Example-Based Color Stylization of Images

YOUNGHA CHANG, SUGURU SAITO, KEIJI UCHIKAWA, and MASAYUKI NAKAJIMA
Tokyo Institute of Technology

We describe a new computational approach to stylize the colors of an image by using a reference image. During processing, we take the characteristics of human color perception into account to generate more appealing results. Our system starts by classifying each pixel value into one of the basic color categories, derived from our psychophysical experiments. The basic color categories are perceptual categories that are universal to everyone, regardless of nationality or cultural background. These categories are used to provide restrictions on color transformations to avoid generating unnatural results. Our system then renders a new image by transferring colors from a reference image to the input image, based on these categorizations. To avoid artifacts due to the explicit clustering, our system defines fuzzy categorization when pseudocontours appear in the resulting image. We present a variety of results and show that our method performs a large, yet natural, color transformation without any sense of incongruity and that the resulting images automatically capture the characteristics of the colors used in the reference image.

Categories and Subject Descriptors: I.4.5 [**Image Processing and Computer Vision**]: Reconstruction—*Transform methods*; I.4.6 [**Image Processing and Computer Vision**]: Segmentation—*Pixel Classification*; I.4.10 [**Image Processing and Computer Vision**]: Image Representation—*Statistical*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Basic color categories, color transfer, example-based

1. INTRODUCTION

Color is one of the most influential and powerful elements in images and various color-processing methods have been proposed in the field of image processing. Results sometimes do not match human perception, however, because they capture the characteristics of color by treating them as signals. In this paper, we propose a method that takes human perceptual characteristics into account to produce attractive color transformations.

The colors of images are often changed using image editing tools, which work very well for global color transformations. For more detailed color editing, such as transforming a blue sky into a pale blue sky, green leaves into a more yellowish green, or orange flowers into a more vivid reddish hue, a user's manual interaction is required to produce a satisfying result. In addition, an attractive result can usually be obtained only if the user has a good aesthetic sense. We aim to implement a more intuitive, flexible, and automatic color transformation system that stylizes the colors of images based on a provided reference image.

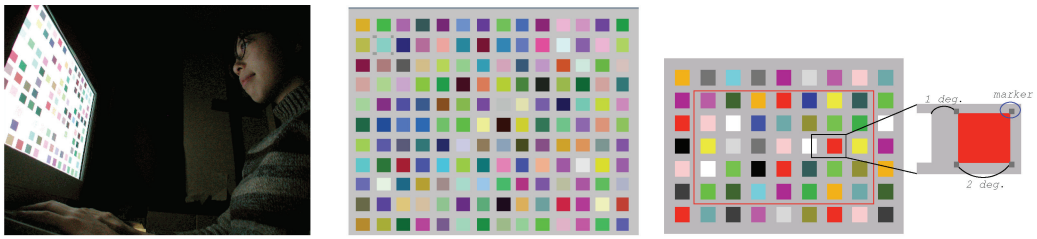
There is a related work that enables an easy and effective color correction by referencing another image [Reinhard et al. 2001]. No artistic talent is required for this color transformation, because it only

Authors' address: Youngha Chang, Suguru Saito, Nakajima Masayuki, Dept of computer science, Tokyo Institute of Technology, Oookayama2-12-1, Meguroku, Tokyo, Japan; email: chang@img.cs.titech.ac.jp; Keiji Uchikawa, Dept of information processing, Tokyo Institute of Technology, Nagatsuda 4259, Midoriku, Yokohama, Japan.

Fig. 1. Demonstration of our automatic color transformation result. The left image shows an input photograph and the middle image shows a reference image. Our algorithm automatically gets tendencies of color use from the middle image and transfers them to the input image. The resultant image is as shown in the right image.

requires the user to set a reference image. This method uses a simple statistical analysis and works very successfully; however, it tries to represent the overall characteristics of color use in an image using only two statistical values per color channel. It then transforms all pixel colors in an image according to the same statistical information. Therefore, when the source and target images are not compatible, a user has to manually set swatches for each color region and define a match between them. A similar approach is used to solve a challenging problem, colorizing gray scale images by example [Welsh et al. 2002]. This method matches luminance values between a gray scale image and a color image, and then transfers matched color values from the color image. This approach is also simple and, by using swatches, we can successfully transfer colors. Yet, there still remains the issue that it requires manual operation for some input images.

We aim to be more detailed and natural color transformation. Thus, it does not try to capture the overall characteristics of color use, but instead captures the characteristics for each color unit. A color unit is a perceptually agreeable color category that provides restrictions on the color transformations to ensure that the resulting images will not appear odd. That is, color transformations are performed within the same perceptual color category. Our algorithm first categorizes pixels into perceptually agreeable color categories and measures the characteristics of the color spread in each category. Then, for each pixel in the input image, it finds a matching color value from the same category of the reference image, and replaces the pixel color value. A user interacts with our system only to provide an input image on which to transfer color, and a reference image, which provides an example of color use. Therefore, no artistic talent is required for the color transformation, because the user only sets his favorite image as a reference image. The most important point of this paper is how perceptually agreeable color categories are generated. For this, we adopt the Basic Color Terms (BCTs), which was originally reported by Berlin and Kay [1969]. They examined 98 languages from several families and found that developed languages have 11 universal BCTs, with the types and color ranges of these being similar across the languages. For example, in English, the BCTs are black, white, red, green, yellow, blue, brown, purple, pink, orange, and gray. Many later researchers have confirmed this idea [Uchikawa and Boynton 1987; Rosch Heider 1972]. In our system, we divide the color space into basic color categories, and perform the color transformation within each category. In Section 2, we describe our experiment to determine the spread of each basic color category. In Section 3, we explain a method for determining the tendency of pixel spreads in each basic color category, and in Section 4, we discuss a method for performing the color transformations. We show various results in Section 5 and, in Section 6, we conclude and give suggestions for future work.



(a) experimental environment (b) stimuli arrangement (c) naming range(inside red rectangle), stimulus size, markers

Fig. 2. Experiment on color naming.

2. A PSYCHOPHYSICAL EXPERIMENT FOR ESTIMATING BASIC COLOR CATEGORIES

Several studies have aimed at capturing the spread of basic color categories. In most of these, one color stimulus was shown at a time with achromatic surroundings. In images, however, the color appearance of a pixel is often affected by its surrounding colors [Fairchild 1997]. This shift of apparent color is particularly seen when the color has very low saturation. These colors are named chromatic, because of the color contrast with their achromatic surroundings. Therefore, we develop a new visual psychophysical test to avoid this problem.

In our new test, we show various color stimuli in one screen as shown in Figure 2(b). We do not set surrounding conditions as one single color nor as a Mondrian collage of random color patches, because (1) the purpose of this experiment is not to measure the exact effects of color contrast for all possible color combinations, but to model an environment where various color stimuli are mixed together in one image; (2) our aim is to carefully avoid the color-assimilation effect, because if we arrange color patches without space between stimuli, many low saturated colors will be perceived as chromatic, because of the color-assimilation effect; (3) the achromatic background used here indirectly indicated that the illumination is an achromatic, $D65$ color.

In Section 2.1, we describe the details of the participants and stimuli choice; in Section 2.2, we describe the details of our experimental procedure.

2.1 Participants and Stimuli

We collect color-naming data from five men and five women, aged 22 to 33; all have normal color vision.

We assume that we use a monitor that is fully compliant with the *sRGB* specification. In this experiment, we use *FlexScan T*766. Because we plan to perform equal interval samplings, we sample test color stimuli from the CIE1931 xy chromaticity diagram at several luminance levels. We select luminance values Y of 2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 95, 98, and 100, where Y is the linear luminance value and $Y = 100$ is referenced to the maximum luminance level of the monitor. We perform a regular sampling on the xy plane with intervals of 0.025 units at each luminance level. Because these experimental results are to be used only for colors appearing in digital images, we choose color samples within the scope of *sRGB*. The total number of test colors is 1254.

However, in fact, because the color gamut of our monitor is slightly misaligned from *sRGB*, we carefully calibrate the monitor by measuring red, green, and blue gamma curves and obtain the maximum luminance levels of red, green, blue, and white. Whenever we use our experimental results, we use corrected color values based on this calibration.

2.2 Procedure

In a dark room, we present color stimuli on an *sRGB* monitor as shown in Figure 2(a). Following 10 min of dark adaptation, the test stimuli are presented in random order, as shown in Figure 2(b). The arrangements of the test stimuli are different in every experiment. The viewing angle of each color stimulus is set at 2° and the space between stimuli is set at 1° of the viewing angle, as shown in Figure 2(c). Color stimuli are arranged on a uniform gray field. The *sRGB* value of the surrounding gray field is $sRGB = (srgb[0.5], srgb[0.5], srgb[0.5]) = (186, 186, 186)$, where $srgb(x)$ is defined as Eq. (1).

$$srgb(x) = \begin{cases} 255. * [(x + 0.055)/1.055]^{2.4}, & if\ x > 0.03928 \\ 255. * (x/12.92), & otherwise \end{cases} \tag{1}$$

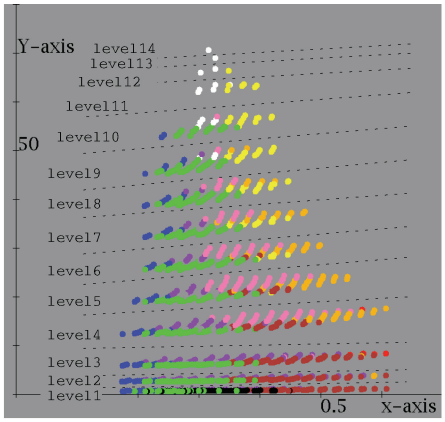
The participant observes each stimulus for several seconds 50 cm away from the monitor. The duration is not limited, so the participant can observe it as long as is needed to decide on the color response. The participant names each color by using one of the 11 BCTs—red, green, yellow, blue, orange, purple, brown, pink, white, gray, or black. This is achieved by typing one of 11 keys on the keyboard. After typing, each participant receives a voice feedback confirming his choice. If he makes a mistake in typing a key, he can rename that stimulus. The naming process is done for each stimulus in a raster order except for the color stimuli that bordered the edge of the screen [Figure 2(c)]. This is because the stimuli outside the red square have fewer interactions with surrounding color stimuli. Because so many stimuli are shown on one screen, there might be a possibility that the participant confuses which stimulus to fixate on next. To avoid this problem, we place small markers at the corners of the current fixated stimulus, as shown in Figure 2(c). After the test stimuli on one screen are completed, the screen is filled with gray for 2 s to minimize the influences of any afterimages. After the experiments are finished, the color stimuli are grouped according to each participant's categorization and these sorting results are shown to the participant for confirmation. If the participant finds mistakes in this step, he can rename that stimulus. The participant responses are recorded in a computer.

