Simulation of Color Pencil Drawing using LIC

Heekyung Yang[1] and Kyungha Min[2]
[1] Dept. of Computer Science, Graduate School, Sangmyung Univ.,
Hongji-dong, Jongro-gu, Seoul, Korea
[2] Div. of Digital Media, School of Software, Sangmyung Univ.,
Hongji-dong, Jongro-gu, Seoul, Korea
[e-mail: { hkyang ,rminkh}@smu.ac.kr]
*Corresponding author: Kyungha Min

Received August 2, 2012; revised November 29, 2012; accepted December 6, 2012; published December 27, 2012

Abstract

We present a novel approach for the simulation of color pencil effects using line integral convolution (LIC) to produce pencil drawings from images. Our key idea is to use a bilateral convolution filter to simulate the various effects of pencil strokes. Our filter resolves the drawbacks of the existing convolution-based schemes, and presents an intuitive control to mimic the properties of pencil strokes. We also present a scheme that determines stroke directions from the shapes to be drawn. Smooth tangent flows are used for the pixels close to feature lines, and partially parallel flows inside regions. The background is rendered using a flow of fixed direction. Using different styles of stroke directions increases the realism of the resulting images. This approach produces convincing pencil drawing effects from photographs.

Keywords: non-photorealistic rendering, pencil drawing, convolution, feature, coherent lines

This study was supported by a research grant from Sangmyung University in 2011.

http://dx.doi.org/10.3837/tiis.2012.12.015

1. Introduction

The simulation of pencil drawing is an interesting research topic in non-photorealistic rendering (NPR). Many researchers have presented monochrome pencil drawing techniques using approaches such as stroke overlapping [3][4][5][20][27], tonal maps [9][10][13][19][22][25], physically-based models [6][7][8][23] and LIC [12][14][15][16][21][26][28]. Some researchers have also developed color pencil drawing schemes using LIC [16][29][31][33] or stroke overlapping [18].

We present a technique for re-rendering color photographs into color pencil drawings in various styles. We attack this problem using two strategies: a scheme that simulates color pencil drawing using a bilateral convolution filter and a scheme that builds different types of stroke directions adapted to the shape to be depicted. The filter produces pencil drawing effects by integrating noise stored in the pixels in image space. We have designed the filter to simulate physical pencil drawing properties such as stroke thickness, stroke slant, hand pressure and paper texture. We apply strokes with directions determined in three different ways to suit the shapes in a scene. Smooth and curved strokes are applied to salient features such as boundaries of objects, smooth linear strokes are used to render the inside of objects, and a set of unidirectional strokes are used to draw the background of a scene.

Existing convolution-based approaches to the production of pencil drawing effects use line integral convolution (LIC), which integrates noise on the pixels in a particular direction. Even though this scheme is efficient, and produces acceptable pencil drawing effects, it has many defects. Users have little control over styles, and LIC techniques cannot represent the details of shapes to the extent that they can be depicted by an artist. Furthermore, LIC techniques make no attempt to reproduce the physical properties of pencil drawings [6], such as variable length, thickness, angle of strokes, and hand pressure.

Our color pencil drawing filter is a bilateral convolution filter. The kernel is a curved rectangle, which is configured so that the longer axis follows the drawing direction. Thus, the length of the kernel determines the stroke length and the width controls the stroke thickness. The density of the mark produced by a stroke is controlled by perturbing the direction of the longer axis. The weights used in the filter allows it to play the role of a bilateral filter, which prevents the drawbacks of LIC-based techniques, such as edge smearing, unwanted noise and incorrect tone. In Section 3.1, we illustrate these drawbacks and discuss how our filter resolves them by comparing the results. Furthermore, this filter can also mimic the effect of hand pressure and paper texture.

To produce realistic pencil drawing styles, we need to use strokes in different directions which match the shape being depicted. The stroke around the salient features of a scene flow smoothly close to the features, which are extracted using a difference-of-Gaussian (DoG) scheme. The directions of strokes in areas of smooth tonal change inside a region are determined by quantizing the smooth feature flow around its boundary. We can also use unidirectional hatching to fill in the background of a scene.

Fig. 1 provides an overview of our algorithm. In a pre-processing steps, we analyze the input photograph. This analysis includes feature extraction, building smooth feature flow and distance map. We also construct a color pencil drawing filter (see Section 3). Subsequently, we produce color pencil drawing effects by applying this filter to the photograph using the extracted information (see Section 4).

Our technical contributions are listed as follows:

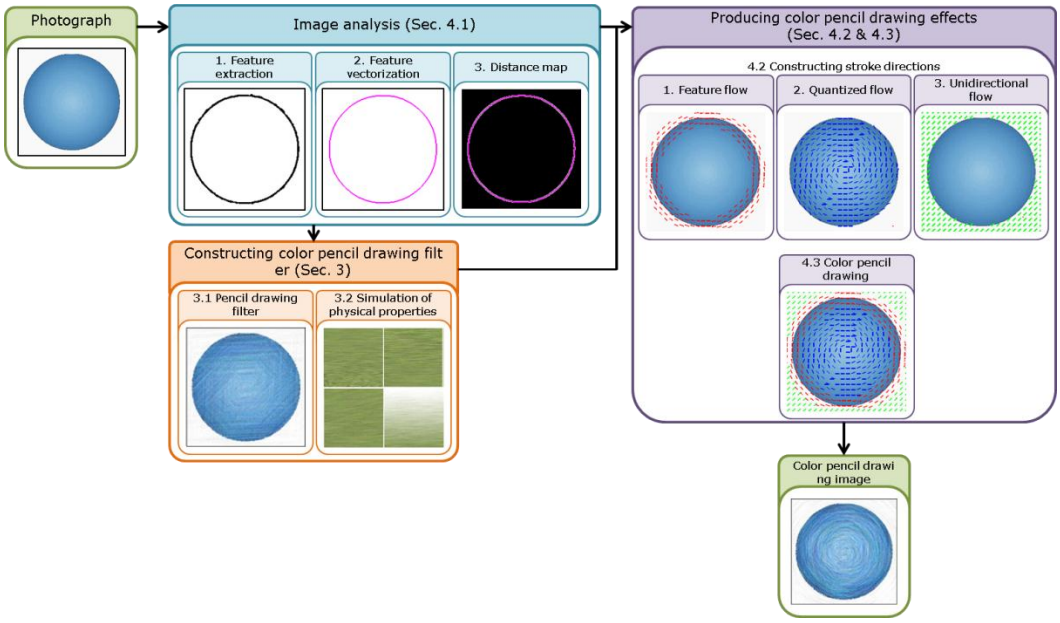


Fig. 1. An overview of the algorithm.

1. Simulating color pencil drawing properties: Our filter can simulate various pencil drawing properties such as stroke length, thickness, density, angle, hand pressure and paper texture.

2. Realistic stroke directions: We present three different types of stroke directions to match the shape to be drawn. These stroke directions successfully draw salient features, regions with smoothly changing tone, and backgrounds in a scene.

The rest of this paper is organized as follows: In Section 2, we briefly review work on monochrome and color pencil rendering. We introduce our simulation of color pencil drawing in Section 3, and explain how the stroke direction is determined and pencil drawing effects are produced in Section 4. In Section 5, we present our results and compare them with those from existing schemes. Finally, we conclude this paper and suggest future directions in Section 6.

2. Related Work

2.1 Monochrome Pencil Drawing

Existing monochrome pencil drawing schemes for re-rendering photographs can be classified into the simulation of physical models [6][7][8][23], the placement of strokes [3, 4,5,20], and the use of line integral convolution (LIC) [12][14][15][21][26].

Physical models that simulate the physical properties of materials including graphite and paper [6][7][8] were used in early stages to simulate the physical drawing process. Al Meraj et al. [23] have recently captured artists' hand motions to produce hand-drawn pencil lines. Although this approach has introduced pencil drawing to NPR field, it cannot produce pleasing pencil drawing images; nor can it support various pencil drawing styles. It also hard to control and requires considerable computations.

Stroke-based schemes place individual strokes in either user-defined [5] or content-based [20] directions. These schemes are not really suitable for creating the large number of strokes needed for hatching effects, and so the results tend to resemble line illustrations. Nevertheless Salisbury et al. [3] and Winkenbach and Salesin [4] have presented schemes mimicking pen-and-ink illustration in which strokes are overlapped to build various tones and patterns.

Line integral convolution (LIC) was originally designed to visualize a direction within an image by integrating values in the direction [2]. Mao et al. [12] applied the LIC scheme to simulate a pencil drawing effect by integrating noise distributed on an image along a uniform direction. Li and Huang [14] introduced an LIC scheme that integrates noise particles along the gradient directions at each pixel. Yamamoto et al. [15] extended Mao et al.'s scheme by adding boundaries and paper effects. The results from different layers are overlapped for the final result. Xie et al. [20] have implemented an LIC scheme on a graphics processing unit (GPU) and applied it to video. Yang and Min [26] combined the LIC with edge tangent flow (ETF) and produced smooth pencil drawing effects, but like other LIC techniques, this approach suffers from lack of detail and incorrect tone. Even though LIC schemes can produce hatching effects efficiently, the results look like rough sketches. Hata et al. [29] presented a saliency map-based LIC scheme that emphasizes and important regions of an image and eliminates unattended regions. The saliency map is a computational model that predicts the focus of attention of an image. A detail control algorithm using multiresolutional pyramid is presented to adapt the rendering parameters that control the density, orientation and width of pencil strokes.

The other approach to pencil rendering is to start with a 3D mesh model [9][10][13] Most schemes of this sort use a tonal map, which is created by capturing pencil strokes and hatching textures of various intensities. Tonal map techniques can be used to produce results ranging from rough sketches to line illustrations [22], and they can operate in real time [19]. However, a tonal map scheme cannot support the re-rendering of images, because images contain complicated objects without an explicit 3D representation. This makes it difficult to determine the direction along which the tonal map should be applied.

2.2 Color pencil drawing

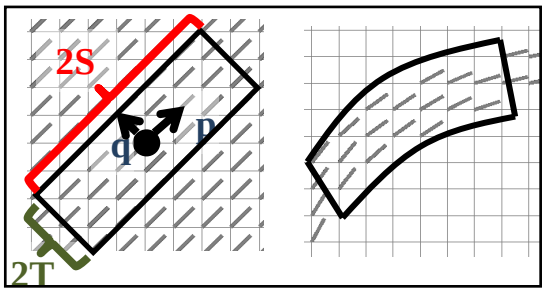
Research on simulating color pencil drawing is sparse. Mao et. al.[17] have presented a pioneering work about color pencil drawing filter using LIC. They applied the LIC filter for the individual R, G, and B channels for a color image and gradient-based directions for a pencil stroke. However, they suffer from the incorrect stroke direction and unclear boundaries of objects. Yamamoto et al. [16] used the LIC scheme to simulate color pencil drawing by requiring a user to define two dominant colors to each segmented region of an image. The LIC applied to each region integrates the noise of each color and creates two monochrome results. These results are combined to a color pencil drawing by the Kubelka-Monk method [1] Matsui et al. [27] developed a stroke-based scheme in which strokes are aligned along offset curves from the contours. Murakami et al. [18] have simulated graphite media such as charcoal, crayon and pencil using a stroke-based scheme. The effect of a stroke is produced from the texture of papers which are illuminated in twelve different directions. Xie et al. [28] produced color pencil drawing textures by coloring gray pencil drawing textures. In the coloring step, users specify their desired colors to the regions of the gray texture. The tone of the gray texture is converted to the specified color using local color-proliferation theory. Xie et al. [31] presented a colored pencil drawing scheme that uses the original color information instead of black dots. They also determined stroke orientation using the flow extruded from the luminance of the original image. Recently, Lu et al. [32] presented a color pencil drawing

