Creating Colored Pencil Style Images by Drawing Strokes Based on Boundaries of Regions

Hajime Matsui* Henry Johan† Tomoyuki Nishita‡

The University of Tokyo

ABSTRACT

A lot of Non-Photorealistic Rendering methods have been proposed for creating an artistic image from an image. In this paper, we propose a method for creating colored pencil style images. The feature of colored pencil drawings is that, though colored pencil drawings are drawn with limited number of colors of pencils, we can express a lot of colors and gentle textures by changing the strengths when drawing strokes and by overlapping strokes of different colors. In order to realize this feature, we determine which colors of pencils to use and how deep to push the pencils (equivalent to the strength when drawing strokes), then draw several types of strokes, such as strokes for outlines, basecoats, and shading, allowing the strokes to overlap each other. When we create strokes for shading, we make their directions to align along the boundaries of regions, resulting in images that are more like drawings made by human.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithm; I.3.m [Computer Graphics]: Miscellaneous—Non-Photorealistic Rendering

Keywords: non-photorealistic rendering, colored pencil, strokes creation, colored pencils selection, Kubelka-Munk

1 INTRODUCTION

In the early days of computer graphics, many researches have been done with the purpose for creating realistic images (photorealistic rendering). However, in recent years, researches for creating images that emphasize the *features* (for instance, edges) while omitting the small details of a scene, or images that look like paintings (*artistic style images*) have gained much attention. These research fields are the examples of Non-Photorealistic Rendering. Techniques for creating artistic images are proven to be useful for processing and synthesizing images, and currently available in many commercial image editing applications.

There are many styles of artistic images. However, most of them can be categorized into two categories, which are region-based images, such as mosaic and stained glass, and stroke-based images, such as oil painting and colored pencil drawing. In this research, we focus on colored pencil drawing. Up to now, there are only a small number of methods proposed for colored pencil drawing and one of the limitations of the methods is that they did not consider how to generate the strokes used in drawing. In order to help users, in particular those who are not so good at drawing, to easily create colored pencil style images, it is important to determine the directions of strokes automatically.

Colored pencil drawings are usually drawn with limited number of colors of pencils. For instance, the smallest colored pencils set consists of only twelve pencils of different colors. Nevertheless, we can express a lot of colors and gentle textures by changing the strengths when drawing strokes and by overlapping strokes of different colors. This is an important feature of colored pencil drawings.

To realize this feature, in our method, we determine which colors of pencils to use and how deep to push the pencils. Then, we draw several types of strokes, such as strokes for outlines, basecoats, and shading, allowing the strokes to overlap each other. We create strokes for shading such that their directions align along the boundaries of regions. This approach is similar to strokes drawn by human which are, in our observation, often drawn in one direction or aligned along the boundaries.

The main contributions of our method are:

- Automatic strokes creation such that they are drawn in one direction or aligned along the boundaries.
- Multiple colored pencils selection to express the colors in each region of an input image.

2 RELATED WORK

In the field of Non-Photorealistic Rendering, a lot of methods have been proposed for creating various artistic images. Since the input to our method is an image, we only describe the related methods that take an image as the input. Moreover, we restrict our discussion to the methods that create artistic images by drawing strokes.

Haeberli [3] presented a method to create impressionistic images by drawing an ordered collection of rectangular brush strokes. Salisbury *et al.* [13] described a method for creating pen-and-ink style images, by drawing strokes to represent texture and tone. Takagi *et al.* [16] proposed a method for creating colored pencil style images by performing volume rendering on the paper modeled based on the physical laws. Sousa *et al.* [15] proposed a method for creating graphite pencil drawings by performing low-level simulation models for wood-encased graphite pencils and drawing paper. Durand *et al.* [2] described a method for interactively creating tonal drawings. Kim *et al.* [7] presented a method for creating pen-and-ink illustrations by interpolating strokes specified by user. Rudolf *et al.* [12] presented a method for creating crayon drawings by using physically-inspired model of wax crayons. Many commercial softwares, such as Painter [19], also have various artistic brushes, including colored pencil style brushes. All the methods described above require users to specify the directions of strokes manually, and do not aim to create strokes automatically from the given input image.

Curtis *et al.* [1] proposed a method for creating watercolor paintings by using fluid simulation. In their paper, they also proposed a method for creating a watercolor painting from an input image automatically, but basically the strokes created by using their method have constant directions.

