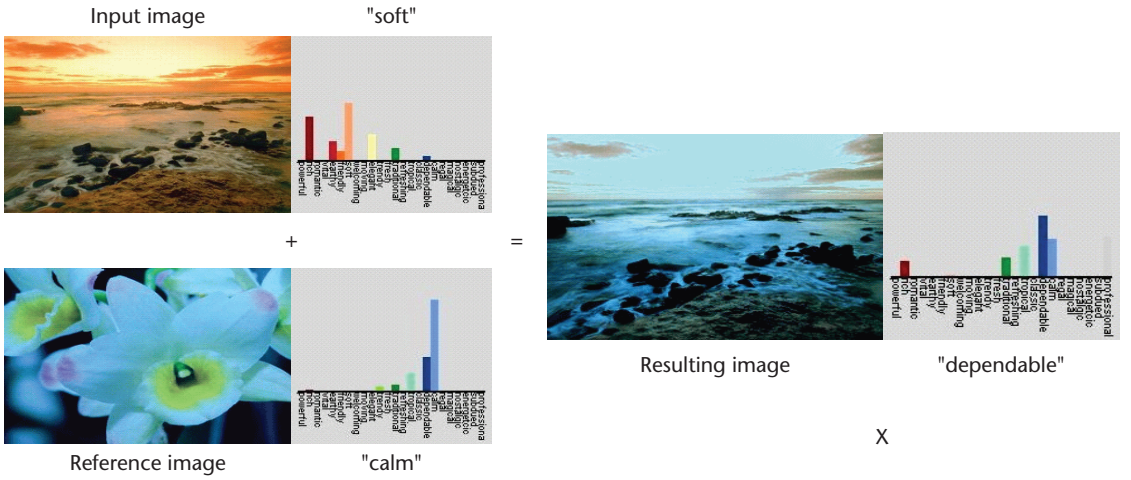

Figure 4. Incorrect color-mood transfer. Such results can occur when the process transforms the input image's pixels.

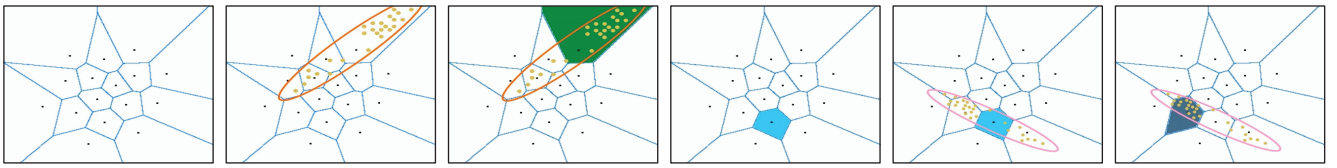

Figure 5. Challenges in converting an input image into the correct color mood. (a) All mood category colors are represented as black dots, forming a Voronoi diagram. (b) When most pixels reside in a single region, we classify the image according to the region's corresponding mood. (c) The green region has the most pixels, so we use it to classify the image. However, (d) a reference image might assume a particular mood, and then affect the (e) transformation that brings the input image into a new region after color-mood transfer. (f) The resulting undesirable classification differs from the reference image's mood.

by locating the mood with the highest frequency. However, if two (or more) color moods enjoy similar frequency, the image's dominant tone would be ambivalent. To eliminate this potential ambiguity, we require that all mood database images meet the following histogram constraint: their second highest frequency must be less than 75 percent of the highest frequency, thus clearly making the given image's highest-frequency color mood its primary color. We also endeavor to gather a wide variety of images within each mood category to ensure the best result in the final stage.

Color-mood transfer

We use the RGB color system for both image classification and color conversion. To make the input image align more closely with the reference image, we adopt Xiao and colleagues' approach for matching the input image's color distribution with that of the reference image.[3] Essentially, we use the principle-component-analysis technique to find the RGB space's main trend or distribution axis for both images, and then match the input image's distribution with that of the reference image.

First, we find the covariance matrix of the involved pixels' values in the RGB space; then we determine the matrix's eigen values and eigen vectors. The eigen vector corresponding to the largest eigen value represents the desired principle axis, which we can use to derive the rotation transformation that aligns two color distributions. Because the transformations might put the adjusted RGB values out of range—either overflowed or underflowed—we clip these values to within the normal range. Figure 3 shows the results of applying our color-alignment process.