work that combines line drawing and tonal texture. In their work, the line drawing is executed from the gradient extracted from an image and the pencil texture is produced from the tonal map. However, their scheme cannot produce pencil stroke patterns in depicting the tone of an image. Yang et al. [33] presented a stylized scheme for producing color pencil drawing effects from a photograph. They use an LIC scheme to produce pencil stroke patterns and present a color customization scheme that customizes the color to the user-selected color paletes. Their scheme, however, lacks of simulating the pencil drawing effects from the convolution filter.

3. Simulating Color Pencil Drawing

3.1 Pencil Drawing Filter

We introduce a filter that determines the color of a pixel in color pencil drawing from information close to that pixel. For this purpose, we generate color noise in the image space, which is represented with the CMY subtractive color model. Pencil is an artistic medium whose narrow tip leaves pigments over paper by the abrasion between the tip and the paper surface. Therefore, the direction along which the pencil tip is moving across the paper is important. Pixels in locations lying along that direction will have similar colors, while pixels in locations that cross that flow will have different colors. Thus we have designed our filter to sum noise on pixels located in the stroke directions.



(a) (b)

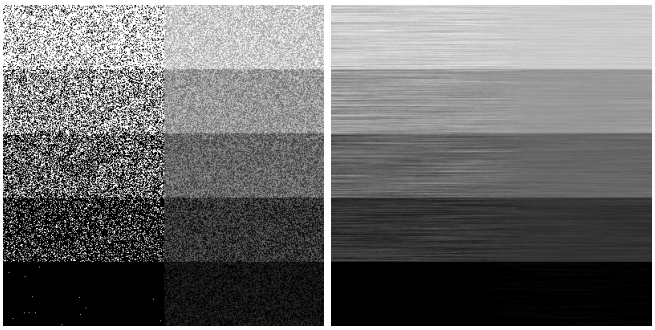
Fig. 2. The kernel of our filter: (a) applied to straight flow, and (b) to curved flow.

The kernel of our filter is a rectangle curved along its longer axis (see Fig. 2). The direction of the longer axis of the filter can be varied to match the stroke direction, which is constructed by the scheme in Section 4. The color of the pencil drawing, $R(\mathbf{x})$ at a pixel $\mathbf{x}$ is estimated by the following formula:

$$R(\mathrm{x}) = \mathrm{P}\left(\frac{1}{\nu}\int_{-S}^{S}\int_{-T}^{T} G_\tau(\|x-y\|)G_\sigma(|I(x)-I(y)|)N_t(y)dtds\right). \quad (1)$$

This formula is derived from the conventional LIC algorithm from [2] and the bilateral Gaussian convolution from [24]. The terms in Eq. (1) are explained as follows:

I(x)	the color at **x** in *Lab* color space	y	$\mathbf{x} + s\, p_t(\mathbf{p}) + t\, \mathbf{q}$
q	the direction of the short axis	$p_t(\mathbf{p})$	a random perturbation of the stroke
p	the direction of the main axis of the kernel	v	The weight normalization term: $\int_{-S}^{S}\int_{-T}^{T} G_\tau(\|x-y\|)G_\sigma(\|I(x)-I(y)\|)dtds.$
S	the length of the longer axis	T	the length of the shorter axis
N_t (**y**)	the noise stored in pixel **y**	t	the type of noise (see Fig. 3)
G_σ ( )	a Gaussian function, which plays the role of a bilateral filter		



(a) (b)

Fig. 3. Two different types of noise: (a) $t = B$ signifies binary noise and $t = G$ signifies grayscale noise, (b) The pencil drawing effects.

Noise at a pixel **x** is computed as follows:

$$N_B(x) = \begin{cases} 1, & \text{if } I(x) > \phi_1(0,1) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$N_G(x) = I(x) + \phi_2(-1,1)\epsilon, \tag{3}$$

where $\phi_1(0, 1)$ generates a random number in (0, 1). A pixel with higher intensity has higher probability to generate white noise ($N_B = 1$). ϕ_2 (-1, 1) generates selectively -1 or 1 at 50% possibility and ε is the offset of the noise. If ε = 0.2 and $I(\mathbf{x})$ is 0.5, for example, then $N_G(\mathbf{x}\})$ is either 0.3 or 0.7. The noise value is clipped into (0, 1). We convert the color of a photograph into CMY subtractive color model and generate noise for each color channel.

G_σ ( ) is a Gaussian function, which plays the role of a bilateral filter. The basic property of a bilateral filter is illustrated in Fig. 4. The conventional LIC-based techniques integrate noises across the boundary, so the noises from the regions of different colors may be integrated. This causes many problems such as smeared edges, unclear highlights and lost features (see Fig. 5 (a)). They avoided this problem by segmenting a scene into several regions, and limiting the integration range inside the region. This restriction, however, makes LIC-based techniques impractical and pains-taking. Our bilateral filter remedies this problem by assigning higher weights to the noise in a region of similar color and lower weights to the noise in a region of different color (see Fig. 4 for an explanation and Fig. 5 (b) for a comparison with the results from the existing LIC schemes).