Litwinowicz [8] and Hertzmann [6] proposed painterly rendering methods, in which strokes are drawn in the direction perpendicular to the gradient of luminance of the input image. Shiraishi and Yamaguchi [14] extended Haeberli's work [3] by automatically

*e-mail: hajime@nis-lab.is.s.u-tokyo.ac.jp

†e-mail: henry@nis-lab.is.s.u-tokyo.ac.jp

‡e-mail: nis@nis-lab.is.s.u-tokyo.ac.jp

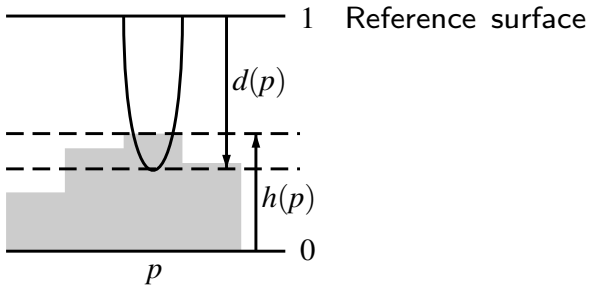


Figure 1: $d(p)$: the depth of the pencil at pixel p from a reference surface, $h(p)$: the height of paper at pixel p.

generating strokes properties such as locations, orientations, and sizes. The methods presented by Litwinowicz [8], Hertzmann [6], and Shiraishi and Yamaguchi [14] can create nice oil painting images, however since colored pencil drawing has different characteristics than oil painting, their methods cannot be applied directly to colored pencil drawing.

Yamamoto *et al.* [18] proposed a method for creating colored pencil drawings by using a texture based vector field visualization technique. However, the generated strokes are limited to straight lines. Commercial softwares, such as Photoshop [20], also have functions which convert images into artistic images including colored pencil drawing, but the results are not good.

Ostromoukhov and Hersch [9] proposed a method, multi-color and artistic dithering, for expressing a color by using limited number of colors. However, their method is designed for inks of printers and not suitable for the color selections when creating colored pencil style images. Yamamoto *et al.* [18] used the halftoning method proposed by Power *et al.* [10] for expressing a color by using two colors of pencils. However, in colored pencil drawing, it is quite often to draw overlapping strokes of more than two colors.

In this paper, we present a method for generating and rendering strokes for colored pencil drawings. The proposed method first divide an input image into several regions and then create several types of strokes for drawing. We create strokes based on the shape of the boundary of each region. From our observation, this approach resembles the way human draw strokes. Several colored pencils, whose colors are close to the color of the region, are selected and used for drawing strokes in that region. The strokes are rendered using the Kubelka-Munk model resulting in colored pencil style images.

3 Stroke Model

We represent a stroke as a point sequence and each stroke has a color. Each point of a stroke has x, y coordinates and the depth of the pencil from a reference surface (see Figure 1) when drawing the stroke at that point.

A stroke is drawn by updating the information of pixels within a radius W (user-specified value) from each point in the point sequence of the stroke. The user can control the width of the stroke by changing the value of W. In our experiments, we set the value $W = 2$. We, as well as Takagi *et al.* [16] and Yamamoto *et al.* [18], compute the color at each pixel by using the method proposed by Curtis *et al.* [1], which is based on the Kubelka-Munk model [4]. To use this model, beside the color of the pencil, we also have to know the thickness of the layer of the attached colored pencil pigments. Therefore, we have to determine the thickness of the pigment at each pixel.

In our method, the surface of paper is represented as a height field. The height field data of the paper is downloaded from the internet [21]. We compute the thickness $\delta(p)$ of a pigment at pixel p by using the following equation (see Figure 1).

$$\delta(p) = \mu(p)(d(p) + h(p) - 1), \tag{1}$$

where $h(p) \in [0,1]$ is the height of the paper at p, and $d(p) \in [0,1]$ is the depth of the pencil from a reference surface when drawing the stroke at p. This equation represents the fact that the deeper we push the pencil and the higher the paper is at a location, the more the pigments are going to attach.

$\mu(p)$ represents the probability that the pigments are going to attach at p, and computed as

$$\mu(p) = \begin{cases} \mu_0(1 - \frac{\delta(p)}{\delta_F}) + \mu_1 \frac{\delta(p)}{\delta_F} & (\delta(p) \le \delta_F) \\ \mu_1 & (\delta(p) > \delta_F) \end{cases}, \tag{2}$$

where δ_F is a threshold to determine if a location at the paper has already filled with pigments or not. We set $\delta_F = 0.5$ in our experiments. μ_0 is the probability that pigments are going to attach to the paper when a location in the paper has no pigment ($\delta(p) = 0$), whereas μ_1 is the probability when a location in the paper has already filled with enough pigments ($\delta(p) \ge \delta_F$). In our experiments, by setting $\mu_0 = 0.2$, $\mu_1 = 0.05$, we represent the fact that pigments are less likely to attach to the parts where other strokes have already been drawn.

4 Image Segmentation

Generally, when human draw a scene, he/she first selects the objects in the scene, and draw the objects in the scene one at a time. In our method, we perform image segmentation in order to extract the objects in the scene. We use the method presented in Healy *et al.* [5] for image segmentation. However, automatic image segmentation does not always produce results that satisfy the user. For example, if the input image is a photograph of natural scenery, the automatic image segmentation does not always work well, resulting in colored pencil style image that might not be nice. Therefore, we allow the user to modify the result of the image segmentation.

