Color Sketch Generation

Fang Wen   Qing Luan[†][*]   Lin Liang   Ying-Qing Xu   Heung-Yeung Shum
Microsoft Research Asia   [†]University of Science and Technology of China

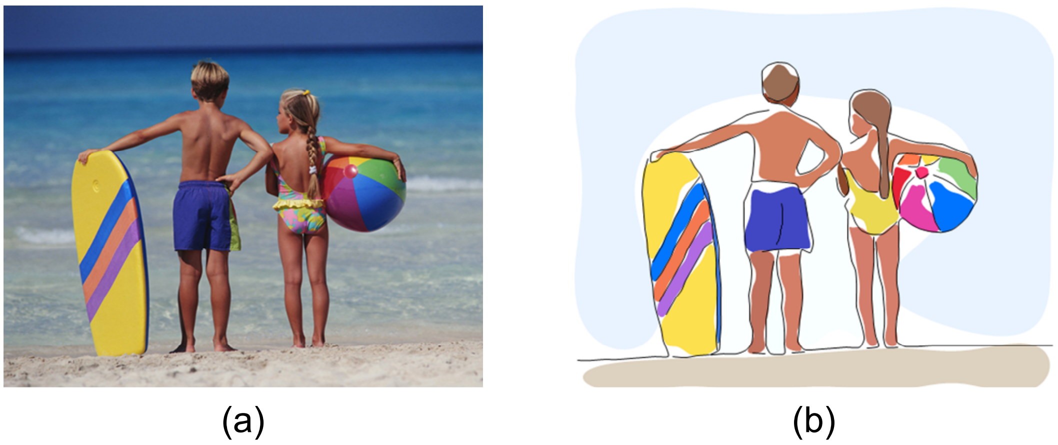

Figure 1: (a) Input image (b) Generated color sketch

Abstract

In this paper, we introduce an NPR system to generate color sketches in a free-hand drawing style. From observation, we found out that in a free-hand style both shape and color of image regions are stylized, which contributes cooperatively to the visual effect of a sketch. With the color regions obtained by our interactive segmentation algorithm, we create a stylized color sketch in two steps. First, in order to manifest the lighting and shading of each region, we shrink the boundary of the color regions with a luminance-based algorithm emphasizing the highlight parts, which enforces consistency with the free-hand artist style of the sketch. Second, the proposed color shift algorithm is applied to image regions to emphasize the main content of the sketch and acquires a visually pleasing combination of color. We choose the optimal combination by minimizing the energy function based on an artist drawn color database, artistic drawing rules and input image colors. The generated results show that our system can produce aesthetically pleasing sketches.

Keywords: non-photorealistic rendering, boundary shrinking, color shift

1 Introduction

Image abstraction is an interesting and long-standing problem in the field of non-photorealistic rendering (NPR), in that it provides a simple and pleasant way to represent the image while expressing the artist's understanding of the scene. Color sketch is a drawing style that attempts to represent the objects in an abstract way, which frequently appears in graphics design, cartoon drawing, advertisement design, landscape, and art galleries. Figure 2 shows a few examples of the color sketches drawn by an artist. By analyzing the techniques of artists' work, we observed two phenomena that is interesting for further reflection. First, regions in the image are not completely filled with color, which not only gives a pleasant look but also reflects the global lighting of the object. Second, the artist adjusts original colors to emphasize the main content of the image, as well as to maintain a color combination that is harmonious and beautiful. How to imitate the artist in creating a color sketch is an interesting topic in NPR. In this paper, we present an easy-to-use system to enable a non-professional user to abstract images like a professional artist.

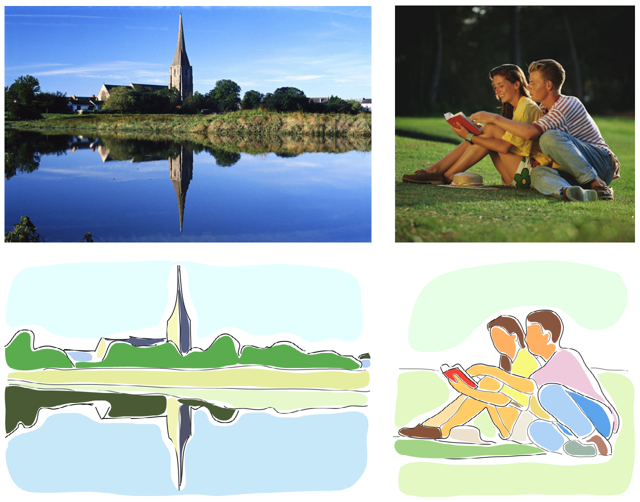

Figure 2: Two examples of color sketches drawn by an artist. The first row is original images. The second row is color sketches drawn by an artist.

[*]This work was done while Qing Luan was a visiting student at Microsoft Research Asia