$P(\mathbf{c})$, the pressure and paper effect for a color **c**, is applied by controlling $\mathbf{c}_v$, the V value of **c** in the HSV color space, as follows:

$$c_v' = c_v + \mathrm{p}.c + p_r, \tag{4}$$

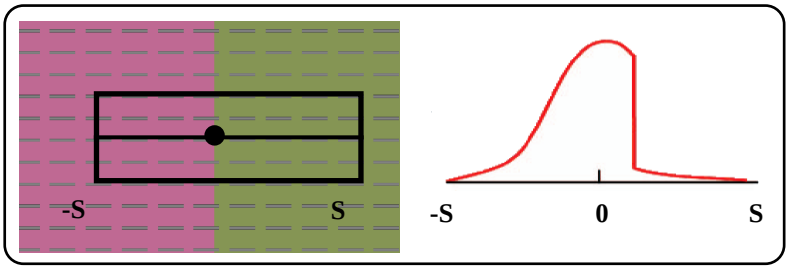


Fig. 4. The property of a bilateral filter: (a) the central pixel is located at red and some of the kernel is in green pixels; (b) the shape of $\boldsymbol{G_\sigma}$ ( ) from this kernel. The noises in the red pixels have higher weights than the noises in the green pixels.

where p.c, the paper effect, is estimated as κ **n** · **p**. κ is the weight controlling this effect, **n** is the normal to the paper texture, and **p** is the stroke direction. Hand pressure is denoted by p_r. This approach is derived from the formula suggested by the observational models [7,8].

Fig. 5. The comparison of the results. For comparison, both images are produced with a fixed direction and *S* is 10. Compare the boundary between two colors.

3.2 Simulation Of Physical Properties

The style of pencil drawing is determined by the stroke properties. We can vary the following four basic properties: stroke length, stroke thickness, stroke density and paper texture effect. Other properties such as stroke angle and hand pressures can be represented by combining the four basic properties.

1. Stroke length varies depending on the shape and tone being depicted. Our filter changes the stroke length by changing *S*, which controls the integration range along the longer axis of the filter. We use 5 for the shortest strokes and 20 for the longest strokes (see Fig. 6 (a)).

2. Stroke thickness is affected by the area of the pencil tip. Thin strokes are used to produce details while thick strokes are used to draw smooth tone. Thickness is represented by *T*, the integration range along the shorter axis of the filter. To increase the thickness, more pixels away from the stroke direction need to be included in the integration process. This is achieved by increasing *T*, the integration range along the short axis. We set *T* to 3 to produce the thickest strokes (see Fig. 6 (b)).

3. Stroke density is determined by the physical properties of the pencil core and the drawing style. A slanted pencil tip, for example, makes coarse marks. Dense strokes produce darker marks from a small amount of pixel noise, while coarse strokes produce less intense marks as the amount of noise increases. We change the density in three ways within our filter. One is to increase the perturbation p_t ( ), which makes the stroke coarser. Another way is to alter the type of noise. Generally, binary noise produces coarser results than grayscale noise. The third way is the pressure, which interacts with the paper texture. Reducing the pressure factor emphasizes the paper texture, making the stroke coarser (see Fig. 6 (c)].

4. Paper texture becomes increasingly important as strokes become slanted and hand pressure is reduced. The paper texture, which is sampled from real paper, is represented as a height map. We compute a normal vector at each pixel of the map. We assume that the amount of pigment left on the paper surface is proportional to the normal and the stroke direction (see Fig. 6 (d)).

Other variable factors such as stroke angle, and hand pressure [6] can be mimicked by combining some of our properties. A stroke with an angled pencil point is represented by a thicker and coarser stroke with an increased paper texture effect. Hand pressure is represented by the density and thicknesses (See Fig. 6 (e)).

(a) (b) (c) (d) (e)

Fig. 6. Simulating the physical properties of pencil drawings using our filter: (a) different stroke lengths (S = 5 at bottom and S = 20 at top); (b) different stroke thicknesses (T = 1 at bottom and T = 3 at top); (c) different stroke densities (perturbation angle is 0° at bottom and 20° at top); (d) different paper textures can be modeled (finer textures at bottom, and coarser textures at top); (e) different hand pressures (p_r = 0 at bottom and p_r = -0.8 at top).

4. Producing Color Pencil Drawings

Preprocessing

At the image processing stage, we execute the following tasks: (i) extract features from the input photograph, (ii) vectorize the features, and (iii) build a distance map of the flow vectors.

4.1. Feature Extraction

The analysis of an input photograph begins with the extraction of features using edge tangent flow (ETF), followed by the application of DoG filters [24], where the ETF is computed by the following formula:

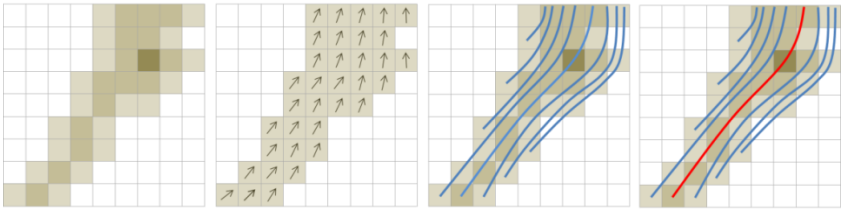
$$t'(x) = \frac{1}{k}\iint_{\Omega_\mu} \phi(x,y)t(y)\omega_s(x,y)\omega_m(x,y)\omega_d(x,y)dy, \quad (5)$$

where $t(x)$ and $t'(x)$ are the original tangent vector and the smoothed tangent vector at pixel x, respectively. Ω_μ denotes the kernel of radius μ at x and k is the normalization term. The terms of Eq. (5) are listed as follows:

$\phi(x,y)$	the orientation correction term	$\omega_m(x,y)$	the magnitude weight term
$\omega_s(x,y)$	the spatial weight term	$\omega_d(x,y)$	the direction weight term

4.2. Feature Vectorization

The extracted features are represented as a set of pixels, which we vectorize to improve the efficiency of the subsequent processing [26].