Image segmentation

Given these transformations, we might expect the transformed input image to match the reference image. However, as Figure 4 shows, performing these transformations alone is sometimes insufficient.

Figure 5 offers an example of why: In Figure 5a, we plot all mood categories' representative colors as black dots to form a Voronoi diagram. Each dot corresponds to a (closed or half-open) region, plotted in blue. For any point within this region, the distance to the black dot must be smaller than that to all other dots on the plane. Therefore, when most of an input image's pixels—represented as yellow dots in Figure 5b's orange ellipse—fall into one region (the green region in Figure 5c), we classify the image as within the corresponding mood. However, as Figure 5d shows, a reference image might assume a particular mood (blue region), but the color-mood transfer (affected by the reference image's principle axis) might trans-

form the input image's pixels into new positions, as the yellow dots in the pink ellipse in Figure 5e show. Such a points distribution can lead to an undesired classification—the dark blue in Figure 5f—that differs from reference image's mood.

As the "Related Work in Color Conversion" sidebar describes (next page), researchers traditionally address this issue using histogram matching. However, most such approaches fail to preserve spatial coherence and, as a result, produce undesired artifacts. Our new histogram matching algorithm more accurately matches the input and reference images, while also preserving the spatial coherence. We do this using image segmentation, which lets us further adjust the transformed input image to better resemble the reference image (see Figure 6). However, image segmentation is still not a well-defined problem and thus typically requires predetermined thresholds; in the worse case, such thresholds depend on the input image. Fortunately, our color-mood classification scheme offers an inherent image segmentation scheme. That is, we can form each segment from neighboring pixels belonging to the same color mood. This native segmentation scheme sets the stage for the ensuing critical step: color-mood adjustment. The sidebar, "Histogram Matching with Spatial Coherence" (page 59) describes our approach in more detail.

Color-mood adjustment

Our goal with color-mood adjustment is to ensure that, following color conversion, we can turn any input image into the user-specified target mood. As we noted earlier, for all images in the mood image database, their second most-frequent color mood must contain less than 75 percent of the number of pixels belonging to the most-frequent color mood. This prevents ambiguity as to the image's color mood. In keeping with this, when we adjust a transformed image's color mood, we must meet our *dominance constraint*. That is,

- the target image's desired color mood must have the highest frequency, and
- the second dominant color mood must have less than 75 percent of the highest frequency's pixels.

Color-mood adjustment consists of five steps:

1. We perform color segmentation using the 24 color moods and initially label each segment "untouched."
2. For each segment, we calculate its centroid in the RGB space and measure the distance between this centroid and the target color mood's representative color (represented as a point, C_t, in the RGB space).

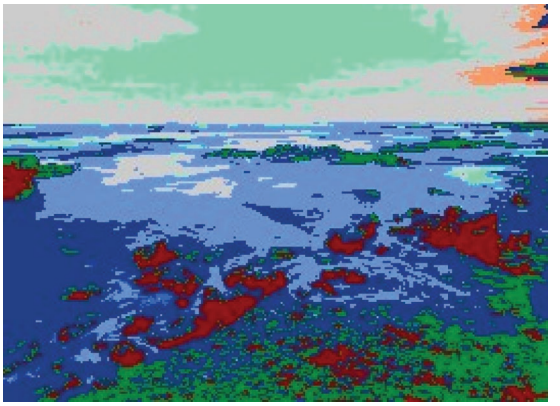

Figure 6. Color segmentation of Figure 4's input image. Each segment is colored with the representative color of the corresponding "dependable" color mood.

3. We identify the segment with the least distance to C_t and denote it by $S_{closest}$.
4. For each pixel near $S_{closest}$, we reassign it a new color: the average of its original color and C_t. After adjusting each pixel, we label the segment "adjusted." We use such a moderate approach, rather than converting the whole segment directly into the target mood, to avoid the visual artifacts that can result from a sudden color migration. However, after adjusting by averaging, segment boundaries might be affected—that is, some segment areas might increase or decrease. For efficiency's sake, we choose not to update segment information immediately, but rather to defer updates until all segments have been adjusted.
5. Finally, we repeat steps 3 and 4 until we satisfy the dominance constraint. It's possible, however, that we could adjust all segments without meeting the constraint. In that case, we revert to step 1 and start the process over. Convergence is guaranteed, because step 4 operations always change some pixels, increasing the number of pixels that belong to the target color mood.