5 Strokes Creation

Although the number of colors of pencils that are available to be used to draw colored pencil drawings is limited, nevertheless colored pencil drawings that are rich in both the color and the texture can be created. This is possible due to the technique of overlapping strokes of different colors and strengths when making the drawings. To achieve these features, we create three types of strokes, that is the strokes for outlines, basecoats, and shading, on each region obtained in the image segmentation process. The directions of each type of strokes are determined based on the observation of strokes drawn by human.

In order to emphasize the objects in the foreground, sometimes, for the background regions, human only draw the basecoats. Therefore, we also allow the user to divide the input image into foreground and background area, then we draw only the basecoats in the background regions.

The three types of strokes are created as follows:

Outlines: Strokes for outlines are created by connecting the pixels on the boundaries of the regions.

Basecoats: Since the basecoats are seldom drawn by considering the boundaries of the regions, the strokes for basecoats are created by drawing strokes in a constant direction. In order to make the strokes look hand-drawn, we add some randomness when drawing the strokes.

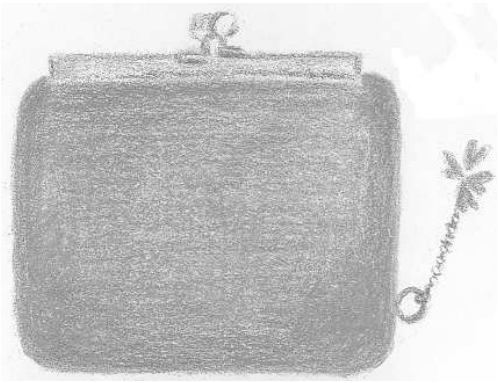


Figure 2: One of the drawings that we observed (Courtesy of Rie Matsubara [11]).

Shading: From our observation of colored pencil drawings drawn by artists (see Figure 2), we found out that strokes for shading are often drawn in one direction or aligned along the boundaries. Therefore, in our method, these strokes can be drawn in one direction for producing rough sketches or aligned along the boundaries of regions for producing more precise drawings. We will describe our method for aligning strokes along the boundaries in the rest of this section.

The basic idea of our approach is to select a part C_1 from the boundary of a region and to draw strokes that move away from C_1 (Figure 3(a)), and move closer to another part C_2 of the boundary (Figure 3(b)). In order to produce a nice shading, it is important to change the shapes of the strokes, that is, when drawing a stroke near C_1, then it shape should be almost align along C_1, whereas when drawing a stroke near C_2, then it shape should be almost align along C_2. To achieve this, in our method, we first divide the boundary into several parts (*feature edges*) and then compute the direction field of the strokes (*stroke fields*).

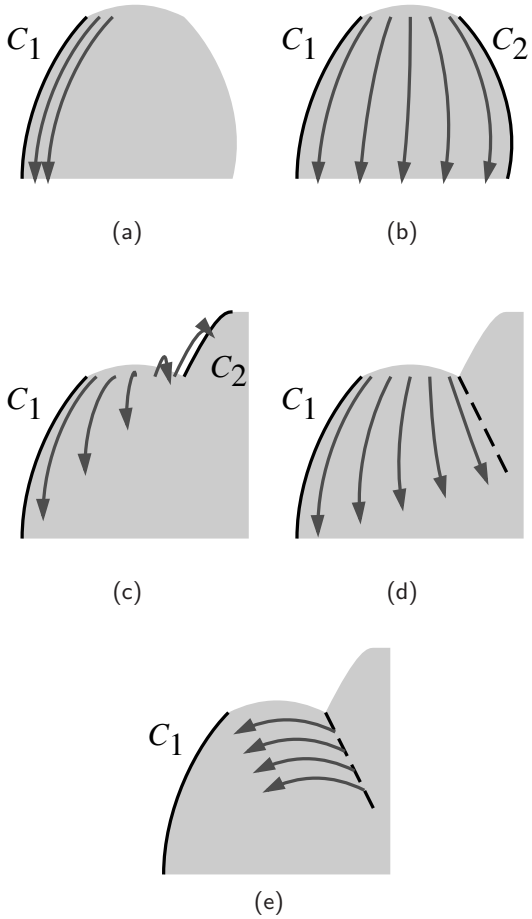


Figure 3: (a) Stroke directions along C_1, (b) stroke directions between C_1 and C_2, (c) undesired interpolation, (d) insertion of an additional edge (dashed line), and (e) the stroke field for the additional edge.