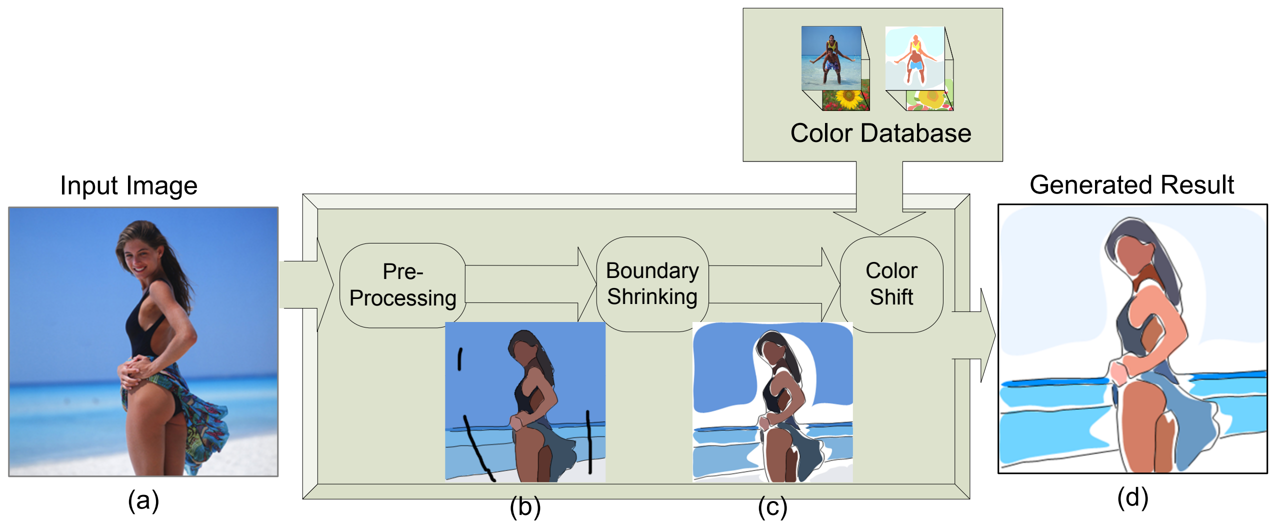

Figure 3: System Overview. (a) is the input image. (b) is the result after pre-processing stage. The image is divided into several regions with vectorized outlines. The black lines in the result represent that the regions which black lines pass through are background regions. (c) is the result after boundary shrinking stage. (d) is the generated color sketch.

2 Related Work

Some existing image stylization techniques, such as painting with brush strokes [Hertzmann 1998][Hertzmann 2001] [Hertzmann et al. 2001] [Shiraishi and Yamaguchi 2000] [Healey et al. 2004] mainly focus on simulating the effects of certain drawing tools (say, oil painting, watercolor, pencil sketches and so on), without taking into consideration the content of the picture, while proper artistic stylization requires some understanding of the main content in the picture that need to be emphasized.

There exists a previous work on improving the areas of emphasis in sketch rendering. DeCarlo et al.[DeCarlo and Santella. 2002] transform an image into a line drawing with bold edges and large regions of constant color. They derive the user areas of interest using eye-tracking and emphasize them by adjusting their resolution, i.e., in areas of emphasis drawing regions with higher resolution while in other areas drawing regions with lower resolution. Our work is different in that the main content of the picture is emphasized by adjusting colors.

There are also some works on improving the color of images. However, most of these works relate to transferring colors to monochrome images, doing the inverse to convert a color image into gray [Welsh et al. 2002][Levin et al. 2004][Gooch et al. 2005], or transferring color according to a reference image[Reinhard et al. 2001].

3 Our Algorithm

3.1 System Overview

The system framework is shown in Figure 3. Given an input image, in the pre-processing stage we do interactive segmentation and outline drawing to get the abstract of the image. The image thus becomes a set of regions $\{\Omega_i\}$ with vectorized outlines. In addition, the user needs to manually specify which regions are background/foreground. Then, a two-stage stylization algorithm is used to simulate properties of the artists' free-hand drawing style that are described in the previous section.

In the first stage, we use a boundary shrinking algorithm to shrink and smooth the boundary of each region. The blank areas are based on the luminance distribution in the region and make the picture look like a freehand painting.

In the second stage, we do the color shift. Rather than simply selecting the most visually similar color from the database or giving a global transformation to the background/foreground region colors, we formulate the color shift as an optimization problem and define an energy function on candidate colors of all regions, trying to integrate information from a pre-drawn color pairs database, artist drawing rules and reference information from the input image to obtain a globally optimal color shift solution.

In the next subsections, we will describe the algorithm in detail.

3.2 Interactive Segmentation

Image segmentation has long been a hot topic in computer vision and graphics. However, it is still an open problem in today. It is hard to obtain a satisfactory result with any automatic segmentation algorithm for our application. In recent years, several interactive segmentation techniques, such as [Li et al. 2004] and [Barrett and Cheney 2002], were developed for object selection. Borrowing some ideas from these works, in this paper we do not contribute to segmentation itself, but focus on how to provide a friendly user interface (UI) to help the user to refine the automatically segmented result. Firstly we use the mean-shift algorithm [Comaniciu and Meer 2002] to obtain a rough segmentation, then our UI can help the user to refine the segmentation result easily.

3.2.1 UI design

Merging regions and re-segmenting of the merged region

In general, rough segmentation will generate some tiny regions that we need to manually merge into a large semantic region. Figure 4 shows how a user merges the tiny regions by simply drawing a stroke. On the other hand, the segmentation with the mean-shift algorithm can be redone on the individual merged regions by using new parameters if it is necessary. This design allows a user to choose different parameters for different parts of the image.

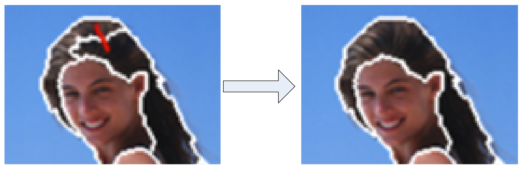

Figure 4: Merging regions. The red stroke is drawn by user. White lines represent boundaries of regions.