2.3 Experimental Result

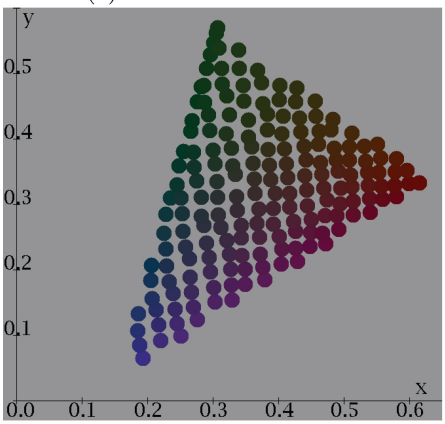
We show the overall experimental results in Figure 3(a). Because we show corrected stimulus values based on our monitor calibration, the color values are slightly shifted from their regularly sampled position. Although the luminance values are slightly different from each other, in Figure 3(b-1–o-2), we attempt to visualize the color stimuli and responses on the two-dimensional plane. We show the *sRGB* value of each datum for reference, but note that their surrounding color conditions were different from those during the experiments, which means that the appearance of each color might be different.

The small pie charts represent colors according to the participant responses to the color stimuli. If all the responses for a test stimulus are the same, the pie chart is filled with one color. If the responses differ between experiments, the pie chart shows these different responses through arcs with different angles proportional to the number of ratings. For example, if all the responses are "black," the pie graph is filled with black. If three of the participants response the stimulus as red and seven response it as brown, then 30% of the pie chart are filled with red and 70% are filled with brown.

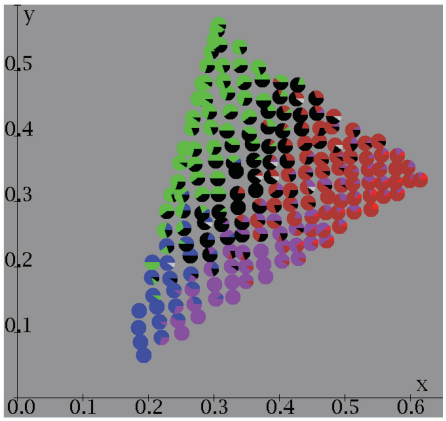
For discussing the effectiveness of these experimental results, we compare them with our previous report [Uchikawa et al. 1994]. We attempt to clarify the main differences of the two approaches in Table I. Note that although the sampling rate of the test color is the same in both cases, we cannot directly compare them. This is because the sampling positions are changed slightly after data correction and the sampling domains are different. Therefore, we only compare test colors that are placed within the cross-domain of these two.



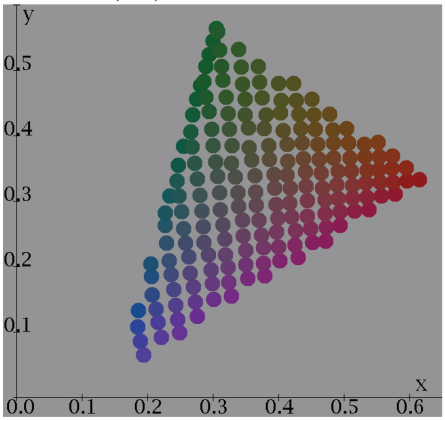
(a) overall view of results



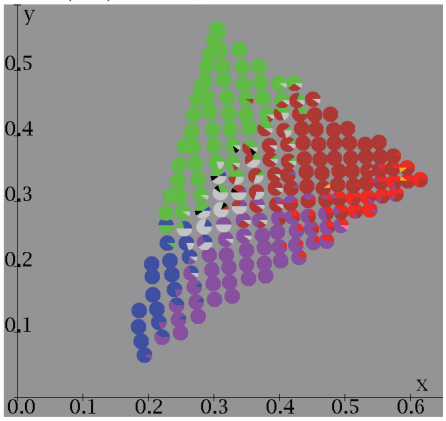
(b-1) colors at *level*1



(b-2) naming results at *level*1

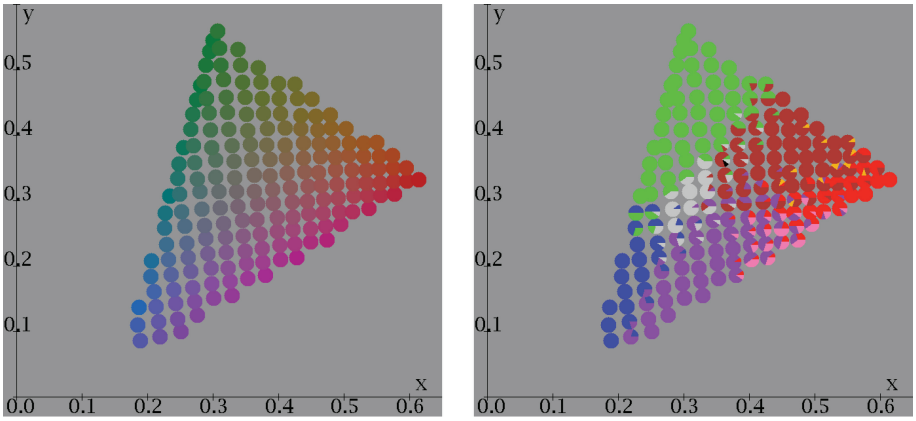


(c-1) colors at *level*2

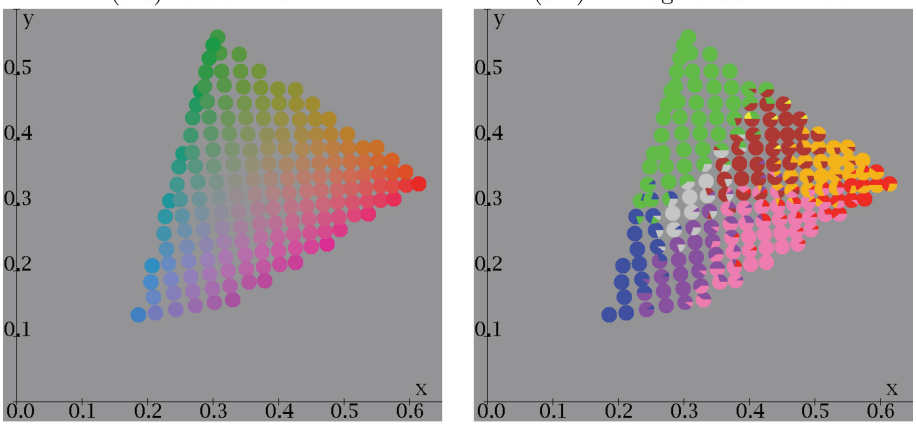


(c-2) naming results at *level*2

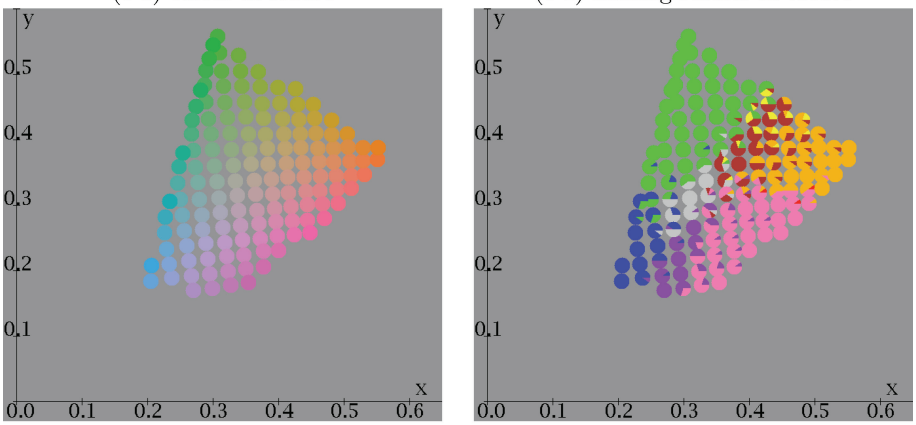
Fig. 3. Results of our psychophysical experiment. (Continued).