5.1 Boundary Division

When human draw strokes for shading, they usually do not draw strokes with complicated shapes, but draw strokes with the shapes of simple curves, that is curves with no high curvature regions and almost can be represented using quadratic equation. Therefore, in our method, we divide the boundary into simple curves and we call them feature edges. We use two thresholds θ_L and θ_A. θ_L is a threshold to divide a boundary at a location with large curvature, and θ_A is a threshold to divide at a location which has a large cumulative of curvature so the curve has a shape close to a quadratic curve. In our experiment, we set $\theta_L = 0.2\pi$ and $\theta_A = 0.5\pi$.

Let the point sequence representing the boundary be $\{P_i\}$, and angle $\angle P_{i-1}P_iP_{i+1}$ be θ_i, we divide the boundary as follows:

1. For each point P_i, if $\theta_i \geq \theta_L$, set P_i as the division point.

2. For each pair of the adjacent division points P_s, P_t, if a point P_i between them ($s < i < t$) satisfies $(\sum_{s<j<i} \theta_j) \geq \theta_A$, set P_i as the division point.

3. Repeat 2 until a new division point cannot be added.

We regard the curve between the adjacent division points as feature edge.

5.2 Stroke Fields for Shading

We make a stroke field like the one shown in Figure 3(b) for each feature edge by interpolating its two adjacent feature edges. However, in the case when the angle made by a feature edge and its adjacent one is concave, the interpolation of its two adjacent feature edges does not make natural stroke directions (Figure 3(c)). In this case, we can make a more natural stroke direction by inserting an additional edge (dashed line in Figure 3(d)). We also make a stroke field starting from the additional edge, by translating the adjacent feature edge (Figure 3(e)). We also regard the additional edges as the feature edges.

The additional edges are inserted as follows. For each example in Figure 4, we need to insert an additional edge at point P_i because angle $\angle P_{i-1}P_iP_{i+1}$ is concave. We have the following four cases.

1. The case when both $\angle P_{i-2}P_{i-1}P_i$ and $\angle P_{i+2}P_{i+1}P_i$ are convex (Figure 4(a)), we insert an additional edge P_iP_i' by interpolating $P_{i-1}P_{i-2}$ and $P_{i+1}P_{i+2}$ with the ratio of $|P_iP_{i+1}| : |P_iP_{i-1}|$ ($|PQ|$ means the length of line segment PQ).

2. The case when $\angle P_{i-2}P_{i-1}P_i$ is concave and $\angle P_{i+2}P_{i+1}P_i$ is convex (Figure 4(b)), we insert an additional edge by translating $P_{i+1}P_{i+2}$ such that P_{i+1} is translated to P_i.

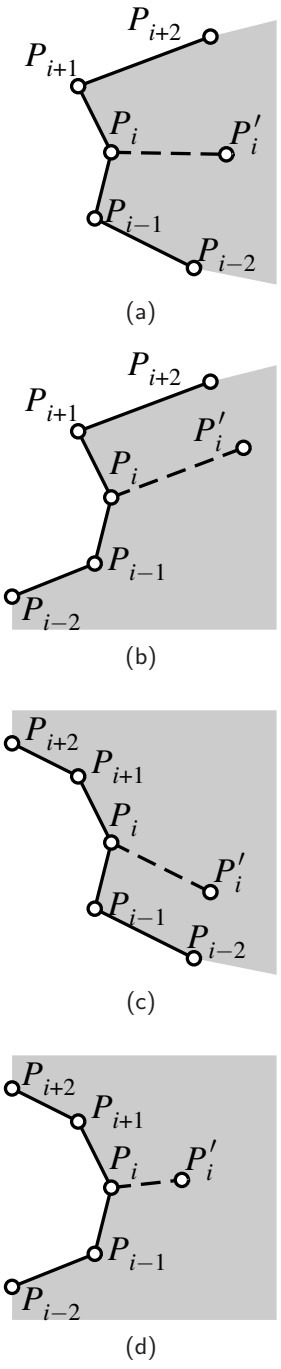


Figure 4: Additional edges when (a) both $\angle P_{i-2}P_{i-1}P_i$ and $\angle P_{i+2}P_{i+1}P_i$ are convex, (b) $\angle P_{i-2}P_{i-1}P_i$ is concave and $\angle P_{i+2}P_{i+1}P_i$ is convex, (c) $\angle P_{i-2}P_{i-1}P_i$ is convex and $\angle P_{i+2}P_{i+1}P_i$ is concave, and (d) both $\angle P_{i-2}P_{i-1}P_i$ and $\angle P_{i+2}P_{i+1}P_i$ are concave.

3. The case when $\angle P_{i-2}P_{i-1}P_i$ is convex and $\angle P_{i+2}P_{i+1}P_i$ is concave (Figure 4(c)), we insert an additional edge by translating $P_{i-1}P_{i-2}$ such that P_{i-1} is translated to P_i.