(a) (b) (c) (d)

Fig. 7. The steps of vectorization in [26]: (a) The pixels of different feature magnitudes, (b) The ETF in the feature pixels, (c) Smooth flow computed along the ETF on the feature pixels, (d) The selected smooth flow whose feature magnitude is maximum.

4.3. Distance Map For Flow Vectors

For each pixel, a distance Γ inside the vectorized features, we estimate the distance between that pixel and the nearest vectorized feature. We make Γ very large, since determination of the stroke direction requires the distance from several near features. The results of image analysis are illustrated in Fig. 8.

Fig. 8. Image analysis of an input photograph: input photograph (left), the extracted features (middle), and the vectorized features with the distance map (right).

4.2 Constructing Stroke Directions

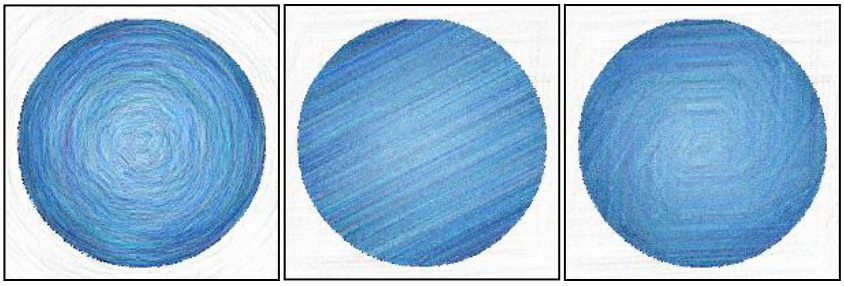
We apply three different stroke directions for producing pencil drawing effects.

1. **Feature flow**: Tangent flow vectors are used as the stroke direction for the pixels close to salient features. Thus, smooth tangent flow vector at the pixel is used as a stroke direction (**p** in Eq. (1)). The effects produced by this method of choosing the stroke direction can depict salient features and details of smooth shapes such as hair or fur.
2. **Quantized flow**: The quantized flow vectors derived from the smooth tangent vectors is used to build stroke directions for the pixels inside a region. The quantization process is similar to the mean shift algorithm [11], which quantizes colors according to their similarity and separation. A flow vector $\mathbf{p}_i$ at a pixel $\mathbf{x}_i$ is propagated to those pixels that

satisfy the following properties: (i) $d_a(\mathbf{x}_i) = d_a(\mathbf{x}_j)$, where $d_a(\mathbf{x})$ is the distance between $\mathbf{x}$ and a^{th} vectorized feature, and (ii) $\mathbf{p}_i \cdot \mathbf{p}_j < \delta$. We control the level of quantization using δ. Fig. 9 compares four different ways of determining stroke directions.

3. **Unidirectional flow**: Unidirectional vectors are used to render the background of a scene. Their direction is automatically chosen by averaging the gradient vectors in the background area; but users can specify their own direction.

Fig. 10 (a) illustrates the flow directions at the sampled points from an image.



(a) (b) (c)

Fig. 9. Three different stroke directions to depict the tone inside a region: (a) smooth tangent flow, (b) unidirectional flow, (c) quantized tangent flow with $\delta = 30°$, and (d) quantized tangent flow with $\delta = 45°$.

4.3 Color Pencil Drawing

Fig. 10. (top-left-(a)) The stroke directions at the sampled points: red lines are feature flow, blue lines are quantized flow, and green lines are unidirectional flow, (top-right-(b)) The pencil drawing effects from feature flow, (bottom-left-(c)) the effects from from quantized flow, (bottom-right(d)) the effects from unidirectional flow

At each pixel in an image, we determine the type of stroke direction to apply, and decide suitable parameters of the filter described in Eq. (1). We produce three color pencil drawings according to the type of stroke direction. These drawings are combined to construct a final result. The combination process is executed as follows: (i) we apply unidirectional flow for every pixels in a scene and produce pencil drawing effects (see Fig. 10 (d)), (ii) we produce pencil drawing effects for those pixels whose stroke directions are from feature flow or quantized flow (see Fig. 10 (b) and (c)), (iii) The effects in the pixels in step (ii) will overlap the effects produced in step (i). This strategy prevents holes that can happen on the boundary of the flows. The final result is shown in Fig. 15 (d).

5. Implementation and Results

5.1 Implementation

Our algorithm was implemented on a PC with an Intel Pentium QuadCore™ Q6600 CPU and 4Gb of main memory. The programming environment was Microsoft VisualStudio™ 2010 with the OpenGL libraries. **Fig. 11** shows the sample photographs. **Fig. 19** and **Fig. 20** show the pencil drawing results. The resolution, computational time and parameters are specified in **Table 1**.

Table 1. Parameters, time, and resolutions of the images in **Fig. 11**

Image	Resolution	Time (sec)	S	T	σ	τ	κ	perturbation	δ
17.(a)	800 x 800	108.5	40	2	10	15	0.1	25 °	10 °
17.(b)	371 x 548	26.1	38	2	15	15	0.2	10 °	10 °
17.(c)	704 x 352	48.1	43	1	13	15	0.2	15 °	10 °
18.(a)	700 x 840	37.7	43	1	15	15	0.1	15 °	10 °
18.(b)	720 x 860	68.7	40	1	15	15	0.0	10 °	10 °
18.(c)	544 x 900	61.2	42	2	13	15	0.1	10 °	10 °

Fig. 11. Sampled photographs

5.2 Comparison

We compared our results with those from some existing color pencil drawing schemes [16][27][28][31][32]. Instead of implementing those algorithms, we captured their original images from the literatures and used them to produce the images to compare. Yamamoto et al. [16] produce color pencil drawing effects by applying an LIC scheme for two principal colors in segmented regions of a scene. Since they do not vary the way of choosing stroke directions, the quality of the result is limited. Furthermore, they limited the colors of strokes to fill a region by two, which results in pencil drawings of unnatural colors. The sample photographs in [16] are re-rendered using our scheme in **Fig. 12**.