(d-1) colors at *level*3 (d-2) naming results at *level*3

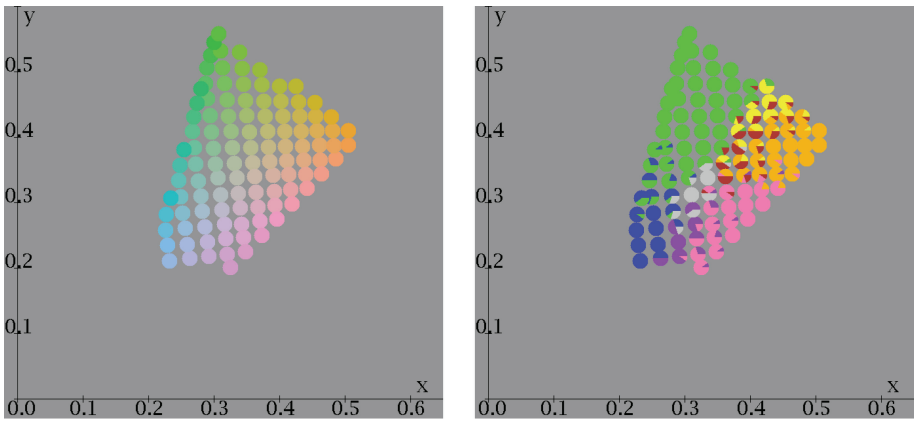


(e-1) colors at *level*4 (e-2) naming results at *level*4

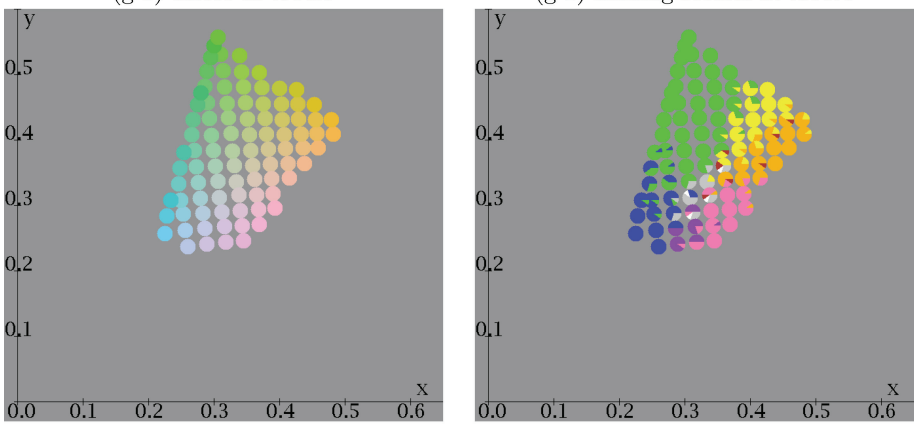


(f-1) colors at *level*5 (f-2) naming results at *level*5

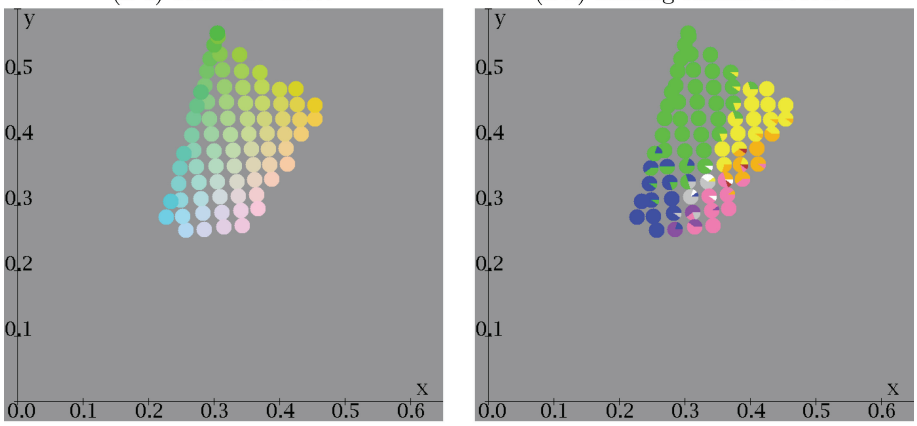
Fig. 3. (Continued).



(g-1) colors at *level*6 (g-2) naming results at *level*6

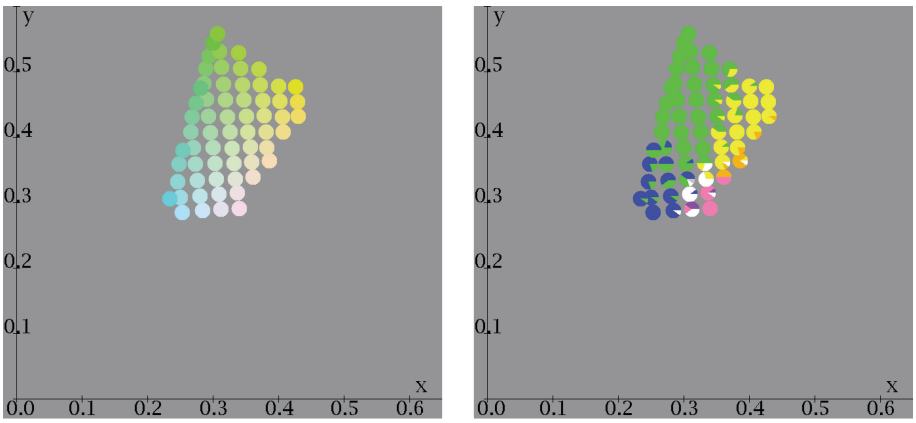


(h-1) colors at *level*7 (h-2) naming results at *level*7

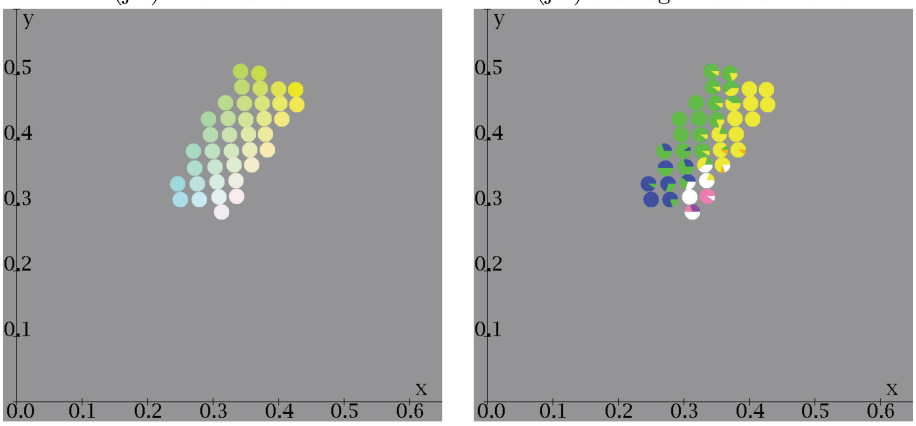


(i-1) colors at *level*8 (i-2) naming results at *level*8

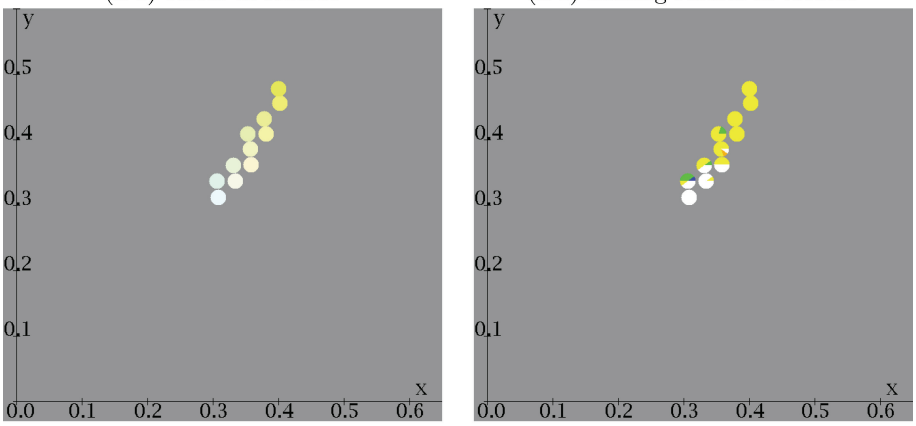
Fig. 3. (Continued).



(j-1) colors at *level*9 (j-2) naming results at *level*9

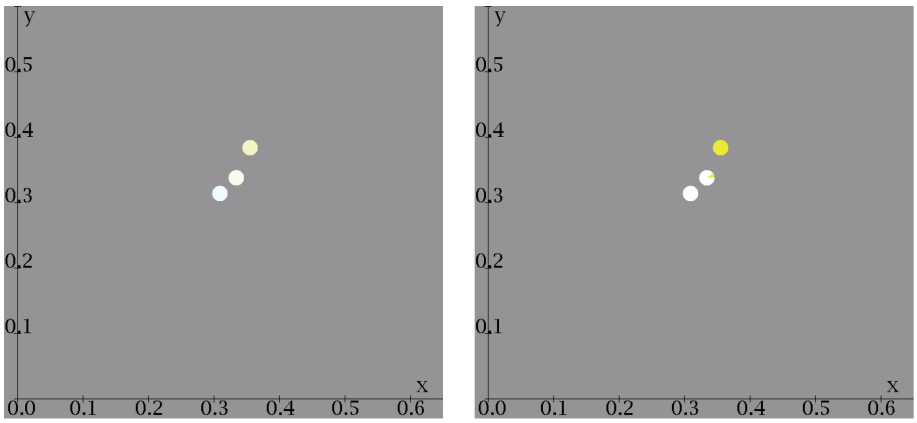


(k-1) colors at *level*10 (k-2) naming results at *level*10



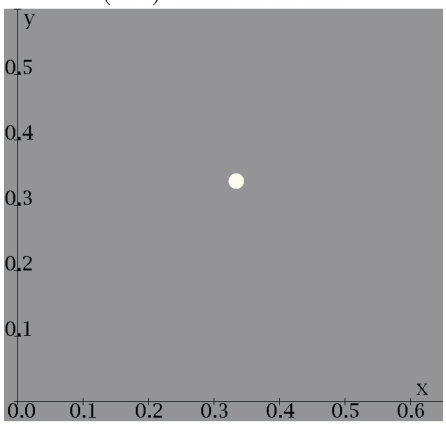
(l-1) colors at *level*11 (l-2) naming results at *level*11

Fig. 3. (Continued).