Output-image selection

When a user selects a specific mood, we use each of its images as a reference image and output the one with the most similar content (or, to be precise, the most similar brightness histogram) to the input image, as well as the most similar color tone (mood histogram) to the reference image.

To measure the difference in brightness (or *luminance*) between the input and output images, we use the following equation:

$$L(input, output) = \frac{\sum_{i=1}^{w_{output}} \sum_{j=1}^{h_{output}} | l_{input}(i,j) - l_{output}(i,j) |}{w_{output} \cdot h_{output}} \quad (2)$$

where $l_{\text{input}}(i, j)$ and $l_{\text{input}}(i, j)$ are the luminance values of the input and output images' pixels (i, j), respectively (normalized to the 0-1 range for convenience), and w_{output} and h_{output} represent the

Related Work in Color Conversion

Researchers have long known that color could arouse different feelings. For instance, warm colors, such as red and orange, could encourage and energize people, while green and blue could offer refreshing and peaceful feelings. Given this, it's possible to classify images on the basis of their colors, as we do in our work.

Semantic descriptions

By treating color as a feature in an image, Wang and colleagues use a color-semantic-description approach for image query.[1] They first segment images and classify them using a fuzzy clustering algorithm applied on the L*C*h color system. They then associate semantic terms with various color characteristics—including lightness, saturation, hue, and contrast. For example, for saturation, possible semantics might include colorless, weak, moderate, strong, and very strong. Such a scheme lets them define a particular emotion by combining specific semantics and characteristics; user queries can then return images satisfying the desired properties.

Color transfer and conversion

Reinhard and colleagues' pioneering work on transferring a color image's tone or mood to another image is the foundation for our work and that of others, including Welsh and colleagues' grayscale image colorization.[2] Reinhard and colleagues' algorithm consists of the following steps: First, it converts an input image from its original RGB color space into an Lαβ color space to eliminate RGB color representation's normally high correlation. (Here, high correlation refers to the fact that, when a pixel has a large value associated with its red channel, the blue or green channel values are often large as well.) Second, the source and reference images might have different lightness distributions; they should therefore be "aligned" before conversion occurs. The alignment process starts by finding the means and standard deviations of the Lαβ color values for both the input and reference images. The output image is obtained by translating and scaling the input image so its mean and standard deviation match that of the reference image.

Xiao and colleagues proposed a similar approach,[3] but their method differs from that of Reinhard and colleagues in at least two respects. First, instead of Lαβ, they apply the popular RGB color system and demonstrate that this decision doesn't affect their approach's effectiveness. Second, instead of aligning the input image with the reference image only in terms of mean and standard deviation (as Reinhard and colleagues did), they also align both images' principle axes. Such an alignment might sometimes be necessary, particularly when two different color distributions have the same mean and standard deviation. In such cases, they can achieve alignment by performing principle component analysis—that is, in addition to translations and scalings, they can use rotations to seek a more matched color distribution.

Greenfield and colleagues[4] focus their color conversion work on oil painting images. Adopting the Lαβ color system, they first perform color segmentation, and then merge regions with similar colors. They then reduce the segments' averaged colors into a more restricted set of representative colors to eventually derive the color palettes for both the input and reference images. They use a naive color-palette correspondence rule to transfer only the α and β values from the reference image's palette color to that of the target image. To speed up computations, they use an image pyramid structure.

Chang and colleagues[5] perform color conversion using 11 basic colors and assume that the reference images are paintings. They convert both the input and reference images into the Lαβ color space, which is partitioned into 11 regions. They build 11 convex hulls for both images and the correspondence of pixel colors—represented as points in their individual color spaces—according to their relative positions to the centroids and corresponding convex hull boundaries. In situations where some of the 11 colors are missing, users can select multiple reference images.

Use of reference images

Finding a proper reference image is a key issue for Reinhard and colleagues, as well as for those who adopt their approach. This includes Greenfield and colleagues[4] and Chang and colleagues[5] for colored image-style conversion, as well as Welsh and colleagues[2] for colorizing grayscale images. Viera and colleagues[6] address the issue by proposing several metrics for identifying the most suitable reference image via comparisons of the luminance channels between input and candidate reference images. Their proposed metrics include histogram, average brightness, gradients, hybrid of histogram and gradients, hybrid of average brightness and gradients, and direct comparison on pixel-wise differences. Their system can then choose, as the reference image, the image that most closely matches the input image. Viera and colleagues also built an image database after experimenting with the proposed metrics. However, their metrics are not directly applicable for selecting a reference image for a colored input image.