Matsui et al. [27] produce color pencil drawing effects by overlapping individual pencil strokes along directions which follow the contours of objects in a scene. The major drawback of their technique is that details of shapes in the scene are poorly reproduced. Another drawback is that the directions of their strokes, which are aligned along the contours, are too inflexible to produce the smooth tonal change represented by horizontally aligned strokes. The advantage of our technique over their approach is that varying the stroke directions can produce much more visually pleasing results (see **Fig. 13**).

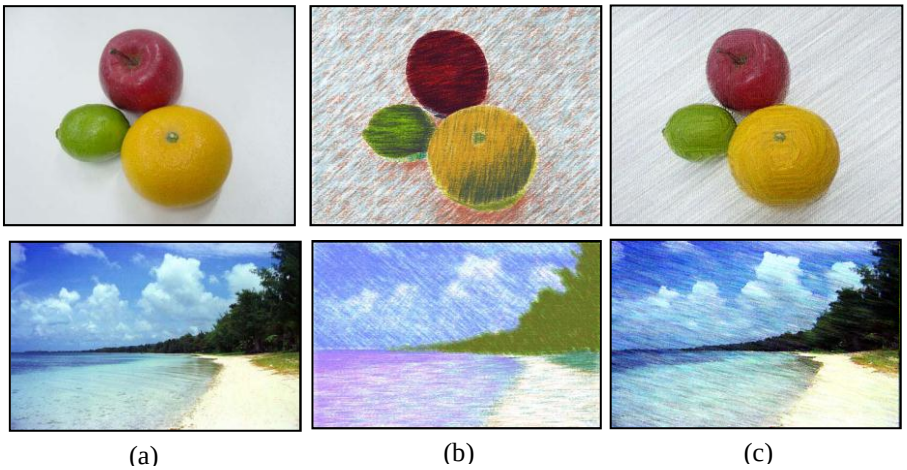


Fig. 12. Comparison with [16]: (a) Input, (b) Results from [16], (c) Our results

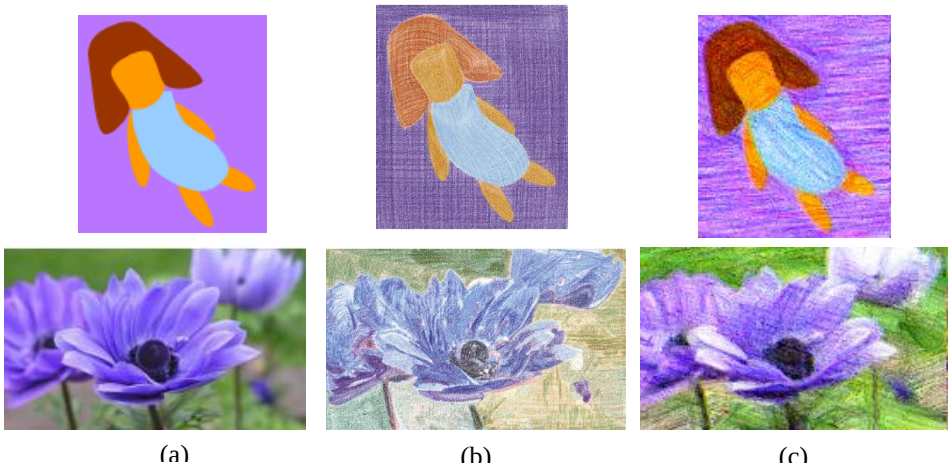


Fig. 13. Comparison with [27]: (a) Input, (b) Results from [27], (c) Our results

Xie et al. [28] produce color pencil drawing textures by coloring gray pencil drawing textures according to the user-specified colors. Since we sample the color from the input image, our scheme shows much realistic color pencil drawing effects. Furthermore, we apply LIC along various directions, which allows more realistic pencil stroke directions than Xie et al.'s (See **Fig. 14**).

Xie et al. [31] produce color pencil drawing from the colors sampled from an input image, which is similar to our work. They use a binary color value in HSV channel, while we use binary as well as grayscale value to represent color. They use the luminance value to determine the stroke direction. This strategy, however, has a limitation is conveying the boundary of objects (See **Fig .15**).