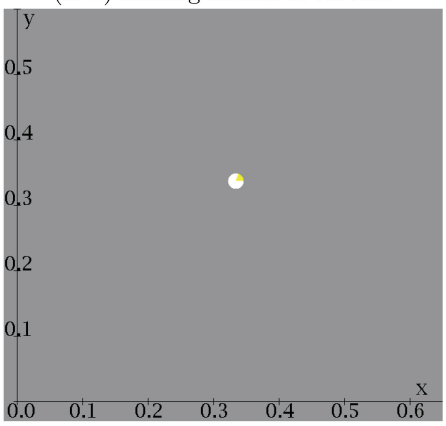


(m-1) colors at *level*12

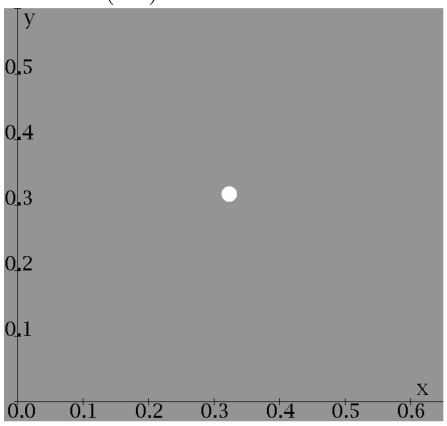
(m-2) naming results at *level*12



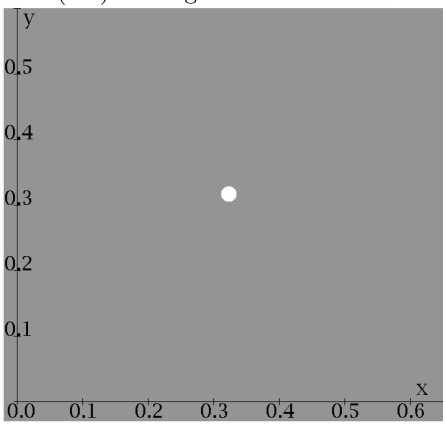
(n-1) colors at *level*13



(n-2) naming results at *level*13



(o-1) colors at *level*14

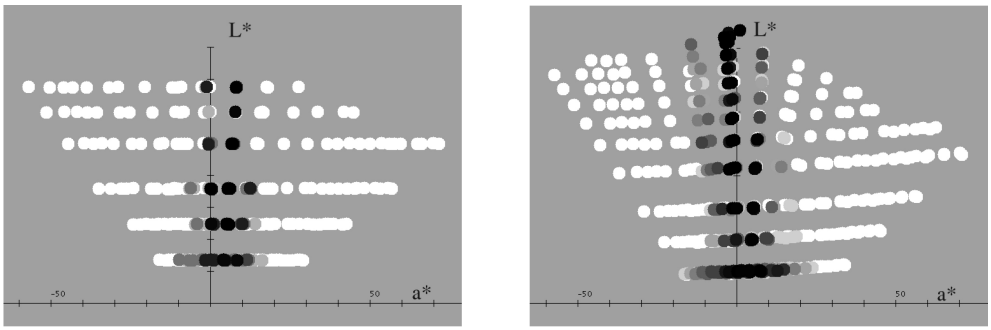


(o-2) naming results at *level*14

Fig. 3. (Continued).

Table I. Differences between Previous and New Work

	Previous Work	New Experiment
# of stimuli	802	1254
# of participant	3	10
# of experiment	6(twice/person)	10(once/person)
# of stimulus in one screen	one stimulus	various stimuli
background	gray	various colors on gray



(a) Previous work [Uchikawa et al. 1994] (b) New experiments

Fig. 4. Comparison with results of our previous work and new work.

There are no significant differences in the color-naming results near focal colors and responses are in agreement with each other. For colors placed near the boundary of basic color categories, responses are ambiguous for both methods. However, in our new experiment, the subjects' responses are more ambiguous for some color stimuli. We define "variability" as the average number of BCTs used to describe each color sample. The variability for the new experimental results is 1.81, while that for the previous results is 1.47. This is because we use one single achromatic surrounding color in our previous work, which confines the color appearance of the test stimulus to one specific environment. On the other hand, in our new method, we randomly arrange a variety of colors near the test stimulus and these stimuli affect the color appearance of the test stimulus. Therefore, we can obtain a variety of naming probabilities in various situations, which is useful for practical purposes.

Next, the achromatic categories are expanded in our new method. We define "achromaticity" as the percentage of test color that is described as "black," "white," or "gray" in at least one experiment. The achromaticity of these new experimental results is 15.5%, while that of previous results is 11.9%. In particular, at luminance level 2, achromaticity in our new method is 80.6% whereas in our previous method it is 33.3%. In our previous work, because only achromatic colors are shown at the periphery, the saturation of the test stimulus is emphasized by saturation contrast. Therefore, most of the colors tend to be named as chromatic colors, which results in small achromatic regions. On the other hand, because we put various color stimuli around the test stimulus in our new method, it better enables participants to perceive achromaticity for very low luminance colors.

We visualize some of our experimental results on the *CIE* $L^*a^*b^*$ color space and show the expansion of achromatic regions in Figure 4. We show the data whose a^*b^* values are located between -15 and 15° from the a^* axis on the a^*b^* plane. For showing the naming results of low saturated colors, we also show those colors whose distance from the L^* axis is less than 20 on the a^*b^* plane. For visibility, we plot each naming result in a gray scale according to the percentage of achromatic naming. If the test color is always named as black, gray, or white, then the test color is shown in black; if the test color

is never named as black, gray, or white, then the test color is shown in white. We can see that in our new experiment, the achromatic regions are expanded. Moreover, because our new experiment was performed for lower and higher luminance levels than the previous one, we can estimate the naming of lower and higher luminance colors more effectively.

The effects of expanded achromatic categories will be shown in Figure 9, in Section 5.

3. CALCULATING THE FEATURES OF COLOR SPREAD FOR EACH BASIC COLOR CATEGORY

Our purpose is to generate a synthetic image that takes on the color characteristics of a reference image. For achieving this, we first compute the feature of color distribution for both the input and reference images. In this work, we calculate the centroid, shape and location of the color distribution in each basic color category. These features will be used for the color-transformation procedure, described in Section 4. Note that all the procedures in Section 3 and 4, except those in Section 3.3, are performed on the *CIE* $L^*a^*b^*$ space, which is used because it is a perceptually uniform color space.

In Section 3.1, we classify each pixel into one of basic color categories. In Section 3.2, we explain our method for capturing the feature of color distribution. In Section 3.3, we discuss a case where some pixels of the input image are in a basic color category, while pixels of the reference image are not in that category.

3.1 Automatic Pixel Naming

We described our psychophysical experiment in the previous section. Based on these experimental results, our algorithm clusters the color space into eleven basic color categories, and the algorithm carries out automatic color naming based on these clusters.

For this, the algorithm first categorize the color space using the experimental results. The algorithm is used to transfer all the test colors of our color-naming experiments from the *Yxy* color space to the *CIE* $L^*a^*b^*$ color space. It then divides the color space into 11 basic color categories, in a manner similar to Voronoi tessellation, by treating each data point as a site. Finally, the algorithm combines the Voronoi cells belonging to the same basic color category. The basic color category of a cell is the category that gives the highest rating for its site.

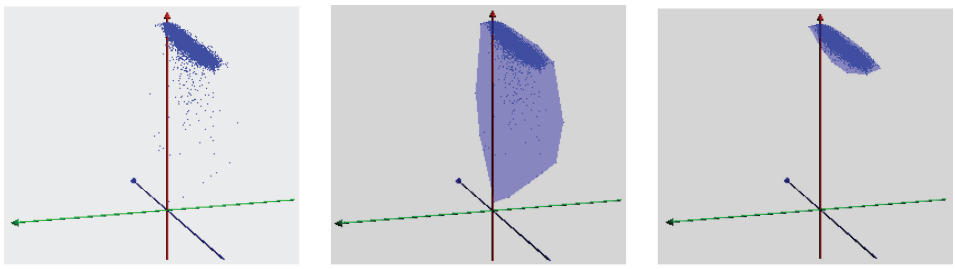
The algorithm then performs automatic color naming for every pixel color of an image. For this, it transfers each pixel color value, p_i^{in},[1] to the *CIE* $L^*a^*b^*$ color space, and determines which basic color category it belongs to.

3.2 Generation of Basic Convex Hulls

In this section, we describe our method of capturing the location, shape, and center of each basic color category. For this, we define a convex hull that encloses all of the pixel color values within a category. We refer to these as basic convex hulls.