output image's width and height, respectively. The input and output images have the same dimension, as we convert the output image from the input image. However, the input and reference images needn't share the same dimensions. To accommodate a histogram discrepancy between the output and reference images, we first normalize the output image's histogram as follows: Let $H_{ref}(i)$

Histogram matching

Researchers have conducted numerous studies on histogram matching; here, we'll review four of them. Pichon and colleagues[7] use a mesh deformation approach to achieve histogram equalization—that is, a transform that sets an equal frequency for all colors. As long as the equalization transform for any histogram is invertible (that is, it has an inverse transform), we can match two arbitrary histograms by treating the equalized histogram as the intermediate histogram. Using a 2D color space as an example, they apply a mesh deformation as follows: First, they start from a uniform mesh, where each cell's area is a constant, and the number of cells represents the number of histogram bins. They then iteratively deform the uniform mesh, so that each cell contains the same number of points (or pixels in the image space). The iteration adjusts the mesh vertice positions by moving them toward or away from their corresponding cells' centroids to shrink (or enlarge) the area so that it contains less (or more) points. They use the final deformed mesh to define a piecewise linear deformation function of the color space. By mapping the original and deformed meshes, they establish the color mapping and thus achieve histogram equalization.

Morovic and colleagues[8] proposed a histogram matching algorithm for grayscale images using a fast, noniterative Exact Histogram Matching algorithm, rather than the traditional iterative Earth Mover's Distance (EMD) algorithm. Essentially, they lay their source and target histograms end-to-end in ascending order and build a look-up table accordingly to transform the source pixel value into the target one. They later generalized their approach to deal with 3D color histogram matching with an EMD algorithm.[9] They also discussed several implementation issues, including the number of bins used in a histogram, the maximal number of iterations, and undesired artifacts due to spatial frequency change.

Neumann and colleagues[10] perform histogram matching in the hue, saturation, and value color space, and propose using three sequential 1D histogram matchings to achieve a 3D histogram matching. They also apply histogram smoothing and suppression to mitigate the artifacts incurred by the histogram matching process.

Our approach and existing work

Almost all of the histogram matching approaches focus on building a perfect mapping from the input image's color space to that of the reference image, without accounting for a transformed pixel's spatial properties—including its color and its relationship with neighboring pixels. As a result, a color mapping deemed perfect in the color space can cause differences between two spatially adjacent pixels, thus leading to more visible artifacts. This fact prodded us to contrive a new algorithm to achieve histogram matching while also attempting to preserve spatial coherence (see the "Histogram Matching with Spatial Coherence" sidebar on page 59).

Overall, previous studies have inspired us in several ways. We adopted Xiao and colleagues' approach[3] as part of our color conversion process, for example. We also perform color segmentation, as proposed by Wang and colleagues and Greenfield and colleagues.[4] Finally, we built a mood image database, similar to that of Viera and colleagues.[6]

References

1. W.N. Wang and Y.L. Yu, "Image Emotional Semantic Query-Based on Color Semantic Description," *Proc. 4th Int'l Conf. Machine Learning And Cybernatics*, IEEE CS Press, 2005, pp. 4571-4576.
2. T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring Color to Grayscale Images," *Proc. Siggraph*, ACM Press, 2002, pp. 277-280.
3. X. Xiao and L. Ma, "Color Transfer in Correlated Color Space," *Proc. ACM Int'l Conf. Virtual Reality Continuum and Its Applications*, ACM Press, 2006, pp. 305-309.
4. G.R. Greenfield and D.H. House, "Image Recoloring Induced by Palette Color Associations," *Proc. 11th Int'l Conf. Central Europe Computer Graphics, Visualization, and Computer Vision* (WSCG), Union Agency–Science Press, 2003, pp. 189-196.
5. Y. Chang, S. Saito, and M. Nakajima, "A Framework for Transfer Colors Based on the Basic Color Categories," *Proc. Computer Graphics Int'l*, IEEE CS Press, 2003, pp. 176-183.
6. L. Vieira et al., "Automatically Choosing Source Color Images for Coloring Grayscale Images," *Proc. XVI Brazilian Symp. Graphics And Image Processing*, IEEE CS Press, 2003, pp. 151-158.
7. E. Pichon, M. Hiethammer, and G. Sapiro, "Color Histogram Equalization through Mesh Deformation," *Proc. Image Processing*, IEEE CS Press, 2003, pp. 117-120.
8. J. Morovic, J. Shaw, and P. Sun, "A Fast, Non-Iterative and Exact Histogram Matching Algorithm," *Pattern Recognition Letters*, vol. 23, nos. 1–3, 2002, pp. 127-135.
9. Morovic and P. Sun, "Accurate 3D Image Color Histogram Transformation," *Pattern Recognition Letters*, vol. 24, no. 11, 2003, pp. 1725-1735.
10. L. Neumann and A. Neumann, "Color Style Transfer Techniques Using Hue, Lightness and Saturation Histogram Matching," *Proc. Computational Aesthetics in Graphics, Visualization and Imaging*, Eurographics Assoc., 2005, pp. 111-122.