Editing region boundary

Inspired by the Lazy Snapping system [Li et al. 2004], an overriding brush with a variable width is used to edit the boundary of a region, as shown in Figure 5. It is similar to the Lazy Snapping system in that the user does not need to give an exact boundary. When a user brushes a stroke, the brush with a width of several pixels gives a small band and a pixel-based graph cut segmentation algorithm is used in this band to obtain an optimal stroke, which starts and stops at two points on the original boundary. These two points split the original boundary into two parts. The part that is closer to the stroke is replaced by the user stroke to generate a new boundary of the region. For every user stroke, the width can be adjustable. By default we set the width to 7 pixels. Of course, in the worst case, the user can set the width of the brush to 1 pixel to give an exact boundary. After editing the region boundary, one or more subregions that previously belong to the region would be separated. These subregions are automatically merged into their adjacent regions according to the color similarity.

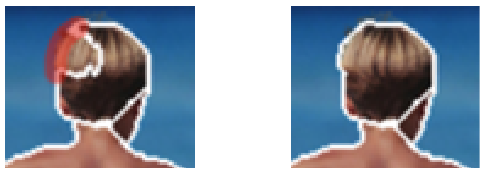

Figure 5: Editing region boundary with overriding brush. White lines represent boundaries of regions. (a)Result before boundary editing. (b)Result after boundary editing.

Splitting region

We use a brush to split a region into two regions, as shown in Figure 6. Two types of user interaction with the original boundary split the region into two parts. If the stroke starts and stops at two points on the original boundary, which split the original boundary into two parts, every part connecting with the user stroke becomes the boundary of a new region. If the stroke forms a closed curve and has no interaction with the original boundary, the inner area of the closed curve become a new region. Similar with region editing, the user also does not need to give an exact boundary. An optimal stroke is determined with graph cut segmentation in a small band around the input stroke.

3.3 Outline Drawing and Background Masking

With the regions drawn, we superimpose their outlines. We trace boundaries of regions, down-sample and fit boundary points to Cardinal splines. Also, we allow the user to manually add some lines that represent strong edges in the region, delete existing ones, merge two splines into single one or vice versa.

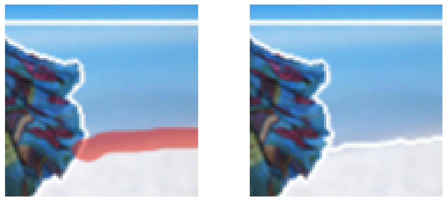

Figure 6: Split a region into two regions. White lines represent boundaries of regions. (a)Result before splitting. (b)Result after splitting.

After segmentation and outline drawing, the image becomes a set of regions with vectorized outlines. The user can manually point out the backgrounds by drawing only a few lines; all the regions that the drawn lines pass through will be considered as backgrounds.

3.4 Boundary Shrinking

From artist works in Figure 2, we observed that the blank areas are not randomly drawn but correspond to the brightness of a region. So the key is how to establish the relationship between shrinking and luminance distribution in the region. In general, shrinking an entire region around its center of mass (with pixels weighted inversely to intensity) is not a perfect solution because it may cause shrinking along the dark edges that we do not want. Here we propose the following shrinking algorithm, and our experimental results show that this method works well.

Firstly, after tracing the region boundary, we smooth the boundary using low pass FFT filtering and down-sample the boundary points to get a list of control points, with a down-sample rate of $1/K$ ($K = 10$ in our implementation).

Secondly, we independently adjust the position of control points to shrink the region, based on the luminance distribution along its normal direction. Here we refer to the luminance as *intensity*. The algorithm is to compute the ”center of intensity” along a chord through the region, then shift the boundary point inward so that the geometric center of the new chord shifts toward the center of intensity. The adjustment strategy of a control point is as illustrated in Figure 7.

a) For control point P_0, take the closest opposite boundary point P_1 along its normal direction.

b) Uniformly sample N points along the segment $\overrightarrow{P_0P_1}$, take their *intensities* and normalize them into $[0,1]$, where N is proportional to the length of $\overrightarrow{P_0P_1}$, i.e., $N \propto |\overrightarrow{P_0P_1}|$. Denote the 2D coordinate of the k^{th} point as X_k, and denote its normalized intensities as $\{I_k\}$, $1 \le k \le N$.

c) Set weights $W_k = 1 - I_k$ for the k^{th} sample point, $1 \le k \le N$.

d) Calculate the coordinate of the weighted center point P_w and the center point P_c:

$$X_w = (\sum_{k=1}^{N} W_k * X_k)/\sum_{k=1}^{N} W_k$$

$$X_c = (\sum_{k=1}^{N} X_k)/N.$$

e) Adjust the position of P_0 according to P_w and P_c. If $|\overrightarrow{P_0P_w}| > |\overrightarrow{P_0P_c}|$, that means more darker points are closer to P_1, so adjust P_0

to P' along its normal, $|\overrightarrow{P_0P'}| = |\overrightarrow{P_cP_w}|$. Otherwise, do not adjust the position of P_0.

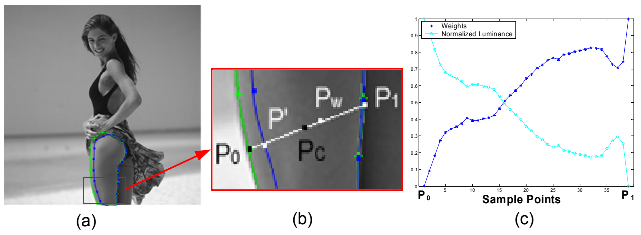

Figure 7: Boundary Shrinking. (a)Intensity image. Blue line is the original boundary and green line is the shrunk boundary. (b)Adjustment of control point P_0. (c)Normalized *intensity* and weights of sample points.