However, in some cases, convex hulls cannot capture the features of the color distribution effectively. For example, if pixel colors of an image that belongs to basic color category C_i have the distribution shown in Figure 5(a), the process described above generates a basic convex hull ch_i^{in} as shown in Figure 5(b). This contains large unused color regions because of the very small number of isolated pixel points. This results in the colors of many pixels in the input image being transformed into the unused

[1]In this paper, a superscript on the right side of a variable indicates where the variable belongs: p^{in} means that the variable is from an input image, p^{ref} means that the variable is from a reference image, and p^{univ} means that the variable is from universal color space. Variables without a superscript are from either an input image or a reference image. A subscript on the right side of the variable, i, indicates the basic color category to which the variable belongs: $i = \{black \mid white \mid red \mid green \mid yellow \mid blue \mid brown \mid pink \mid orange \mid purple \mid gray\}$. Variables shown in bold are vectors, whereas variables not shown in bold are points.



(a) pixel spread (b) convex hull for (a) (c) after reconstruction

Fig. 5. A problem of representing color features by convex hull.

color regions in the reference image. In this case, we remove such isolated and very sparse points so that the algorithm can capture the color features of each color category more efficiently. This process also helps to reduce any artifacts caused by image noise. For rejecting outliers, we used the squared-Mahalanobis distance that is one of the most promising metrics for detecting outliers. For each pixel color p in C_i, the algorithm is used to calculate the squared-Mahalanobis distance using Eq. (2),

$$d^2 = (\boldsymbol{p} - \boldsymbol{c_i})^t \; V_i^{-1} (\boldsymbol{p} - \boldsymbol{c_i}) \tag{2}$$

where V_i is the covariance matrix for pixel colors in C_i, and c_i is the center of gravity of pixels in C_i. Next, the algorithm serializes the pixels in d^2 order and checks for any significant changes in the d^2 values in the sequence. If there is such a point in the sequence and its order in the sequence has a percentage larger than a threshold t, we delete the points having larger d^2 values. In our work, we used $t = 97$. Our algorithm then regenerates the basic convex hull ch_i using the remaining pixel color values within the category, as shown in Figure 5(c). Finally, it remaps the deleted points onto points where the vector from $\boldsymbol{p_i}$ to $\boldsymbol{c_i}$ intersects with the boundary of ch_i.

3.3 Estimating Basic Convex Hulls with No Direct Reference

In some cases, a user might select an input image having pixels p_i^{in} from category C_i, although the reference image has no pixels from C_i. In this case, the algorithm has no reference to transfer p_i^{in}. We propose two solutions to this problem.

The first is simple and can reflect each user's preference. Because our algorithm allows multiple reference images to be set, another reference image can be chosen for category C_i.

However, it may be difficult to prepare harmonious multiple reference images based on their color use. The second approach automatically estimates suitable color distributions of C_i. For this, we statistically analyze other existing categories and estimate the center of gravity c_i^{ref} and the shape of ch_i^{ref}. To do this, the algorithm first estimates the hue, saturation, and brightness of c_i^{ref} separately in the *HSB* space. The values of saturation and brightness for c_i^{ref} can be estimated using Eq. (3),

$$\begin{aligned} c_{Si}^{ref} &= c_{Si}^{univ} + \sum_{j \in T} w_j \left(c_{Sj}^{ref} - c_{Sj}^{univ}\right), \\ c_{Bi}^{ref} &= c_{Bi}^{univ} + \sum_{j \in T} w_j \left(c_{Bj}^{ref} - c_{Bj}^{univ}\right), \\ w_j &= \frac{n\left(ch_j^{ref}\right)}{n(I^{ref})} \end{aligned} \tag{3}$$

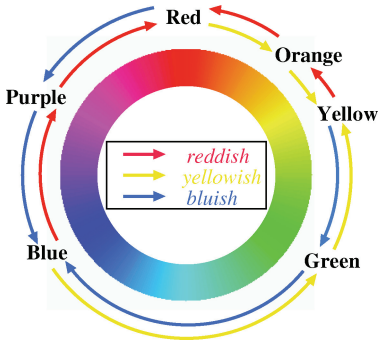


Fig. 6. The color circle.

where subscript S and B represent saturation and brightness, respectively. T is a set of basic color categories that exist in the reference image and c_j^{univ} is the foci of category i. The foci is chosen by participants during the psychophysical experiment by asking them to pick out the best examples of each category. n(A) is the number of pixels in A and I_{ref} is the reference image. This equation means that if colors of other existing categories are of low saturation, the saturation of c_i^{ref} is also estimated as a low. However, hue cannot be calculated in a similar way because the meaning of hue is not originally one dimension. The hue value is represented in the phase angle. The values of hue corresponds to color as shown in Figure 6. If hue is 0 or 1, the color is red. The hue value increases as we move clockwise around the color circle shown in Figure 6.

Thus, instead of calculating a mean value of hue, the algorithm calculates the variables of *yellowish*, *bluish*, and *reddish* separately for each category. We choose these three variables because they are the minimal variable set that can describe all the color shifts on the color circle. For example, if red becomes *yellowish*, then it becomes orange; if orange becomes *yellowish*, then it becomes yellow; if yellow becomes *bluish*, then it becomes green, and so on. The *reddish*, *yellowish* and *bluish* values of the reference image can be computed as follows:

$\mathrm{r} = 0; \quad \mathrm{y} = 0; \quad \mathrm{b} = 0;$
for each $\mathrm{j} \in \mathrm{T}$
$x_j = w_j\left(c_{Hj}^{ref} - c_{Hj}^{univ}\right)$

j = red:	if($x_j \geq 0$)	$\mathrm{y} = \mathrm{y} + x_j$	else	$\mathrm{b} = \mathrm{b} - x_j$
j = orange:	if($x_j \geq 0$)	$\mathrm{y} = \mathrm{y} + x_j$	else	$\mathrm{r} = \mathrm{r} - x_j$
j = yellow:	if($x_j \geq 0$)	$\mathrm{b} = \mathrm{b} + x_j$	else	$\mathrm{r} = \mathrm{r} - x_j$
j = green:	if($x_j \geq 0$)	$\mathrm{b} = \mathrm{b} + x_j$	else	$\mathrm{y} = \mathrm{y} - x_j$
j = blue:	if($x_j \geq 0$)	$\mathrm{r} = \mathrm{r} + x_j$	else	$\mathrm{y} = \mathrm{y} - x_j$
j = purple:	if($x_j \geq 0$)	$\mathrm{r} = \mathrm{r} + x_j$	else	$\mathrm{b} = \mathrm{b} - x_j$
j = brown:	if($x_j \geq 0$)	$\mathrm{y} = \mathrm{y} + x_j$	else	$\mathrm{r} = \mathrm{r} - x_j$
j = pink:	if($x_j \geq 0$)	$\mathrm{r} = \mathrm{r} + x_j$	else	$\mathrm{b} = \mathrm{b} - x_j$

where, c_H is the hue value of c, r, y, and b represent *reddish*, *yellowish*, and *bluish*, respectively. After calculating the values of *yellowish*, *bluish*, and *reddish*, we estimate the hue of c_i^{ref} by adding these

(a) Sectional view of ch_t^{in} (b) Sectional view of ch_t^{ref}

Fig. 7. Definition of each variable.

values to the hue of c_i^{univ} as shown below:

$$\begin{array}{lll} \text{if(i = red)} & c_{Hi}^{ref} & = c_{Hi}^{univ} + \text{y} - \text{b} \\ \text{if(i = orange)} & c_{Hi}^{ref} & = c_{Hi}^{univ} + \text{y} - \text{r} \\ \text{if(i = yellow)} & c_{Hi}^{ref} & = c_{Hi}^{univ} + \text{b} - \text{r} \\ \ldots & & \end{array}$$

The algorithm then estimates the shape of the basic convex hull ch_i^{ref} by using Eq. (4),

$$R\left(ch_i^{ref}\right) = R\left(ch_i^{univ}\right) \cap R'\left(ch_i^{in}, c_i^{ref}\right) \tag{4}$$

where, $R(ch)$ is the region of the basic convex hull ch and $R'(ch, c)$ is the shifted region of ch to c. The union with ch_i^{univ} is done in the first equation so that the estimated basic convex hull does not cover another basic color category.

In this way, we can estimate the basic convex hull ch_i^{ref} when pixels in C_i do not exist in the reference image.

4. COLOR TRANSFORMATION

In this section, we describe our color transformation method. In Section 4.1, we explain how the color transformation is done using color features as calculated in the previous section. However, because we do not consider the relationship with neighboring pixel colors on the image, there might be pseudocontours. For avoiding this, we also consider color features in the image space and regenerate the resultant image. This process will be described in Section 4.2.

4.1 Matching Point Calculation and Replacement

In this section, the algorithm is used to find a corresponding color value for every pixel of the input image. For each pixel color value p_i^{in} in the input image, the corresponding color value, p_i^{ref}, is computed as a relative position from the centroid of the basic convex hull. This is given by Eq. (5),

$$\boldsymbol{p_i^{ref}} = \frac{\|\boldsymbol{p_i^{in}} - \boldsymbol{c_i^{in}}\|}{\|\boldsymbol{b_i^{in}} - \boldsymbol{c_i^{in}}\|}\left(\boldsymbol{b_i^{ref}} - \boldsymbol{c_i^{ref}}\right) + \boldsymbol{c_i^{ref}} \tag{5}$$

The definition of each variable is given in Figure 7. b_i^{in} is the intersection point between ch_i^{in} and a vector v whose origin is c_i^{in} and direction is from c_i^{in} to p_i^{in}. b_i^{ref} is the intersection point between ch_i^{ref} and a vector whose starting point is c_i^{ref} and whose direction is the same as that of v.

Next, the algorithm substitutes each pixel color value in the input image with its corresponding color value calculated as described above. In this way, we can produce an image whose color features are similar to those of the reference image.

(a) input image (b) naming result by 3.1 (c) naming result by 4.2

Fig. 8. Restriction on diffusion of naming.