4. The case when both $\angle P_{i-2}P_{i-1}P_i$ and $\angle P_{i+2}P_{i+1}P_i$ are concave (Figure 4(d)), we insert an additional edge $P_iP'_i$ such that $\angle P_{i-1}P_iP'_i = \angle P_{i+1}P_iP'_i$ and that the length of $P_iP'_i$ is the average length of $P_{i-1}P_i$ and $P_{i+1}P_i$.

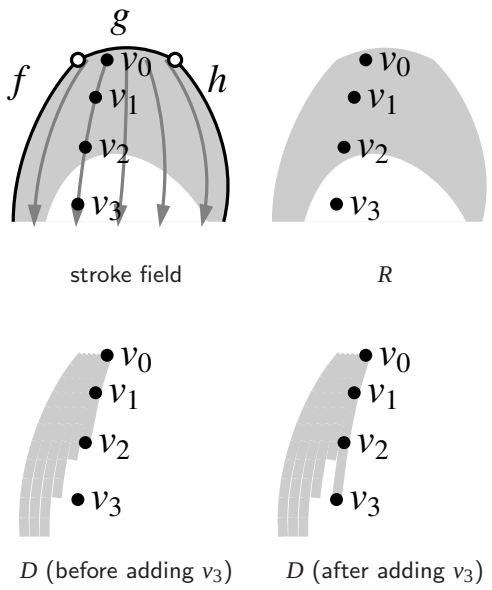


Figure 5: The addition of v_3 makes the difference between the reference image R and the drawn image D larger, so we do not add v_3.

5.3 Strokes for Shading

In order to create strokes on a region, we prepare two images, a *reference image R* and a *drawn image D*. Reference image R is an image which represents where and how many strokes we want to create. We first set a parameter r, which determine the density of the strokes, and then initialize $R(p) = r$ if pixel p is located inside the region being processed and $R(p) = 0$ if pixel p is located outside the region. Drawn image D is an image which represents where and how many strokes we have created. We initialize $D(p) = 0$ at every pixel p and increase the value at each pixel by one if the pigment of a colored pencil is attached to the pixel. By creating strokes such that the difference between the reference image and the drawn image is minimized, we can make, on average, the number of drawn strokes at each pixel in the region to r strokes. We set $r = 2$ in our experiment.

The strokes creation algorithm is as follows.

1. Select an unprocessed feature edge f.
2. Select a stroke field **G** where we can create strokes that are aligned along f.
3. Create strokes based on **G**.

In step 1, we select the longest feature edge because long feature edges have high possibilities that they represent the features of the regions. However, we do not select the additional edges because they are not on the boundaries of the region.

In step 2, we can create strokes along f based on the stroke field of the feature edge next to f. Since there are two feature edges next to f, there are two stroke fields that can be used to generate the strokes. We select the stroke field as follows. We first compute the priority for the stroke field of each adjacent edge of f, and then select the stroke field with a higher priority as **G**. We define the priority Pr of a feature edge, based on the lengths L_1, L_2 of its two adjacent feature edges, as

$$Pr = 1 - \frac{|L_1 - L_2|}{L_1 + L_2}. \tag{3}$$

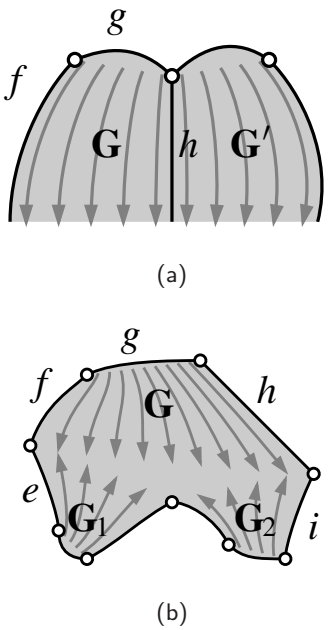


Figure 6: Realizing strokes coherency by (a) drawing strokes based on $\mathbf{G}'$ after drawing strokes based on $\mathbf{G}$, and in another example, (b) drawing strokes based on $\mathbf{G}_1$ and $\mathbf{G}_2$ after drawing strokes based on $\mathbf{G}$.

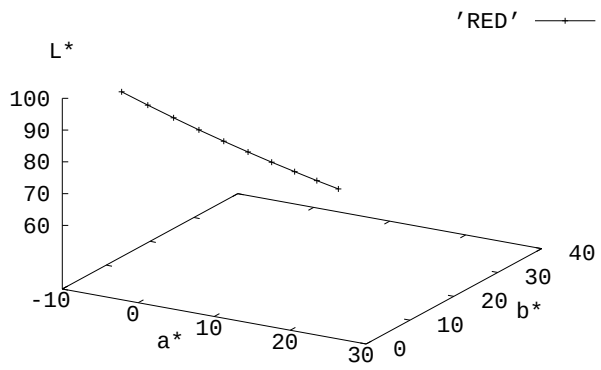


Figure 7: The graph of plotting the color (in the L*a*b* color space) produced by the red pencil when we vary the depth of the pencil when drawing strokes.