Thirdly, we look for self-crossings in the new boundary path to detect over-shrunk control points, then remove over-shrunk control points and connect remaining control points in order, as shown in Figure 8. Finally we fit cubic B-splines using all these remaining control points to obtain a smoothed and shrunk boundary.

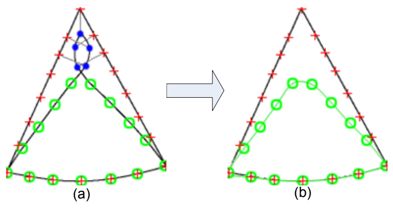

Figure 8: Processing for Over-shrinking. (a) Red crosses represent original control points. Blue points represent over-shrunk control points. Green circles represent remaining control points. (b) Green line is the final shrunk boundary.

3.5 Color Shift

Here we mainly process colors in the Munsell Renotation System, which is organized in terms of three perceptual attributes of color: hue, chroma, and value. To simplify notation we will use bold letters ($\mathbf{O}, \mathbf{P}, \mathbf{R}$...) to denote a color triple (*hue*, *chroma*, *value*) and use $H(.)$, $C(.)$ and $V(.)$ to take the *hue*, *chroma* and *value* component of the color respectively. We adopted this color space because of its uniform and orthogonal properties for each perceptual attribute. The conversion between the Munsell system and *CIE XYZ* color space can be found in [Wyszecki and Stiles 1982].

In the color shift stage, we desire to maintain the color hue of the input image but learn how artists use chroma and brightness to deemphasize the background and emphasize the foreground. For this purpose, we choose 46 images with 85 background color regions and 861 foreground color regions to compose the background and foreground color database. For each image, we sketch the regions, as shown in Figure 9(b), calculate average colors for every region, as shown in Figure 9(c), and ask an artist to draw a color sketch, requiring that the drawn color has similar hue with the original color and maintain the harmony of the whole image, as shown in Figure 9(d). For every region Ω_k, the average color in the photo, denoted as $\mathbf{O}_k$, and the painting color in the color sketch, denoted as $\mathbf{P}_k$, forms a color pair. Finally color pairs are put into background and foreground color database respectively.

There are a few different methods that can be employed to handle the color shift problem with the help of a pre-defined color database. For example, a straightforward idea is to learn a global transformation from the background and foreground color database respectively and apply it to the image to be processed. However, this global transformation function is hard to learn from our pre-drawn database, especially for background color database, as shown in Figure 10.

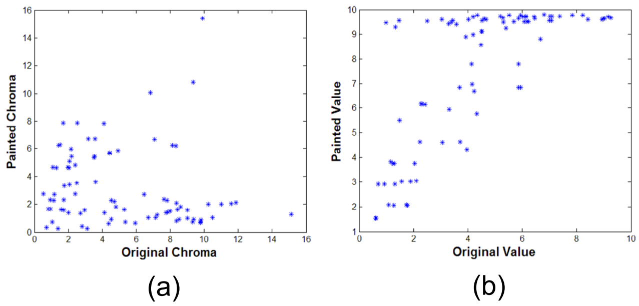

Figure 10: Original and painting Chroma (a) and Value (b) of all regions in background color database. X-axis represents original color and Y-axis represents painted color

Another idea is to directly use the artist's experience to guide the color selection for each region. It means that we can select a 'best' color for every region according to the similarity of the region color and the original color in our database. However, for color sketch generation the selection of the colors depends on not only the original color but also other properties of the region, such as the size of the area, perimeter etc., or even properties of adjacent regions. Therefore, there are always some regions that look unusual in most cases. Figure 11 shows a comparison of this approach (b) and our color shift approach (c). We can see that some regions look unharmonious in Figure 11(b).

The advantage of the first method is global balance of the color selections, and the benefit of the second method is local consideration of the color selection of individual regions. In this paper, we employ a global optimization to produce a convincing color sketch by combining the above two methods. A pre-defined color database can provide candidate colors for the optimization process. Then, an energy function can combine various guidelines for the color selection including the global transformation tendency and some special rules for this particular NPR style. Through the energy minimization process, we can quantitatively evaluate the combination of candidate colors of all regions by integrating artistic hints and information from the color database and the input image, and a globally optimal group of colors can be obtained. Our color shift approach is described in the following subsection.

3.5.1 Our Approach

1) Select candidate colors

Given an image composed of M regions, for the region Ω_i with average color $\mathbf{R}_i$, $i = 1,2,...,M$, select N candidate colors $\mathbf{P}_k$ with label $x_i = k$, $1 \le k \le N$ from the background and foreground color database, respectively. The corresponding original color in the color database $\mathbf{O}_k$, $k = 1,2,...,N$ is also shown in Figure 12. The selection criterion is $H(\mathbf{P}_k) \approx H(\mathbf{R}_i)$ and the distance of $\mathbf{R}_i$ and $\mathbf{O}_k$ in the $C-V$ plane is as small as possible. Actually, in order to guarantee that the colors in the database are enough to cover the full range of hue, the "$H(\mathbf{P}_k) \approx H(\mathbf{R}_i)$" is set loosely in our implementation. The range of hue is [0, 100] in the Munsell color space and we require that the difference of $H(\mathbf{P}_k)$ and $H(\mathbf{R}_i)$ be less than 5.