be the number of pixels that fall into the reference image's i-th color-mood category. Therefore, after normalization, the number of pixels that "should" fall into the i-th color-mood category is

$$H_{should\_output}(i) = \frac{H_{ref}(i)}{w_{ref} \cdot h_{ref}} (w_{output} \cdot h_{output}) \quad (3)$$

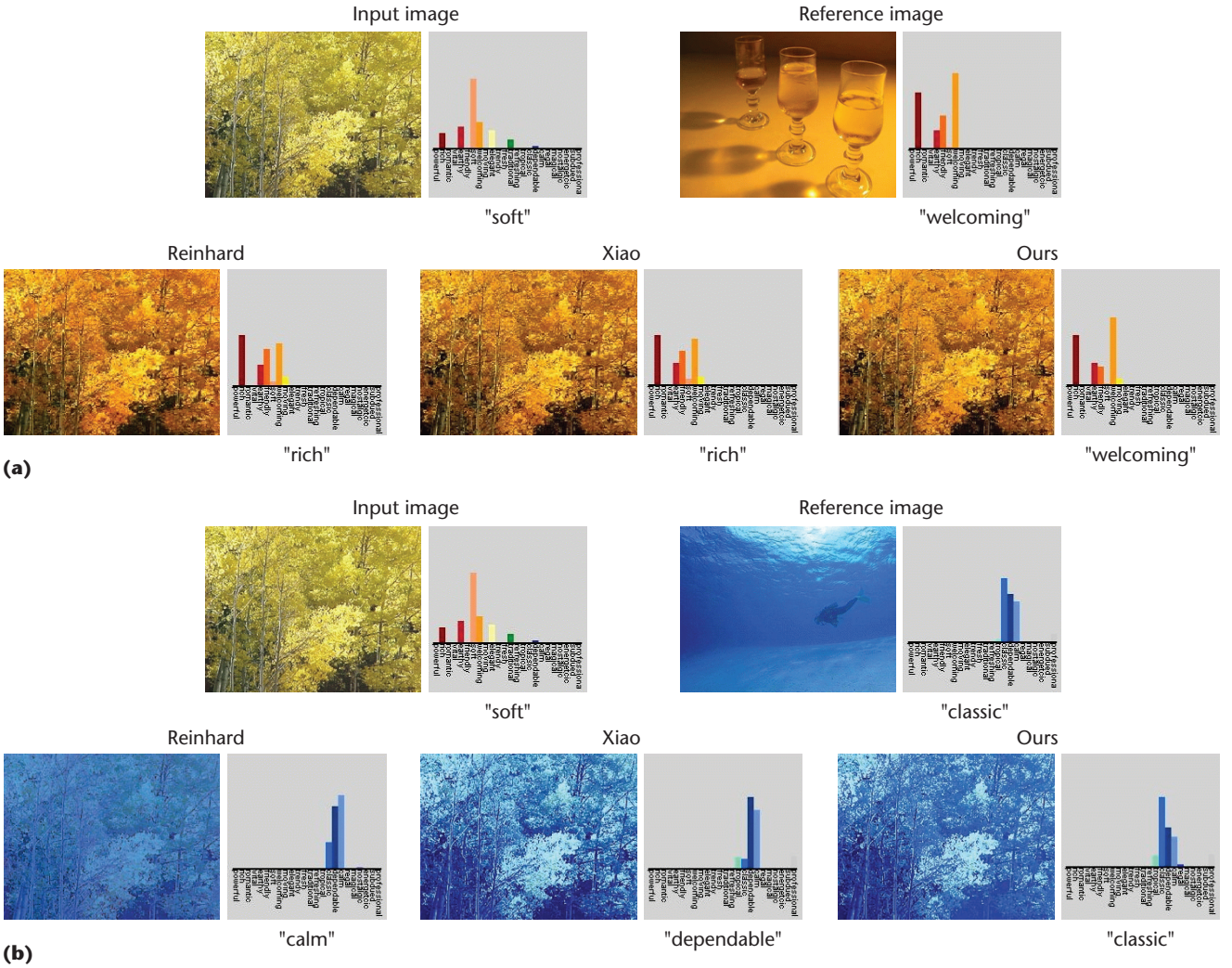

Figure 7. Comparisons of our color-mood adjustment algorithm with other approaches.