This equation is based on the observation that a stroke field is not good when the lengths of the two edges used in the interpolation differ a lot. We set the priority of the additional edges to be zero.

In step 3, we select a starting point v_0 on $\mathbf{G}$ and add the next point to the point sequence of the stroke one after another based on $\mathbf{G}$. When we add a point v_i, we update the drawn image and check if the difference between the reference image and the drawn image is increasing or not. If the difference is increasing, we stop adding a point and restore the drawn image (Figure 5). We repeat this process until we cannot add any new stroke.

In step 3, after we created strokes based on $\mathbf{G}$, we have to choose other stroke field and create strokes on it. In order to produce natural results, it is important to consider the coherency between strokes. To achieve coherency, we select the next stroke field as follows. In Figure 6, we assume that we have created strokes aligned along f in $\mathbf{G}$ by interpolating f and h. We can make strokes to appear more continuous (coherency between strokes) by taking the following two cases into account. If h is an additional edge, we select the stroke field $\mathbf{G}'(\neq \mathbf{G})$ (Figure 6(a)) on which we can create strokes aligned along h as $\mathbf{G}$ and perform step 3. If h is a boundary edge, we select each of the stroke fields $\mathbf{G}_1, \mathbf{G}_2(\neq \mathbf{G})$ (Figure 6(b)) on which we can create stroke along $e(\neq g)$ and $i(\neq g)$ respectively as $\mathbf{G}$ and perform step 3.

We perform the strokes generation algorithm until there is no feature edge left to be processed.

6 Colored Pencils Selection

Colored pencil drawings are drawn with limited number of colors of pencils. However, the input image can consist of variety of colors. In colored pencil drawings, by controlling the strength when drawing strokes, we can create many gradating colors. This is because, the stronger we push a pencil to a paper, the deeper the pencil is pushed and the more pencil pigments are attached to the paper. Thus, for each stroke, we need to determine which colored pencil to use and how deep to push the pencil when drawing strokes. In our method, we use multiple colored pencils whose colors have small differences with the color of the region being processed. We use the L*a*b* color space to compute the color difference because it reflects much the human perception.

The color of a region is computed by averaging the colors of the pixels in the region. As we stated before, the color produced by a colored pencil varies depending on the depth of the pencil when drawing strokes. Therefore, for each colored pencil, we compute a *color curve*. A color curve is a result of plotting the color (in the L*a*b* color space) produced by a colored pencil when we vary the depth (i.e, depth is the parameter of the curve). The example of the color curve is shown in Figure 7. The color at each point of the curve is the result of drawing a stroke on a white paper with a specified depth. The color is computed based on the Kubelka-Munk model [4].

In order to determine which pencil to use and how deep to push it when drawing a region, we compute the distance from the color of the region to the color curves of all pencils. The colored pencil whose color curves have distances to the color of the region less than a threshold are selected as the pencils used in drawing. When we compute the distance to the color curve, we compute the closest point on the curve. We use the parameter value at the closest point as the depth of the pencil when drawing strokes.

In the case when only one colored pencil is selected, we use this pencil for drawing all types of strokes. In this case, we vary the depth such that the outlines and the basecoats are drawn with a shallowly pushed pencil whereas the shading are drawn with deeply pushed one. In the case when more than two pencils are selected, we use the pencil with the brightest color for drawing strokes for outlines and basecoats, and use the rest of the pencils in the order of decreased brightness for drawing strokes for shading. This approach is based on the suggestion found in the book by Matsubara and Miyoshi [11].

Although we perform the image segmentation by color, each segmented region can have color gradation in it. In order to express color gradation, in our method, we perform color clustering in each region using the k-means algorithm [17]. We set the number of clusters to the number of available colored pencils and the colors of the initial clusters to those of colored pencils when drawing on paper. Usually, most of the clusters resulting from the clustering are

Table 1: The computational time and the number of strokes for creating the examples.

Image	Image size	Time	#Strokes
Figure 11(a)	435×535	45 s	5485
Figure 11(b)	360×360	49 s	8414
Figure 11(c)	603×410	238 s	24775

empty sets. For each cluster that is not empty, we select color pencils based on the average color of the pixels in it using the method described above, and create strokes based on the boundaries of the region (not the cluster). Through experiment, we found that, by using color clustering, detailed variations of colors in regions can be expressed.

7 Results

In our experiments, we used twelve colored pencils (see Figure 8). In Figure 9, we compared a hand-drawn colored pencil drawing and an image rendered using the method described in Section 3. We confirmed that our rendering method produces results that resemble real drawings.