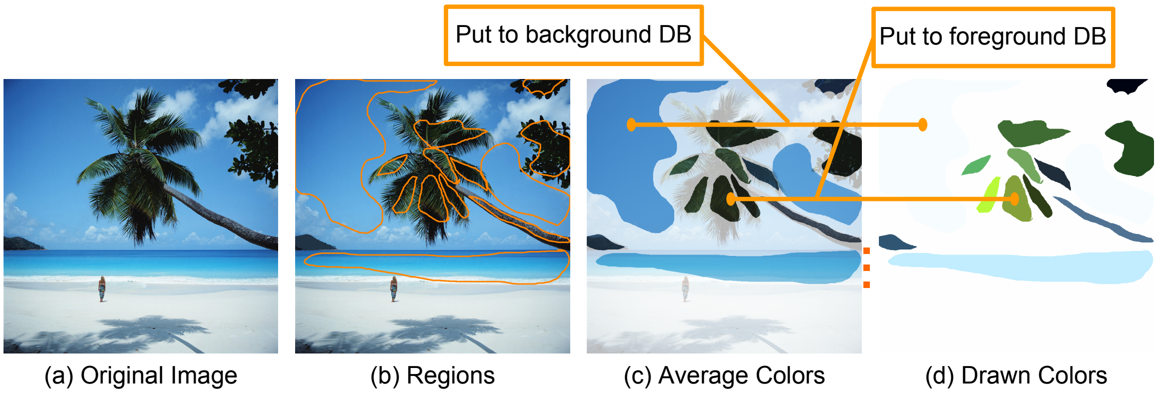

Figure 9: Example of color database construction

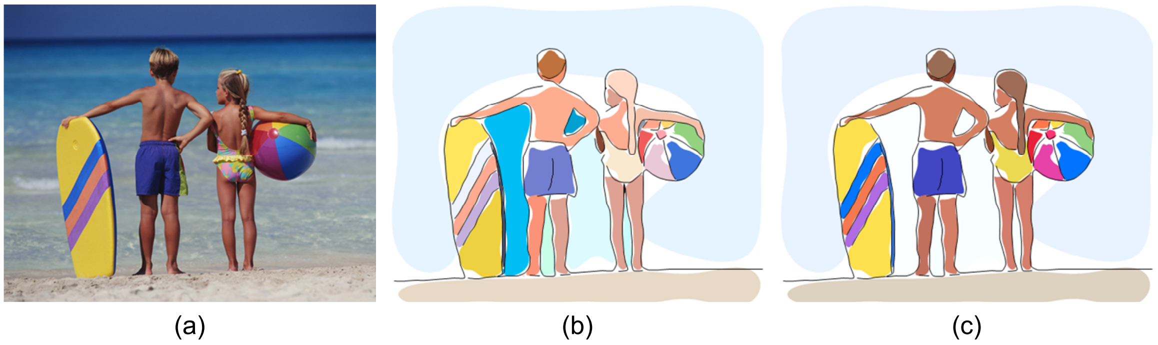

Figure 11: Comparison of our color shift approach and directly selecting the 'best' color according to the color similarity. (a)Input image. (b)Directly selecting the 'best' color. (c)Our approach.

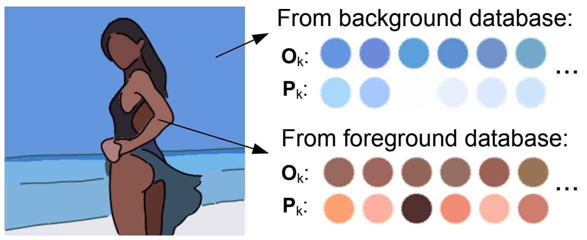

Figure 12: Selection of candidate colors

2) Build graph

Next we build a graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$ from the input image. Vertices $\mathscr{V}$ represent regions, and edges $\mathscr{E}$ denote links between adjacent regions and regions with similar hue. We eliminate the links between the background region and the foreground region. Thus,
$\mathscr{V} = \{\mathscr{V}_b, \mathscr{V}_f\}$ and $\mathscr{E} = \{\mathscr{E}_b, \mathscr{E}_f\}$.
$\mathscr{V}_b/\mathscr{V}_f$ represents the background/foreground regions. $\mathscr{E}_b/\mathscr{E}_f$ represents the edges connecting adjacent background/foreground regions or edges connecting background/foreground regions with similar hue but are not adjacent.

3) Set energy function

Set energy function on $\mathscr{G}$:

$$\begin{aligned} E(X) &= \sum_{i\in\mathscr{V}} E_1(x_i) + \sum_{(i,j)\in\mathscr{E}} E_2(x_i,x_j) \\ &= \sum_{i\in\mathscr{V}_b} E_{1b}(x_i) + \sum_{(i,j)\in\mathscr{E}_b} E_{2b}(x_i,x_j) \\ &\quad + \sum_{i\in\mathscr{V}_f} E_{1f}(x_i) + \sum_{(i,j)\in\mathscr{E}_f} E_{2f}(x_i,x_j) \qquad (1) \end{aligned}$$

This is the key step of the whole approach. The term E_1 represents the local cost of the candidate color of a region, and the term E_2 represents the compatibility cost of candidate colors of two regions. We will explain the energy function definition in an individual subsection below.

4) Minimize energy function

Minimize the energy function to obtain the optimal candidate color $\mathbf{P}_i^*$ for every region i. We adopted the loopy belief propagation algorithm (BP) to minimize the energy function $E(X)$. As a kind of probability inference algorithm proposed by Pearl [Pearl 1988], the BP algorithm has been widely used in machine learning and computer vision [Weiss 1996][Freeman et al. 2000]. BP is a local message passing algorithm that can minimize the Gibbs energy defined on any pairwise undirected graph, e.g., our energy $E(X)$ in Equation (1). The basic mechanism of belief propagation is for each node in a graph to receive messages from its neighbors, then to send updated messages back to each of them. Although it does not guarantee a global minimum for a graph with multiple loops, recent empirical results on several vision problems [Freeman et al. 2000] show that BP is often a good approximation even for a graph with thousands of loops.