where w_{ref} and h_{ref} represent the reference image's width and height, respectively.

To ease the description, we define:

$$Diff(i) = H_{should\_output}(i) - H_{output}(i) \quad (4)$$

where $H_{output}(i)$ is the current number of pixels that fall into the output image's i-th color-mood category. Therefore, $Diff(i)$ represents the difference between the current output and target output images with respect to the i-th color-mood category. Finally, the total difference between the current output and target output images, in terms of color-mood histogram, is

$$H(should\_output, output) = \frac{\sum_{i=1}^{24} |Diff(i)|}{w_{output} \cdot h_{output}} \quad (5)$$

Finally, we judge the effectiveness of color conversion for the output image. To do this, we judge the j-th image belonging to the mood image database's target color mood using the following formula:

$$E(j) = w_1 L(input, output) + w_2 H(should\_output, output) \quad (6)$$

where we use j = 1, 2, ...5 ~ 10, and w_1 and w_2 as weighting factors to combine two types of differences into a unified metric. In our current implementation, we set both w_1 and w_2 at 0.5. According to this formula, we output only the image that produces the least such value. In other words, what we want is to strike the best balance between maintaining the content (or brightness) similarity to the input image, while at the same time achieving color-mood histogram resemblance to the reference image. Alternatively, we could sort all the output image's corresponding values and display each of them accordingly to let users make the final choice.

Results

We conducted experiments on a Pentium 4 2.8 GHz machine with 480 Mbytes of memory, running on Windows XP. Figure 7 compares our method's results with that of the Reinhard and Xiao approaches on two example sets. As the fig-

Histogram Matching and Spatial Coherence

To enhance flexibility, our system provides a histogram matching functionality to create an output image histogram that is as close as possible to that of the reference image. Such a functionality could be quite useful, especially with a more complete mood image database. For example, after users identify the target mood for their input image, our system pops up a window containing all constituent images of the desired mood, thus letting users explicitly specify the reference image. The system then adjusts the input image to strive for the best resemblance to the reference image by matching their relative histograms. To match two histograms, the system iteratively adjusts the color until it reaches a desired histogram similarity. This matching process begins following the color-mood transfer process and the global alignment transformations.

Key terms

Before plunging into the histogram matching algorithm's details, we first define several terms, starting with $M(should\_output, output)$, which we define as:

$$M(should\_output, output) = \sum_{i=1}^{24} |Diff(i)| \quad (7)$$

where the $Diff(i)$ definition is the same as in equation 4. As long as $M(should\_output, output)$ is larger than c_1, the iterative histogram matching process continues; otherwise the process stops, as we regard the two histograms as already similar enough. In our current implementation, we set the threshold value of c_1 to 10 percent of the output image's total number of pixels. Moreover, according to $Diff(i)$'s value, during the following process, we classify the each color mood i's state as either stable, surplus, or deficient according to the following rules:

- stable: $|Diff(i)| < c_2$
- deficient: $|Diff(i)| > c_2$ and $Diff(i) > 0$, and (8)
- surplus: $|Diff(i)| > c_2$ and $Diff(i) < 0$

where c_2 is another threshold value (currently set at 1 percent of the output image's total number of pixels). As the names of these states suggest, surplus and deficient indicate that color mood i contains too many and too few pixels, respectively, compared with the desired number of pixels. Stable indicates that color mood i has already reached an approximate equilibrium and thus requires no adjustment.

The algorithm

Our algorithm is composed of four steps. In step 1, we calculate $M(should\_output, output)$ and determine if the iteration should stop. Assuming we continue, in step 2 we perform image segmentation based on the 24 color moods, initially labeling each segment as "untouched."