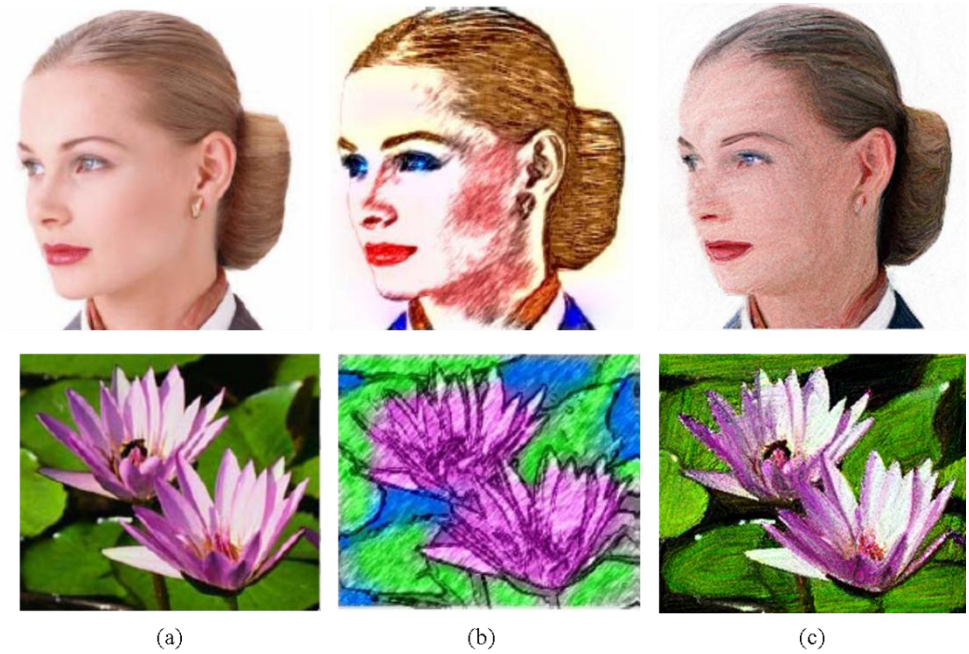


Fig. 14. Comparison with [28]: (a) Input, (b) Results from [28], (c) Our results

Fig. 15. Comparison with [31]: (a) Input, (b) Results from [31], (c) Our results

Lu et al. [32] produce color pencil drawing by combining lines extracted from the gradient of an input image and tone created from the tonal map of the image. Their scheme can produce salient line drawings and smooth tone depiction. This scheme, however, do not present pencil stroke patterns shown in pencil drawings. It cannot express smooth features such as hair and fur, neither. We compare the results from [32] and ours in **Fig. 16**.

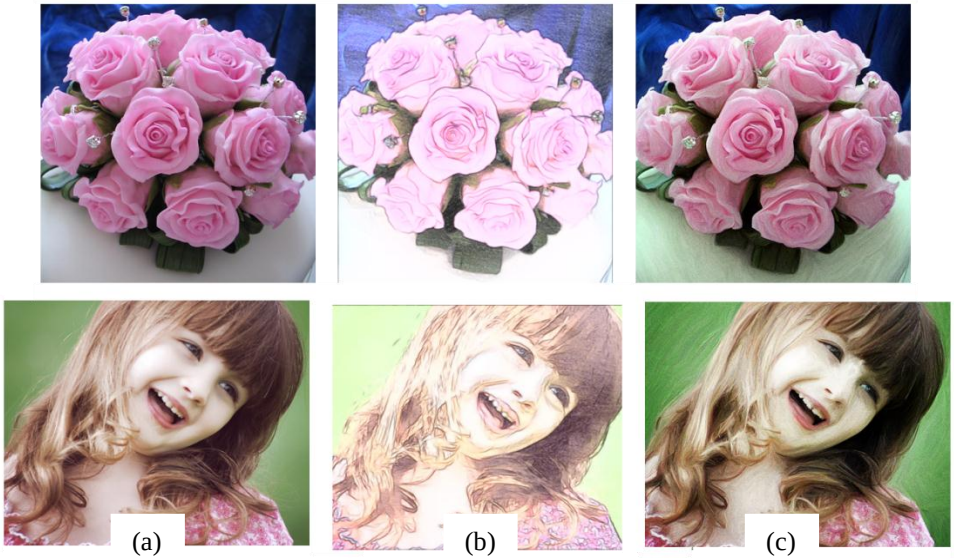


Fig. 16. Comparison with [32]: (a) Input, (b) Results from [32], (c) Our results

(a) (b) (c)

Fig. 17. Comparison with the product from a commercial software: (a) Input, (b) Results from Photoshop CS3, (c) Our results

Finally, we compare our algorithm with the result from commercial software. We choose Adobe PhotoShop CS3 and produce a pencil-drawing effect by applying its various filters. As you see in **Fig. 17**, the pencil-drawing effect is poor and it depends heavily on the user.

5.3 User study

To prove the superiority of our scheme, we compare our results with those from some selected related works. We analyze the images into three categories: (i) landscape, (ii) object, and (iii) portrait. The selected images from related works and our result are shown in Fig. 18. We sample two images for a landscape, an object, an a portrait.
For a user test, we sample 24 subjects and ask them to mark the most visually pleasing pencil drawing image from the candidates. As shown in **Fig. 16**, our result marks 18/24 for a landscape, 15/24 for an object and 19/24 for a portrait. The key factors why they have chosen our result are (i) the salient depiction of important features such as an eye for a portrait and the lamp for an object, (ii) the hatching patterns of pencil strokes shown in the backgrounds and (iii) the color similar to that of input photographs.

For a statistical analysis, we give +1 for those who prefer ours and -1 for those who prefer other results. Therefore, the positive total score reveals our scheme is preferred. We perform a paired t-test to determine whether our results are preferred or not. We test for three different categories of the images: landscape, object and portrait. The scores of the three tests are shown as follows:

	Test 1: Landscape	Test 2: Object	Test 3: Portrait
Prefers others (– 1)	6	9	5
Prefers ours (+1)	18	15	19
Total score	12	6	14

The total score (mean = 10.7, standard deviation = 4.2, number of samples = 3) is significantly greater than zero, $t(3) = -4.4$, two-tailed $p = 0.04$ providing an evidence that our result is preferred.

6. Conclusions and Future Plans

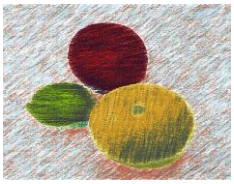
We have developed a scheme that re-renders photographs into color pencil drawings. We proposed a bilateral filter whose kernel is a curved rectangle, and various pencil drawing effects can be simulated by controlling the weights in the filter. We define stroke directions in different styles depending on the contents of a scene: feature flow for salient features, quantized flow for the tone inside objects, and unidirectional flow in the backgrounds. We have shown that this scheme can produce visually pleasing pencil drawings.
An important drawback of our scheme is that the quality of the pencil drawing images depends on the parameters chosen by users. It requires repeated process in testing and determining proper parameters. Another drawback is the computational time, which takes almost two minutes for complex and high resolution images.
An interesting future plan of our work is to apply our scheme in producing pencil-drawing video. A cartoonization of a video such as [30] is a similar application. To apply our scheme to video, we have to devise a temporal coherence of the effects between frames.