Figures 10(a)-(c) show the input images and Figures 11(a)-(c) show the resulting colored pencil style images created automatically using the proposed method. Strokes aligned along the boundaries of regions can be seen. We can also see that multiple colored pencils were used to draw overlapping strokes. The computational times and the numbers of strokes are shown in Table 1. The computations were performed on a machine with a Pentium 4 3GHz CPU.

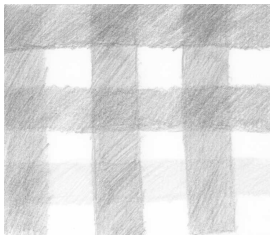
For comparison, Figure 11(d) was created using the gradient-based method proposed by Litwinowicz [8] for determining the stroke directions. Generally, the gradient-based method produces stroke directions that suit for oil-painting but not for colored pencil drawing. It is obvious that our method produces results more resemble to colored pencil drawing than the gradient-based method.

8 Conclusion

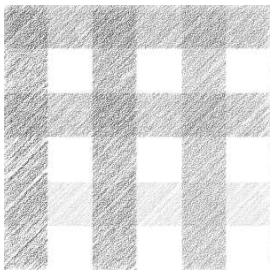
In this paper, we have presented a method for creating a colored pencil style image from an input image. The proposed method first divide the input image into regions and then create strokes on each region. The features of our method are as follows.

- In order to simulate the drawing techniques for colored pencil drawing, we create several types of strokes, such as strokes for outlines, basecoats, and shading, allowing them to overlap each other.
- We create strokes for shading so that they align along the boundaries of the regions by interpolating two feature edges after dividing the boundaries into feature edges. This approach is similar to the approach used by human when drawing strokes for shading.
- In order to produce various colors from limited number of colors of pencils, we compute color curves and use these curves for selecting colored pencils and determining the depths of the pencils for drawing strokes.
- We use a simple model to compute the thickness of pigments on the paper and use the Kubelka-Munk model [4] for the final rendering, resulting in colored pencil style images.

Figure 8: Available colored pencils set.



Hand-drawn drawing



Proposed method

Figure 9: Comparison between a hand-drawn drawing and our rendering result.

As for the future work, we aim to extend this method for creating colored pencil style animations from an input video sequence.

References

[1] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In *Proc. of SIGGRAPH 97*, pages 421–430, 1997.

[2] Frédo Durand, Victor Ostromoukhov, Mathieu Miller, François Duranleau, and Julie Dorsey. Decoupling strokes and high-level attributes for interactive traditional drawing. In *Proc. of the 12th Eurographics Workshop on Rendering*, pages 71-82, 2001.

[3] Paul Haeberli. Paint by numbers: Abstract image representations. In *Computer Graphics (Proc. of SIGGRAPH 90)*, 24(4):207–214, 1990.

[4] Chet S. Haase and Gary W. Meyer. Modeling pigmented materials for realistic image synthesis. In *ACM Transactions on Graphics*, 11(4):305–335, 1992.

[5] Christopher G. Healy, Laura Tateosian, James T. Enns, and Mark Rempel. Perceptually-based brush strokes for non-photorealistic visualization. In *ACM Transactions on Graphics*, 23(1):64–96, 2004.

[6] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proc. of SIGGRAPH 98*, pages 453–460, 1998.

[7] Hye-Sun Kim, Hee-Jeong Jin, Young-Jung Yu, and Hwan-Gue Cho Creating pen-and-ink illustration using stroke morphing method. In *Proc. of Computer Graphics International 2001*, pages 113–120, 2001

[8] Peter Litwinowicz. Processing images and video for an impressionist effect. In *Proc. of SIGGRAPH 97*, pages 407–414, 1997.

[9] Victor Ostromoukhov and Roger D. Hersch. Multi-color and artistic dithering. In *Proc. of SIGGRAPH 99*, pages 425–432, 1999.

[10] Joanna L. Power, Brad S. West, Eric J. Stollnitz, and David H. Salesin. Reproducing color images as duotones. In *Proc. of SIGGRAPH 96*, pages 237–248, 1996.

[11] Rie Matsubara and Takako Miyoshi. Pleasant colored pencil - Anybody can easily draw. Nagaoka Publications, 1998. *(in Japanese)*

[12] David Rudolf, David Mould, and Eric Neufeld. Simulating wax crayons. In *Proc. of Pacific Graphics 2003*, pages 163–175, 2003.

[13] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. In *Proc. of SIGGRAPH 94*, pages 101–108, 1994.

[14] Michio Shiraishi and Yasushi Yamaguchi. An algorithm for automatic painterly rendering based on local source image approximation. In *Proc. of NPAR 2000*, pages 53–58, 2000.

[15] Mario C. Sousa and John W. Buchanan. Observational models of graphite pencil materials. In *Computer Graphics Forum*, 19(1):27–49, 2000.

[16] Saeko Takagi, Issei Fujishiro, and Masayuki Nakajima. Volumetric modeling of artistic techniques in colored pencil drawing. In *Proc. of Pacific Graphics 99*, pages 250–258, 1999.