In step 3, we find the color mood—say, color mood j—that has the largest value of $|Diff(j)|$. If its state is not stable, we must consider two cases. In case 1, the state of color mood j is deficient—that is, we want more pixels—so we look for segment S, which satisfies the following criteria:

- it doesn't belong to color mood j;
- it's marked as untouched;
- it belongs to a color mood with a state of surplus;
- it has fewer pixels than $|Diff(j)|$; and
- among all segments satisfying the previous criteria, segment S is the closest to color mood j, in terms of the distance between its centroid and color mood j's representative color.

Once we locate a segment, we adjust its pixel colors by averaging them with color mood j's representative color to increase the number of pixels that will be classified into color mood j. We then mark this segment as "adjusted." In case 2, where color mood j's state is surplus—that is, we want fewer pixels—we find color mood j's closest color mood k that is in the deficient state, where we define the distance between two color moods as the distance between their representative colors. We then look for a segment S belonging to color mood j that satisfies the following criteria:

- the size of S is less than color mood k desires (that is, $|Diff(k)|$, and
- among all segments meeting the previous criterion, segment S is the closest segment to color mood k. By averaging segment S's pixel colors with color mood k's representative color, we can increase the number of pixels in color mood k, while decreasing the number of pixels in color mood j. We then mark Segment S as "adjusted."

In step 4, if we can't find any untouched segment for color adjustment in step 3, we go back to step 1 to recalculate the color-mood distribution, perform segmentation, and adjust the color until we reach a convergence. Thereafter, if we still can't find an untouched segment for color adjustment, it's because the smallest such segment is still too big. To solve this, we break the segment into even smaller regions, according to the internal pixels' brightness values, and return to step 1 to find further color adjustment opportunities.

As our histogram matching process shows, our algorithm adjusts color based on segments—that is, it accounts not just for pixel colors, but their spatial properties as well. Consequently, our approach might produce satisfactory results in many cases. However, as we note in our article's conclusion, it can also occasionally incur other type of artifacts.

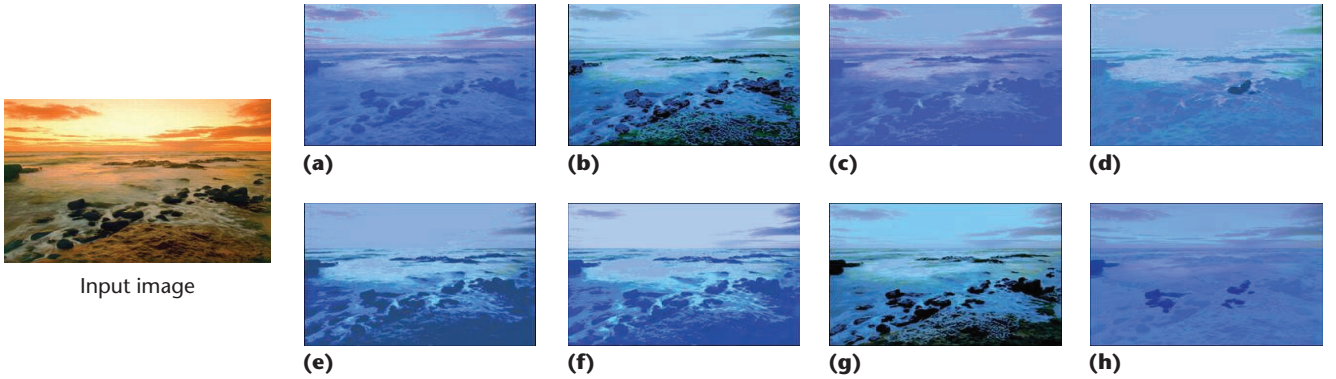

Figure 8. The output-image selection algorithm. The images above show the results of our algorithm as applied to Figure 2's corresponding reference images. As the output image, we chose image g, which has the least value as calculated from equation 6.

Figure 9. Comparisons of the proposed histogram algorithm with existing approaches. These results compare the approaches' results on the "trendy" reference color mood.

ure shows, all three methods perform equally well on the first example, but our approach generated a slightly better result on the desired percentage of orange. In the second example, our approach clearly outperforms the other two, better capturing the "classic" mood.

Figure 8 shows an example of how our output-image selection algorithm works on Figure 2's reference images. This output-image selection scheme further guarantees the mood transfer's success and quality.

Figure 9 compares our proposed histogram matching algorithm with the other two approaches. As the figure shows, Reinhard and colleagues' approach seems to have already converted the input image into the right mood, but a glance at our image shows how our histogram matching process further improves the results. Also, note that the observable minor histogram differences are due to the tolerance threshold setting of c_1 (10 percent of total number of pixels).