4.2 Consideration of the Color Distribution in the Image Space

Now, we can transfer colors using the methods described in Section 3 and 4.1. However, if an input image has a color-graded region that covers more than one basic color category, then a pseudocontour, may appear in the image. This is because of our explicit color-naming method and because no consideration is given to the relation between neighboring pixels in the image space. For example, even though the color changes gradually in the lower part of Figure 8(a), the process described in Section 3.1 categorizes pixels into two different categories—blue and green.

In this section, the algorithm found positions where the pseudocontour occurs, then performs the fuzzy color naming for those pixels, and redoes the color transformation.

For this, we first define pseudocontours as pixels that satisfy the following three properties: (1) the pixel is located on the boundary of the image region segmented by basic color categorization; (2) the pixel does not form an edge in the input image. That is, the pixel is within the smooth color changes in the input image; (3) the pixel has a large color difference with its neighboring pixels in the output image while it has a small color difference in the input image.

For finding pixels that have the first property, the algorithm merges neighboring pixels having the same color name and generates regions in image space. It then finds the boundary pixels. The second property can be checked by using its color difference in the input image. If the color differences are large in the input image, it will be on an edge and vice versa. Thus, we calculate the smoothness probability of a pixel P using Eq. (6),

$$P(x, y) = \begin{cases} 1 - Sigmoid(f^{in}(x, y)), & (x, y)\ is\ a\ boundary\ pixel \\ 0, & otherwise \end{cases} \tag{6}$$
$$Sigmoid(X) = 1/(1 + e^{-0.3(X-\alpha)})$$

where (x, y) is a position in the image space, $f^{in}(x, y)$ is the Euclidean color difference at (x, y) in the input image, $Sigmoid(X)$ is a sigmoid logistic function and α is a predefined constant value. The sigmoid function is employed for its smooth change of shape and for avoiding those artifacts that may occur by an explicit definition of threshold. The third property can be computed using Eq. (7)

$$Linear(T(x, y), t) = \begin{cases} 0, & T(x, y) \le 0 \\ T(x, y), & 0 < T(x, y) \le t \\ t, & T(x, y) > t \end{cases} \tag{7}$$
$$T(x, y) = f^{out}(x, y) - f^{in}(x, y)$$

where $f^{out}(x, y)$ is the color difference at (x, y) in the output image, and (x, y) is a boundary pixel. If $Linear(T, t)$ is large, it is more likely that the pixel forms a pseudocontour.

Next, the algorithm calculates the extent to which pixel naming should be diffused and performs iterative anisotropic diffusion [Perona and Malik 1990]. Diffusion should be done until the pseudocontour becomes indistinct in the output image. We define the number of diffusion, $N(x, y)$ as shown in Eq. (8)

$$N(x, y) = P(x, y) * Linear(T(x, y), t) \tag{8}$$

The algorithm diffuses pixel naming based on the value $N(x, y)$. For each image pixel, we diffuse the pixel naming based on the following algorithm.

We place a restriction on "*diffuse color naming*" that pixel naming will not be diffused when the color difference is apparent, even within the same region. This is realized by using anisotropic diffusion [Perona and Malik 1990], where the diffusivity depends on the value of $P(x, y)$.

while there exist (x, y) such that $N_d(x, y) > 0$ **do**:
  for each pixel (x, y) in an image **do**:
    if $N_d(x, y) > 0$ **then**
      diffuse color naming at (x, y)
      $N_d(x, y) = N_d(x, y) - 1$
      for each neighboring pixel (x', y') **do**:
        if $N_d(x, y) > N_d(x', y')$ **then**
          $N_d(x', y') = N_d(x, y)$

We show the result of applying this process to the previous input image in Figure 8. Because f^{in} values are large in the upper part of the boundary between the blue and green regions, the color naming results in this part are not diffused, as shown in Figure 8(c). On the other hand, because values of f^{in} in the lower boundary part are small, the color naming results in this part are diffused. Now, pixels near the lower boundary part have probabilities that they belong to both the blue and green categories.

Finally, for each pixel color value $\boldsymbol{p}^{in}$, the algorithm recalculates the corresponding color value using the fuzzy color naming result described above. This is achieved using the weighted sum of corresponding color values in each color category, as shown in Eq. (9)

$$\begin{aligned} \boldsymbol{p}^{ref} &= \sum_i \left(cc'_i * \boldsymbol{p}_i^{ref}\right), \\ cc'_i &= \frac{cc_i}{\sum_i cc_i} \end{aligned} \tag{9}$$

where cc_i is the probability that p^{in} belongs to C_i and p_i^{ref} is the corresponding color value calculated using Eq. (5).

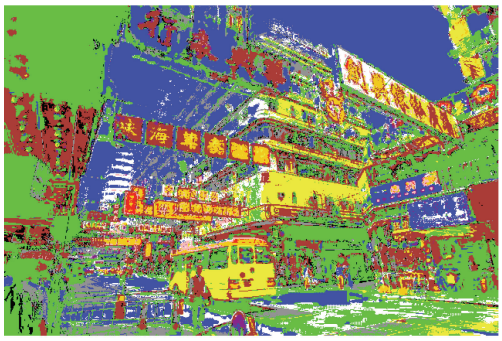
5. RESULTS

In this section, we show that our method is capable of generating a variety of color transformation results.

Figure 9 shows an automatic color-naming result that demonstrates the effectiveness of the color-naming experiment described in Section 2. Figure 9(a) shows an input image and Figure 9(b) shows our prior automatic naming result [Chang et al. 2002, 2003] for comparison. Each pixel is shown in the color corresponding to the automatic naming results. In Figure 9(b), low saturation regions, such as those in the shadows and the sky, are named as chromatic colors. On the other hand, as shown in Figure 9(c), colors with lower saturation are correctly classified into achromatic colors based on our new experimental results. This prevents the algorithm transferring shadows from black to green, the road from gray or white to green, and so on.

Figure 10 shows the algorithm flow visually. Figure 10(a) shows a reference image, while (b) is an input image. The automatic color naming results of Section 3.1 are shown in (c), and the result of the color transformation explained in Section 4.1 is shown in (d). In Figure 10(d), we can see many pseudocontours around the sky, water surface, boat, and deck. This is because we did not consider the neighborhood relationships in the image space and used explicit color segmentation. Therefore, the algorithm checks if there are any pseudocontours, as described in Section 4.2, and then does the fuzzy

(a) input image



(b) by [Chang et al. 2002, 2003]

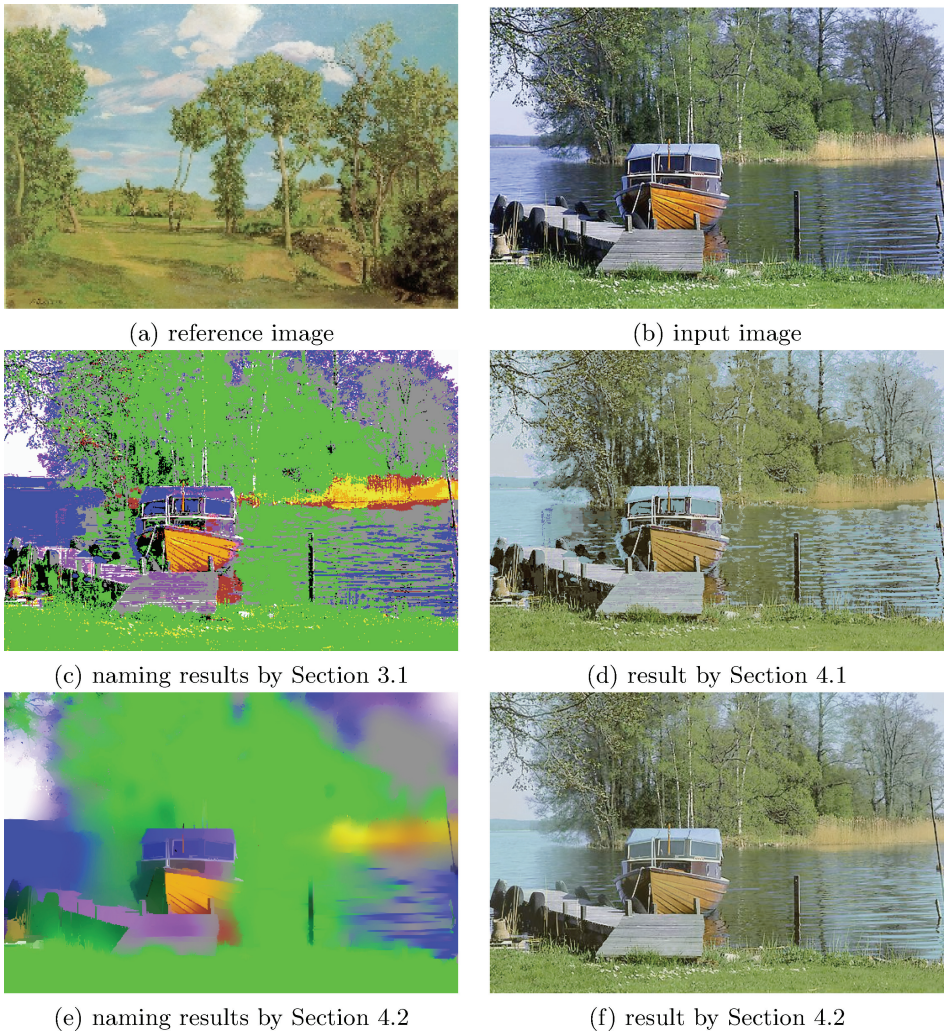
(c) by the method in Section 3.1

Fig. 9. Automatic color naming result.

color naming as shown in (e). Now, the algorithm recalculates the corresponding color value for every pixel value by using fuzzy color naming results, generating the final result shown in (f).