5) Shift color

Shift color of region Ω_i from $\mathbf{R}_i$ to $\mathbf{R}_i^{'} = (H(\mathbf{R}_i), C(\mathbf{P}_i^*), V(\mathbf{P}_i^*))$,

$i = 1, 2, \ldots, M$. Thus, the final result is obtained.

3.5.2 Energy Function

There are several guidelines for integrating the information from the color database, input image and artist drawing rules into the energy definition.

For background regions:

b1) The painting color is dependent on not only the original color, but also the size and shape complexity of the region. The more similar the original color, the size and shape complexity of the region, the more similar the painting color.

Here we define two parameters to represent the size and the shape complexity of the region:
a) *Relative area*, i.e., the ratio between the region area and the image area, represents the size of the region. We denote the *relative area* of the region Ω_i as $S(\Omega_i)$, and denote that of the region which corresponds to the candidate color $\mathbf{P}_k$ as S_k.
b) *Relative perimeter*, i.e., the ratio between the perimeter and the region area, represents the shape complexity. We denote the *relative perimeter* of the region Ω_i as $P(\Omega_i)$, and denote that of the region which corresponds to the candidate color $\mathbf{P}_k$ as P_k.

b2) An *isolated* background region with a small size or complex boundary shape tends to be painted with a color close to white. Here we refer to the 'isolated' background region as a background region whose adjacent regions are foreground regions. In Figure 13 we show some examples of *isolated* and *non-isolated* background regions.

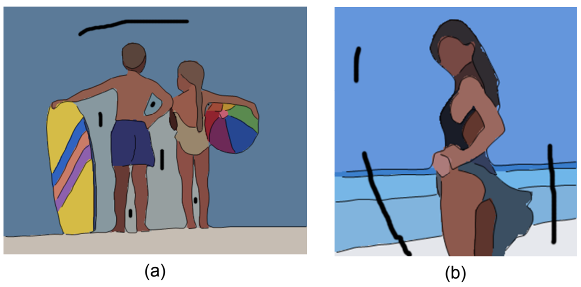

Figure 13: Examples of *isolated* and *non-isolated* background regions. As described before, all regions that a black stroke passes through are background regions. In (a) all background regions are *isolated* while in (b) all background regions are *non-isolated*.

b3) If two background regions are not adjacent but have similar color hue, we tend to paint them with the same color. We call this kind of region a *homogeneous* background region. In Figure 14 we show some examples of *homogeneous* background regions. Actually, there exist four pairs of *homogeneous* background regions in this image. Every background region to the left of the girl has a *homogeneous* one to her right. For clarity, we only point out one pair in the figure.

b4) Tend to keep the original intensity contrast between adjacent regions. We denote the average intensity of the region Ω_i as $I(\Omega_i)$, and denote that of the region which corresponds to the candidate color $\mathbf{P}_k$ as I_k.

For foreground regions:

f1) Tend to keep original chroma and value contrast between adjacent and *homogeneous* foreground regions.

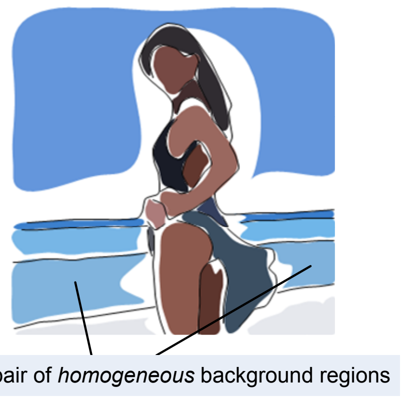

Figure 14: Examples of *homogeneous* background regions.

f2) Increase chroma and value to enhance the foreground. Following this guideline, we apply a global transformation to the original color $\mathbf{R}$:

$$\begin{aligned} H(\mathbf{R}') &= H(\mathbf{R}) \\ V(\mathbf{R}') &= k_v * V(\mathbf{R}) + V_0 \\ C(\mathbf{R}') &= k_c * C(\mathbf{R}) \end{aligned} \qquad (2)$$

Here, k_v, k_c and V_0 are constants that can be learned from the foreground color database by linear fitting. For our color database, the k_v, k_c and V_0 are 0.8, 1.8 and 2.2, respectively. Thus, the guideline is converted to a more easily formulated guideline: tend to use the color that is close to the reference color $\mathbf{R}'$.

Summing up the above guidelines, we give the definition of every energy term in Equation (1) using the following *pseudo-codes*:

1) $E_{b1}(x_i)$ for $i \in \mathscr{V}_b$: corresponding to guideline b1), b2)

```
If $\Omega_i$ is isolated and $S(\Omega_i) < S_0$ and $P(\Omega_i) < P_0$ //guideline b2)
  If $V(\mathbf{P}_k) > 9.5$ and $C(\mathbf{P}_k) < 1$
    $E_{b1}(x_i) = 0$;
  Else
    $E_{b1}(x_i) = 10000$;
Else                                                  //guideline b1)
  $E_{b1}(x_i) = D_{cv}(\mathbf{R}_i, \mathbf{O}_k) + \omega_S * |S(\Omega_i) - S_k| + \omega_P * |P(\Omega_i) - P_k|$
```