Limitations and future work

Though it offers a better spatial coherence, our proposed method nonetheless has its limitations. First, as Figure 10 shows (next page), it's possible for two adjacent, similarly colored pixels to end up in different color-mood partitions. Their color difference can also be further enlarged during color adjustment, leading to undesired artifacts. Second, our color adjustment process sometimes stops before a segment has been fully converted to a different mood (as in Figure 10). This is mainly because our method doesn't require a perfect match between the number of pixels and a segment's adjustable area. In any case, such artifacts might be unavoidable. Imagine, for example, a case in which we're asked to convert an image with one or two colors to a reference image with a larger tonal variety. In such cases, fragmented output images are inevitable.

In the future, we'll strive not only to solve our approach's imperfections, but also to generalize our framework and make it more complete. We

might, for example, adopt a different or more complete scheme of color-mood definitions. Moreover, we currently represent a particular mood through a single color. Because similar colors often convey similar moods, however, we might try to use color combinations or color distributions to represent or model each mood. Also, it could be interesting to generalize our framework to video or animation. Finally, we might build a more complete mood image database by, for example, adopting a Wikipedia-like approach. That is, if users are unsatisfied with existing images, they could add and classify their own reference images, and do so either automatically or manually, to enhance flexibility.

Acknowledgment

The authors thank Prof. Lin-Lin Chen and Prof. Rung-Huei Liang for their valuable comments. They also gratefully acknowledge the support by the National Science Council, Taiwan, under grants NSC 96-2219-E-001-001, NSC96-2219-E-011-008, and NSC-96-2221-E-011-139.

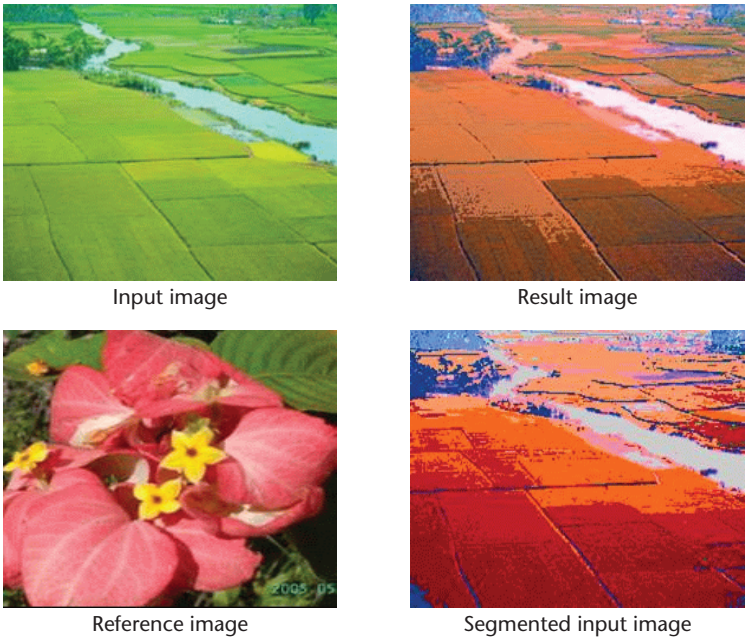

Figure 10. An example showing the limitations of our approach. Two adjacent, similarly colored pixels can end up in different color-mood partitions.

References

1. E. Reinhard et al., "Color Transfer between Images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, 2001, pp. 34-41.
2. B.M. Whelan, *Color Harmony 2*, Rockport Publishers, 1994.
3. J. Morovic, J. Shaw, and P. Sun, "A Fast, Non-Iterative and Exact Histogram Matching Algorithm," *Pattern Recognition Letters*, vol. 23, nos. 1–3, 2002, pp. 127-135.

Chuan-Kai Yang *is an assistant professor in the Department of Information Management, National Taiwan University of Science and Technology. His research interests include computer graphics and multimedia systems. He received a PhD in computer science from Stony Brook University. Contact him at ckyang@cs.ntust.edu.tw.*

Li-Kai Peng *is a junior engineer in CyberLink Corporation. Her research focuses on computer graphics, with an emphasis on color processing. She received an MS in information management at National Taiwan University of Science and Technology. Contact her at M9409101@mail.ntust.edu.tw.*

For further information on this or any other computing topic, please visit our Digital Library at http://www.computer.org/csdl.