[17] Julius T. Tou and Rafael C. Gonzalez. Pattern recognition principles, Addison-Wesley, Reading, MA, 1974.

[18] Shigefumi Yamamoto, Xiaoyang Mao, and Atsumi Imamiya. Colored pencil filter with custom colors. In *Proc. of Pacific Graphics 2004*, pages 329–338, 2004.

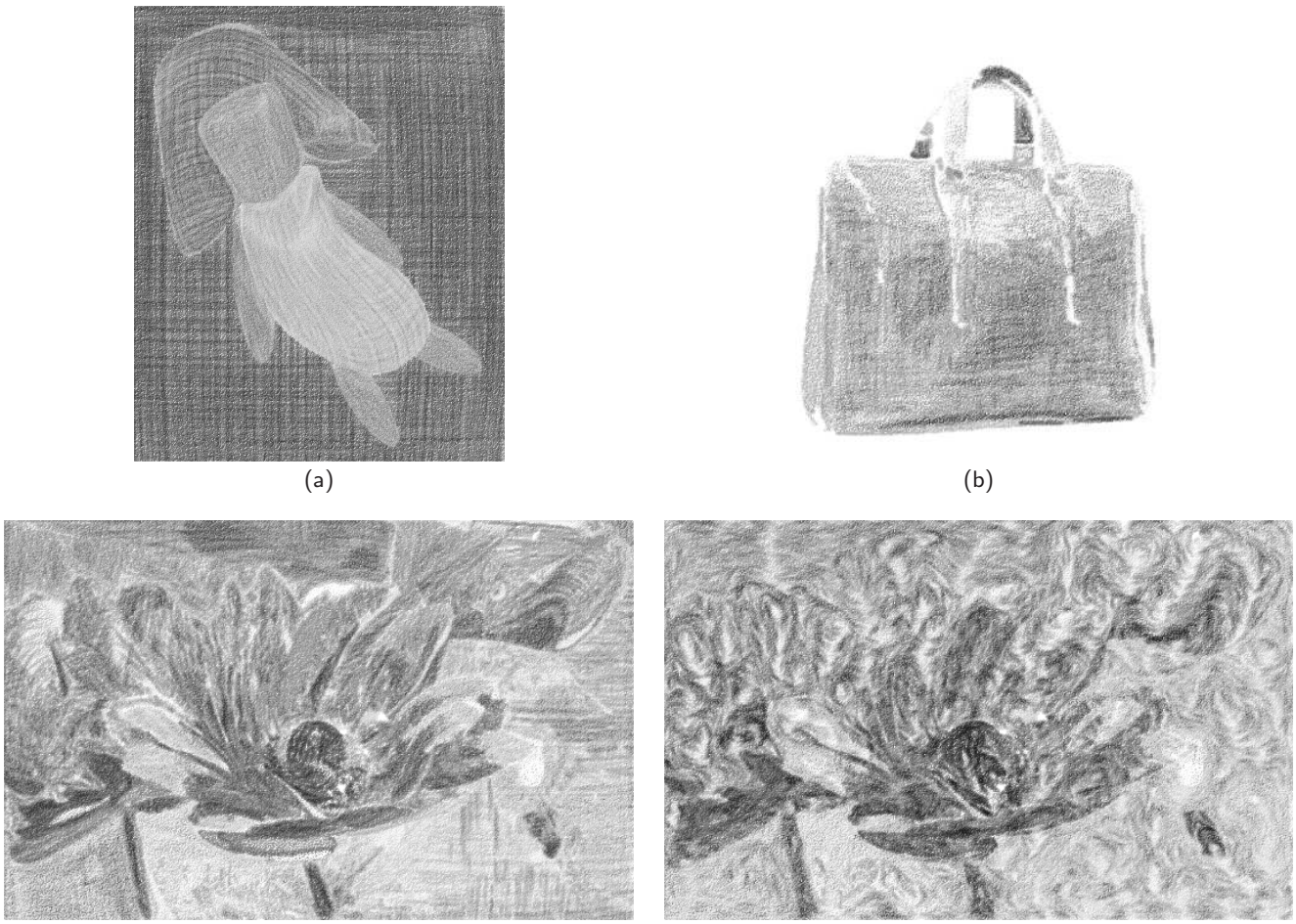
[19] Corel. Corel Painter.

[20] Adobe Systems. Adobe Photoshop.

[21] http://fweb.midi.co.jp/~buru_nyan/download/

(a) (b) (c)

Figure 10: The input images, (a) a doll, (b) a bag, and (c) flowers.



(a) (b)

(c) (d)

Figure 11: The resulting colored pencil images, (a) a doll, (b) a bag, (c) flowers created using the proposed method, and (d) flowers created using the gradient-based method.