Figure 11 shows various examples of results obtained with our color transformation algorithm. Figures 11(a), (b), and (c) are input images, and Figures 11(d) and (e) are reference images. Figures 11(f–k) show color transformed images. We can see that the colors of the resulting images differ according to the reference images, because our algorithm can capture the color features of a reference image and transfer these to an input image. For example, reference image 1, shown in Figure 11(d), is painted with dark yellow, yellowish orange, light blue, yellowish green, and so on. Our algorithm captures this color use and produces resultant images rendered with similar colors, e.g. dark yellow, light blue, and yellowish green, as shown in Figure 11(f), (h), and (j). Moreover, our color transformation method can preserve the color variety in each color region because our algorithm calculates a matching color value for every pixel, instead of only replacing each segmented area with one color. For example, the grass textures in Figure 11(a) have a variety of yellowish colors. In the resultant images, Figure 11(f) and (g), this color variety is preserved. Furthermore, the resulting images appear natural despite the large color transformations, because each pixel's color remains in the same basic color category.

Figure 12 shows the result of estimation with no direct reference. Figure 12(a) is the input image, Figure 12(b) is the reference image, and Figure 12(c) shows the result. Note that the input image has pixels from the blue category, while the reference has not. The method described in Section 3.3 analyzes other existing categories of the reference image and calculates where to transfer blue pixels of the input image. In this example, our method analyzes the reference image to be relatively yellowish and then shifts the pixels in the blue category in the yellowish direction.



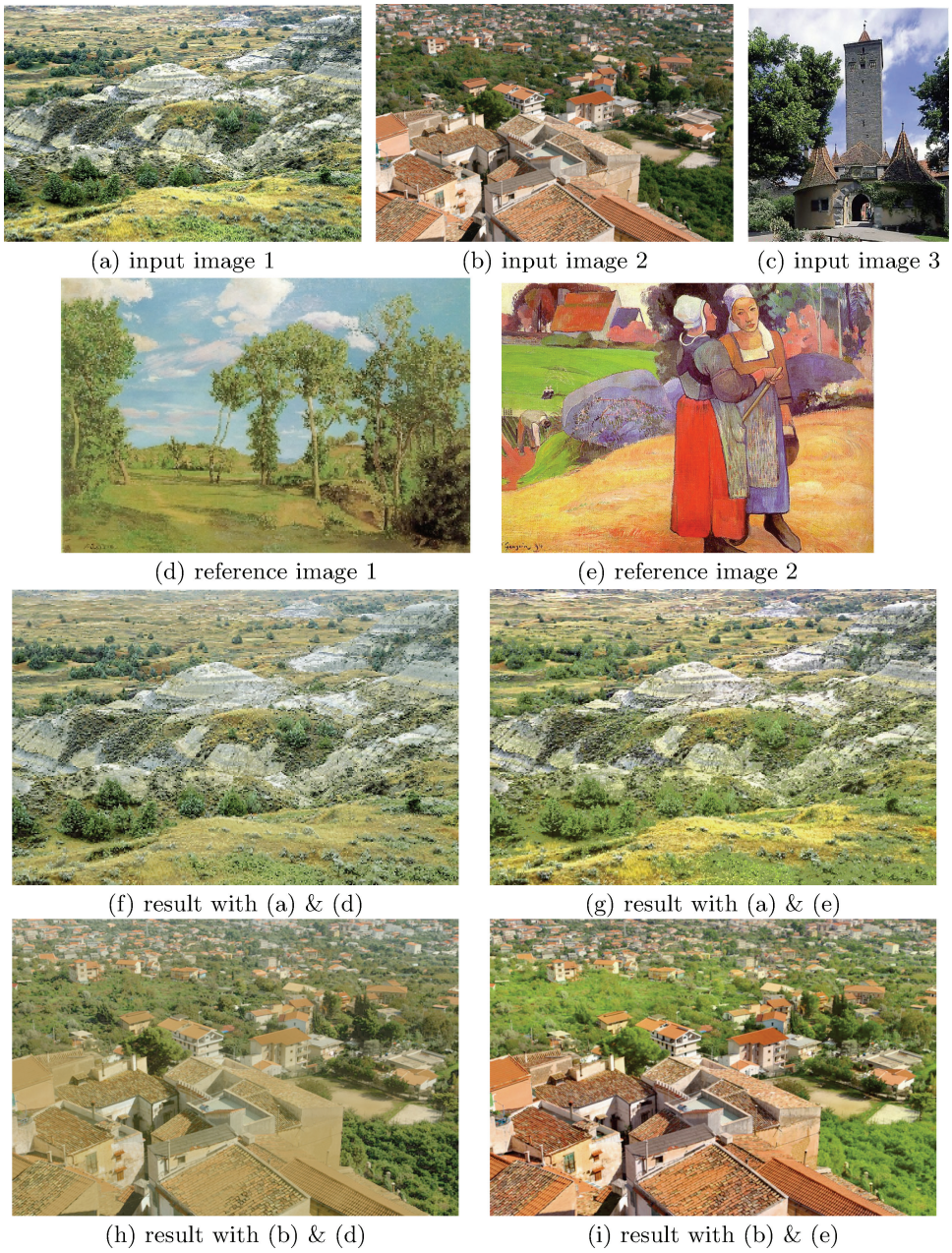
(a) reference image (b) input image

(c) naming results by Section 3.1 (d) result by Section 4.1

(e) naming results by Section 4.2 (f) result by Section 4.2

Fig. 10. Visual flow of our algorithm.

Figure 13 compares our color transformation result with results from previous works. Figure 13(a) is an input image, 13(b) is a reference image, and Figure 13(c) shows the resultant image by our method. In our approach, each pixel color shifts in a different direction according to its basic color category, so the resultant image does not cause any unnatural feelings. Moreover, because we also consider the relationship in the image space, pseudo-outliers do not appear in the resulting image. Figure 13(d) shows our previous color transformation result [Chang et al. 2002, 2003]. In our previous work, we did not consider the relationship in the image space, so the colors in the background, which are defocused and blurred, are transferred discontinuously. This results in pseudo outliers. An even worse problem is that some parts of the hair and webbing strap, which have very low luminance, are named green and transformed to green. Figure 13(e) and (f) show the color transformation results from previous work [Reinhard et al. 2001]. Figure 13(e) is the result acquired only by the automatic process, while 13(f) is the result generated with user interaction. Figure 13(e) seems as if it is a result



(a) input image 1 (b) input image 2 (c) input image 3

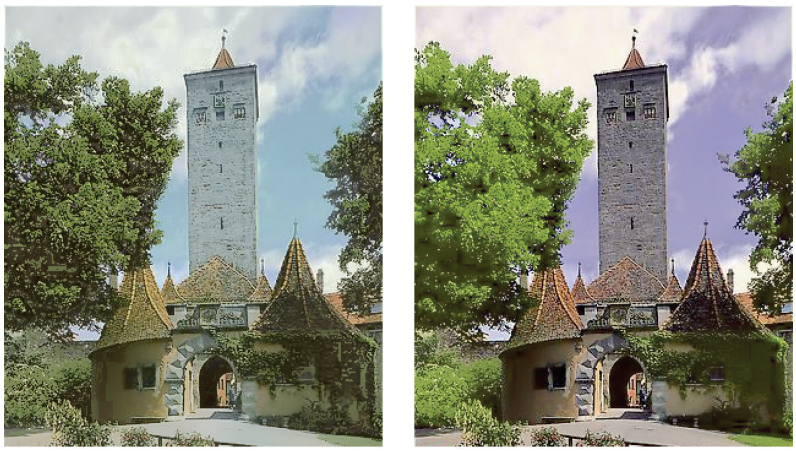
(d) reference image 1 (e) reference image 2

(f) result with (a) & (d) (g) result with (a) & (e)

(h) result with (b) & (d) (i) result with (b) & (e)

Fig. 11. Experimental results.

of filtering the input image by a yellowish filter. This is because the mean color of the reference image is yellowish. For this reason, the color spreads of the input image are moved to a yellowish region. For example, the textures of the woman's clothes are transferred from a reddish color to a yellowish one. Although it does not give such an unnatural feeling, because, in this example, it was just textures



(j) result with (c) & (d) (k) result with (c) & (e)