2) $E_{b2}(x_i, x_j)$ for $(i, j) \in \mathscr{E}_b$: corresponding to guideline b3), b4)

```
If $\Omega_i$ and $\Omega_j$ are adjacent //guideline b4)
  $E_{b2}(x_i, x_j) = \omega_I * |(I(\Omega_i) - I(\Omega_j) - (I_{ki} - I_{kj})|$
Else                        //means $\Omega_i$ and $\Omega_j$ are homogeneous, guideline b3)
  If $C(\mathbf{P}_{ki}) = C(\mathbf{P}_{kj})$ and $V(\mathbf{P}_{ki}) = V(\mathbf{P}_{kj})$
    $E_{b2}(x_i, x_j) = 0$
  Else
    $E_{b2}(x_i, x_j) = 10000$
```

3) $E_{f1}(x_i)$ for $i \in \mathscr{V}_f$: corresponding to guideline f2)

$$E_{f1}(x_i) = D_{cv}(\mathbf{R}_i, \mathbf{O}_k) + \omega_r * D_{cv}(\mathbf{R}'_i, \mathbf{P}_k)$$

4) $E_{f2}(x_i, x_j)$ for $(i, j) \in \mathscr{E}_f$: corresponding to guideline f1)

$$\begin{aligned} E_{f2}(x_i, x_j) = \; & \omega_c * |(C(\mathbf{R}_i) - C(\mathbf{R}_j)) - (C(\mathbf{P}_{ki}) - C(\mathbf{P}_{kj}))| \\ & + \omega_v * |(V(\mathbf{R}_i) - V(\mathbf{R}_j)) - (V(\mathbf{P}_{ki}) - V(\mathbf{P}_{kj}))| \end{aligned}$$

Here, $D_{cv}()$ represents the distance of two colors in the $C - V$ plane. The weights ω are user-defined values. The default weights are set as $\omega_S = 20$, $\omega_P = 50$, $\omega_I = 0.1$, $\omega_r = 1$, $\omega_c = 1$ and $\omega_v = 0.5$ in our implementation.

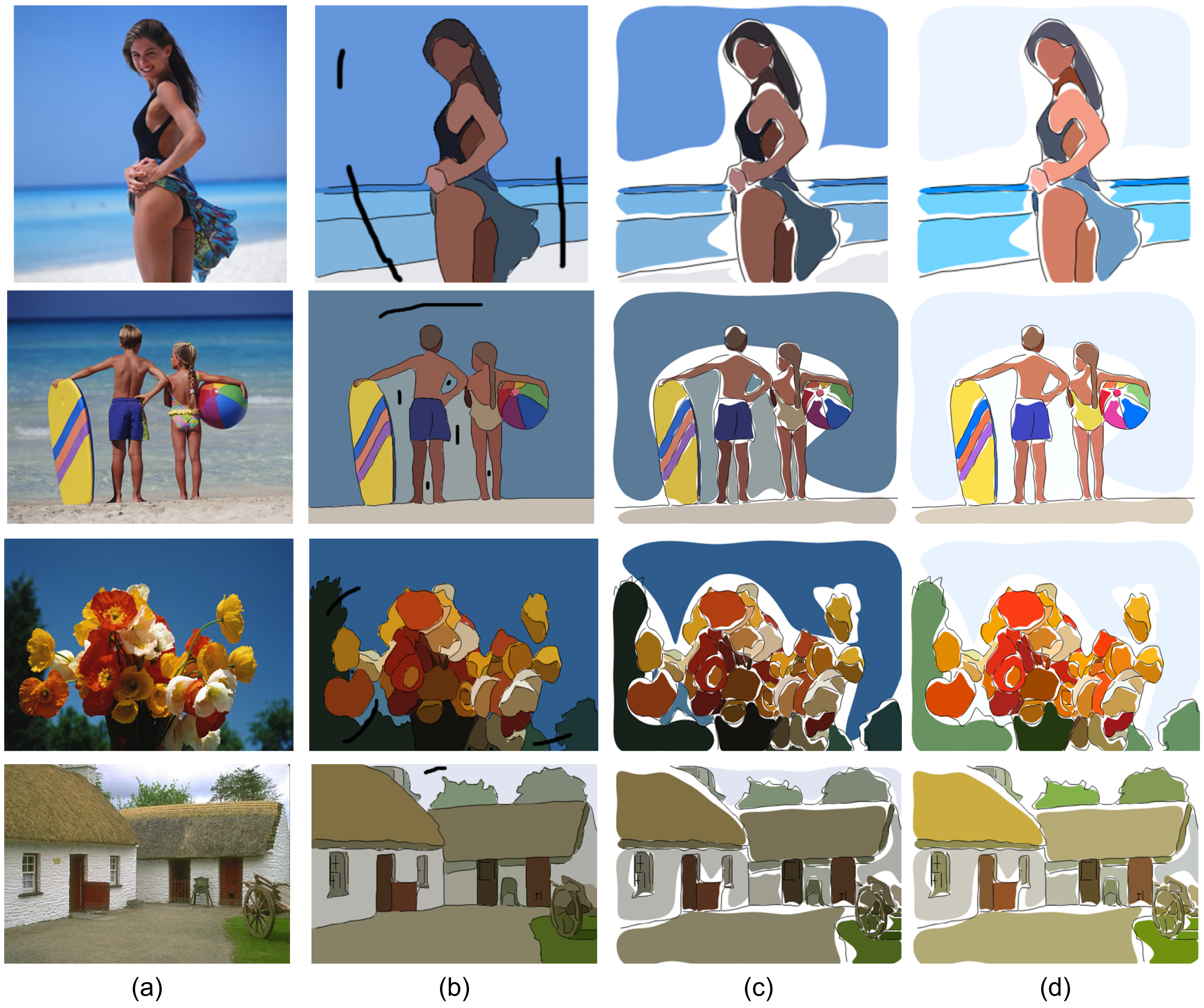

Figure 15: Generated Results. The column (a) is the input image. The column (b) is the results after pre-processing. The column (c) is the results after boundary shrinking. The column (d) is the final results after color shift.