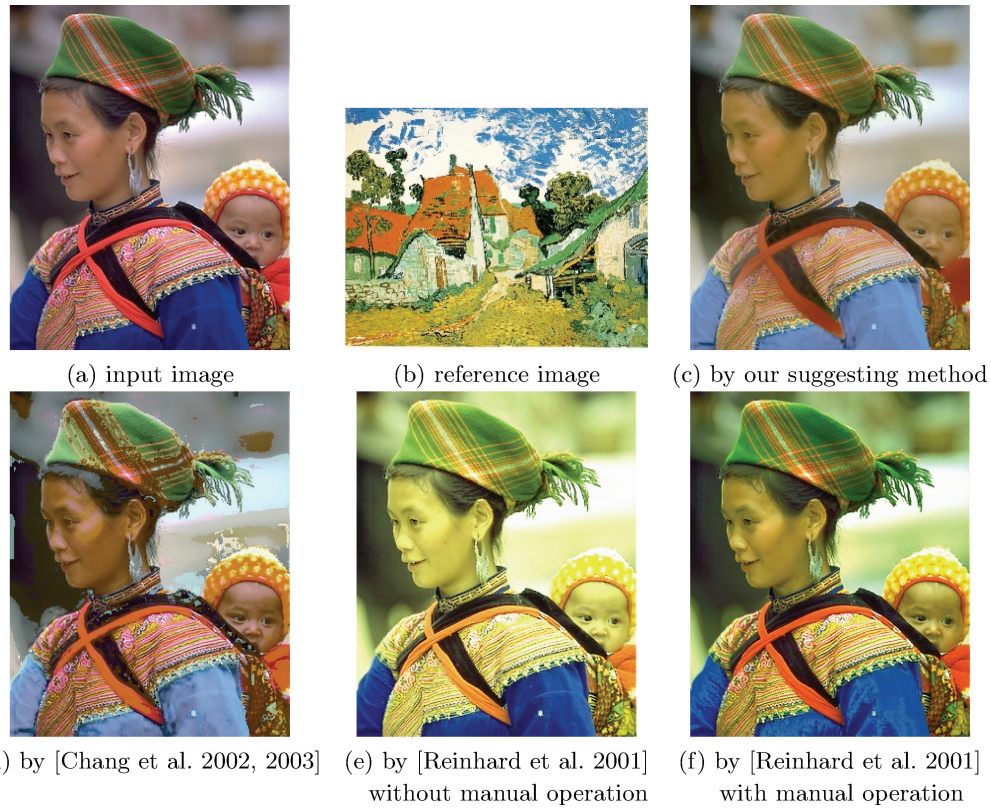
Fig. 11. Experimental results (Cont.).

(a) input image (b) reference image

(c) result image

Fig. 12. Result of estimating colors with no direct reference. In Figure 12(c), pixel colors in blue category as estimated.

if it were a color region whose colors are perceptually meaningful, e.g., red apples or strawberries, this kind of color transformation would cause unnatural feelings. Figure 13(f) is an example of the results generated by defining swatches manually. In 13(f), we show the most reasonable result among a number of our trials of swatch selection from the viewpoint that it can generate a natural result with



(a) input image (b) reference image (c) by our suggesting method

(d) by [Chang et al. 2002, 2003] (e) by [Reinhard et al. 2001] without manual operation (f) by [Reinhard et al. 2001] with manual operation

Fig. 13. Comparison with previous works.

fewer swatches. We set three swatches for the input image [Figure 13(a)]. The first swatch contains the blue cloth, the second is composed of high saturated reddish colors, such as textures of the woman's clothing, and the third contains low-saturated colors, such as those of the background and faces. By using those swatches, we can avoid the unnatural result like that in Figure 13(e) but we perform many trial-and-error adjustments to appropriately select corresponding swatches on the reference image so that the color variance is adequate for each corresponding swatch, the color choice did not cause a problem with the previous selection, and the colors in each region of the resultant image seems natural.

Figure 14 shows the result of applying our algorithm to the example-based NPR (nonphotorealistic rendering) method. Many studies on NPR [Gooch and Gooch 2001] have investigated methods for recreating brush strokes [Haeberli 1990; Meier 1996; Saito and Nakajima 1996; Litwinowicz 1997; Hertzmann 1998] and realistically simulating paints and materials [Curtis et al. 1997; Mario C. Sousa 1999, 2000]. In the real world, there are a variety of styles that artists can use and the chosen styles depend on personal preferences and each situation. For achieving a variety of NPR styles, a method for example-based NPR was proposed [Hertzmann et al. 2001]. This method is based on feature-matching between images and transfers one image's features to another image. This works very well for a wide variety of images, but uses luminance values as a feature vector in the feature matching process. This is because of the "curse of dimensionality" problem. As the dimensionality of the feature vector space increases, the number of training samples needed grows exponentially. Therefore, rather than using color value as features, they use grayscale values.

(a) input image (b) reference image

(c) result of [Hertzmann et al. 2001] (d) texture+color transformation

Fig. 14. Result of adding the effect of brush strokes.

Since color use is also one of the most important features in painting, we combine our example-based color transformation method with the example-based brush stroke effects of Hertzmann et al. As training data, we use the reference image in Figure 14(b) and its blurred image. Figure 14(c) shows the result of touch-texture transformations only, which were done using open source code [Analogies-Web-Page]. Figure 14(d) shows the result of brush stroke effects applied after the transformation of colors. As shown, the combination of our method and theirs can produce very attractive results.

6. CONCLUSION AND FUTURE WORK

In this paper, we have developed a perception based image processing method that enables automatic example-based color transformation.

In our psychophysical experiment, we achieve a categorical color distribution in color space, which takes into account the effects of surrounding colors. The achromatic color categories are expanded compared to a conventional color-naming experiment. Therefore, in digital color imaging, the obtained color classification data is more suitable for naming pixel colors that have low saturation as white, gray, or black.

Our color-transformation algorithm enables straightforward operation; the user just has to input one reference image. This simplicity is achieved by combining perceptual restriction in color space and an evaluation of color differences in the image space.

We expect this work to have a major impact on the digital imaging field. Beyond its clear applications in color transformation, the perceptual categories describe in this paper can be usefully applied in a variety of other areas in digital image indexing where color feature description is important.

There are many promising areas for future research. First, we would like to extend our algorithm to video-recoloring. In this case, we must consider the temporal coherency to obtain flicker-free results. For this purpose, we are planning to extend our algorithm in two ways. First, we are going to apply anisotropic diffusion along the temporal axis. Next, we are going to generate one global basic convex hull using all the pixel values within a shot, instead of generating basic convex hulls for each frame. This will allow us to manage the temporal coherency. Second, the present work assumes that the effect of surrounding color is static throughout the color naming. However, in most cases, the effects of surrounding colors would be more complicated. Color might be named using different basic color terms when the surrounding color varies significanly. We will incorporate these surrounding color effects to allow more perceptually correct color naming. Third, we will investigate a computational model of human color vision. Our current system retrieves color values used in the reference image and applies them to the input image. However, the impression of a color not only depends on each color value, but also depends on the color arrangement and color ratio. If these properties of an input image differ from those of a reference image, the impression of the resultant image may not be the same as the user expects. We plan to investigate ways to handle these properties in the near future. We also would like to integrate perceptual color difference metrics into our method. When our current system performs anisotropic diffusion, it uses the Euclidean color difference metric in the input image as the key factor for diffusivity. However, diffusivity also depends on the image frequency. In addition, we would like to take into account the image frequency and chromatic color sensitivity function for our process.

ACKNOWLEDGMENT

The input images of Figures 1, 10(b), 11(a–c), and 13(a) are from Terra Galleria [Terra–Galleria], reference images are from the Artchive [Artchive].

REFERENCES

ANALOGIES-WEB-PAGE. *http://mrl.nyu.edu/projects/image-analogies/*.

ARTCHIVE. *http://www.artchive.com/*.

BERLIN, B. AND KAY, P. 1969. *Basic Color Terms: Their Universality and Evolution*. University of California Press, Berkeley, CA.

CHANG, Y., SAITO, S., AND NAKAJIMA, M. 2002. Color transformation based on the basic color categories of a painting. In *ACM SIGGRAPH Conference Abstracts and Applications*. 157.

CHANG, Y., SAITO, S., AND NAKAJIMA, M. 2003. A framework for transfer colors based on the basic color categories. In *Computer Graphics International*. 176–183.

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. 1997. Computer-generated watercolor. *ACM SIGGRAPH*, 421–430.

FAIRCHILD, M. D. 1997. *Color Appearance Models*. Addison-Wesley, Reading, MA.

GOOCH, B. AND GOOCH, A. 2001. *Non-Photorealistic Rendering*. A. K. Peters.

HAEBERLI, P. 1990. Paint by numbers: abstract image representation. *ACM SIGGRAPH*, 207–214.

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *ACM SIGGRAPH*. 453–460.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *ACM SIGGRAPH*. 327–340.

LITWINOWICZ, P. 1997. Processing images and video for an impressionist effect. In *ACM SIGGRAPH*. 407–414.

MARIO, C. AND SOUSA, J. W. B. 1999. Computer-generated graphite pencil rendering of 3d polygonal models. In *Computer Graphics Forum*. 195–208.

MARIO, C. AND SOUSA, J. W. B. 2000. Observational models of graphite pencil materials. In *Computer Graphics Forum*. 27–49.

MEIER, B. J. 1996. Painterly rendering for animation. In *ACM SIGGRAPH. 30*, 477–484.

PERONA, P. AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence. 12*, 629–639.

REINHARD, E., ASHIKHMIN, M., GOOCH, B., AND SHIRLEY, P. 2001. Color transfer between images. In *IEEE Computer Graphics and Applications. 21*, 34–41.

ROSCH HEIDER, E. 1972. Universals in color naming and memory. In *Journal of Experimental Psychology. 93*, 10–20.

SAITO, S. AND NAKAJIMA, M. 1996. Automatic production of hand-painted images. In *The Journal of the Institute of Television Engineers of Japan. 50*, 1528–1535.

TERRA-GALLERIA. *http://www.terragalleria.com/*.

UCHIKAWA, K. AND BOYNTON, R. M. 1987. Categorical color perception of japanese observers: Comparison with that of americans. In *Vision Research. 27*, 1825–1833.

UCHIKAWA, K., KURIKI, I., AND SHINODA, H. 1994. Expression of color appearance in aperture and surface color modes with a category rating estimation method. In *Journal of the Illuminating Engineering Institute of Japan. 78*, 41–51.

WELSH, T., ASHIKHMIN, M., AND MUELLER, K. 2002. Transferring color to greyscale images. In *ACM SIGGRAPH. 21*, 277–280.

Received December 2004; revised April 2005; accepted April 2005