4 Results

We show some generated results by our system in Figure 15. All experiments were run on a 2.8GHz Pentium IV with 512MB of RAM. The resolution of the four images in the Figure 15 are 300×304 ('GIRL'), 500×408 ('CHILDREN'), 500×331 ('FLOWERS') and 481×321 ('HOUSES'), respectively. We select 5 candidate colors for each background region and 20 candidate colors for each foreground region. In the Table 1 we list how many manual interventions are needed for segmentation and outline drawing to obtain such results, and the running time of the boundary shrinking stage and the color shift stage. We can see that the greatest cost is on the segmentation part. Fortunately, it is easily to roughly draw some strokes to finish these operations. As to the outline drawing part, only in the 'HOUSES' image, we draw some strokes in the window and door regions. For the other 3 images, this step is fully or almost fully automatic.

Table 1: Performance of our system.

Images	Interventions for segmentation	Interventions for outline drawing	Running time of boundary shrinking	Running time of color shift
GIRL (300 X 304)	4 Mergings 9 Splittings 8 Editings	No intervention	2	1.5s
CHILDREN (500 X 408)	3 Mergings 5 Splittings 7 Editings	Add 1 stroke	11s	3s
FLOWERS (500 X 331)	11 Mergings 11 Splittings 10 Editings	No intervention	15s	12s
HOUSES (481 X 321)	1 Mergings 28 Splittings 12 Editings	Add 15 stroke Break 1 stroke	11s	9s

5 Conclusion

We demonstrate a new technique for producing color sketches in the free-hand drawing style. This artistic style has two interesting properties that existing image abstracting techniques do not cover. These properties can be often observed in real artists' drawings, i.e., incompletely filled regions bring out highlights and add to the artistic-looking effect, and shifted colors emphasize different regions. We propose a new NPR system to simulate this artistic free-hand drawing style. In the pre-processing stage, a friendly user interface helps the user to interactively segment the image into several regions and to draw important outlines. Then, an illumination based region shrinking algorithm and a global optimization based color shift algorithm are used to obtain a stylized color sketch.

In the global optimization framework of color selections, all candidate colors are selected according to the color database, and the global transformation term of foreground colors in the energy formulation also depends on the color database. Therefore the database significantly impacts the quality of results. In our system, we choose 46 images with 85 background color regions and 861 foreground color regions to compose the color database. It already works for a large range of outdoor images. Since this painting style is more suitable for outdoor daylight images, e.g., people, building exteriors and outdoor scenery, both the training set and the images to be processed do not deviate from this range in our experiments. If the style of the image to be processed is totally different from all images in the database, the technique cannot produce a satisfactory result.

The other factor that has a big impact on the quality of the final color sketch is the image segmentation result in the pre-processing stage. As we know, there exists no automatic segmentation algorithm that can produce perfect results for all images. In this system, we provide some tools to help the user interactively modify the segmentation and outlines. Of course, to get a satisfactory result, a more complex image will need more manual interventions.

References

BARRETT, W. A., AND CHENEY, A. S. 2002. Object-based image editing. In *Proceedings of ACM SIGGRAPH 2002*, 777–784.

COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 5, 1–18.

COOPER, E. 2000. *California: A Sketchbook*. Chronicle Books.

DECARLO, D., AND SANTELLA., A. 2002. Stylization and abstraction of photograph. In *Proceedings of ACM SIGGRAPH 2002*, 769–776.

FREEMAN, W., PASZTOR, E., AND CARMICHAEL, O. 2000. Learning low-level vision. *Int. J. Computer Vision 40*, 1, 25–47.

GOOCH, A. A., OLSEN, S. C., TUMBLIN, J., AND GOOCH, B. 2005. Color2gray: Salience-preserving color removal. In *Proceedings of ACM SIGGRAPH 05*, 634–639.

HAEBERLI, P. 1990. Paint by numbers: Abstract image representations. In *Proceedings of ACM SIGGRAPH 90*, 207–214.

HEALEY, C. G., TATEOSIAN, L., ENNS, J. T., AND REMPLE, M. 2004. Perceptually based brush strokes for nonphotorealistic visualization. *ACM Transactions on Graphics (TOG) 23*, 1.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., 327–340.

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of ACM SIGGRAPH 98*, 453–460.

HERTZMANN, A. 2001. Paint by relaxation. *Computer Graphics International*, 453–460.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. In *Proceedings of ACM SIGGRAPH 2004*.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. In *Proceedings of ACM SIGGRAPH 04*, 303–308.

LITWINOWICZ, P. 1997. Processing images and video for an impressionist effect. In *Proceedings of ACM SIGGRAPH 97*, 407–414.

MEIER, B. J., SPALTER, A. M., AND KARELITZ, D. B. 2004. Interactive color palette tools. *IEEE Computer Graphics and Applications 24*, 3, 64–72.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California.

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

SHIRAISHI, M., AND YAMAGUCHI, Y. 2000. An algorithm for automatic painterly rendering based on local source image approximation. In *the First International Symp. on Non-photorealistic Animation and Rendering (NPAR)*, 53–58.

WEISS, Y. 1996. Interpreting images by propagating bayesian beliefs. *Advances in Neural Information Processing Systems*, 908–914.

WELSH, T., ASHIKHMIN, M., AND MUELLER, K. 2002. Transferring color to greyscale images. In *Proceedings of ACM SIGGRAPH 2002*.

WYSZECKI, G., AND STILES, W. S. 1982. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley Interscience Publishers, New York.