[28] 2

[31] 4

[Ours] 18

(a) Landscape



[16] 4

[32] 5

[Ours] 15

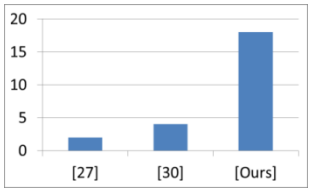
(b) Object

[28] 1

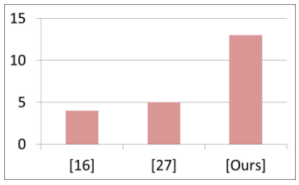
[32] 4

[Ours] 19

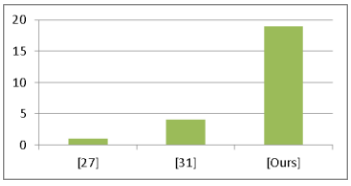
(c) Portrait



(a) Landscape



(b) Objects



(c) Portrait

(d) Graphs

Fig. 18. A user study. We compare our results with those from existing works in three categories: (a) landscape, (b) object and (c) portrait. The numbers below the images are the scores from the user study. In (d), we illustrate the scores in a graph.

We plan to present a user-friendly interface, which allows users to produce pencil drawings

conveniently. Finally, we intend to address the re-rendering video.

References

[1] C. Haase and G. Meyer, “Modeling pigmented materials for realistic image synthesis,” *ACM Trans. on Graphics*. vol. 11, no. 4, pp. 305-335, 1992. Article (CrossRef Link)
[2] B. Cabral and C. Leedom, "Imaging vector field using line integral convolution," In *Proc. of Siggraph 93*, pp. 263-270, 1993. Article (CrossRef Link)
[3] M. Salisbury, S. Anderson, R. Barzel, and D. Salesin, "Interactive pen-and-ink illustration," In *Proc. of Siggraph 94*, pp. 101-108, 1994. Article (CrossRef Link)
[4] G. Winkenbach and D. Salesin, "Computer generated pen-and-ink illustration," In *Proc. of Siggraph 94*, pp. 91-100, 1994. Article (CrossRef Link)
[5] M. Salisbury, M. Wong, J. Hughes, and D. Salesin, "Orientable textures for image-based pen-and-ink illustration," in *Proc. of Siggraph 97*, pp. 401-406, 1997. Article (CrossRef Link)
[6] M. C. Sousa and J. Buchanan, “Computer-generated graphite pencil rendering of 3D polygonal models,” in Proc. of Eurographics 1999, pp. 195-207, 1999. Article (CrossRef Link)
[7] M. C. Sousa and J. Buchanan, “Observational model of blenders and erasers in computer-generated pencil rendering,” in Proc. of Graphics Interface 1999, pp. 157-166, 1999. Article (CrossRef Link)
[8] S. Takagi, M. Nakajima, and I. Fujishiro, “Volumetric modeling of colored pencil drawing,” in Proc. of Pacific Graphics 1999, pp. 250-258 , 1999. Article (CrossRef Link)
[9] A. Lake, C. Marshall, M. Harris, and M. Blackstein, "Stylized rendering techniques for scalable real-time 3D animation," in Proc. of NPAR 00, pp.13-20, 2000. Article (CrossRef Link)
[10] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-time hatching," in Proc. of Siggraph 01, pp. 579-584, 2001. Article (CrossRef Link)
[11] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," IEEE Trans. on PAMI, Vol. 24, No. 5, pp. 603-619, 2002. Article (CrossRef Link)
[12] X. Mao, Y. Nagasaka, and A. Imamiya, "Automatic generation of pencil drawing using LIC," in ACM Siggraph 02 Abstractions and Applications, pp. 149, 2002. Article (CrossRef Link)
[13] M. Webb, E. Praun, A. Finkelstein, and H. Hoppe, "Fine tone control in hardware hatching," in Proc. of NPAR 02, pp. 53-58, 2002. Article (CrossRef Link)
[14] N. Li, and Z. Huang, "A feature-based pencil drawing method," in International Conference on Computer Graphics and Interactive Techniques, pp. 135-140, 2003. Article (CrossRef Link)
[15] S. Yamamoto, X. Mao, and A. Imamiya, "Enhanced LIC pencil filter," in Proc. of the ICCGIV 04, pp. 251-256, 2004. Article (CrossRef Link)
[16] S. Yamamoto, X. Mao, and A. Imamiya, "Colored pencil filter with custom colors," in Proc. of Pacific Graphics 04, pp. 329-338, 2004. Article (CrossRef Link)
[17] Mao, X., Tsuji, K., Yamamoto, S., and Imamiya, A., "Colored pencil filter using LIC," in Proc. of Computer Graphics and Imaging, 2004. Article (CrossRef Link)
[18] K. Murakami, R. Tsuruno, and E. Genda, "Multiple illuminated paper textures for drawing strokes," in Proc. of Computer Graphics International 05, pp. 156-161, 2005. Article (CrossRef Link)
[19] H. Lee, S. Kwon, and S. Lee, "Real-time pencil rendering," in Proc. of NPAR 06, pp. 37-45, 2006. Article (CrossRef Link)
[20] K. Melikhov, F. Tian, X. Xie, and H. S. Seah, "DBSC-based pencil style simulation for line drawings," in Proc. of ICGRD, pp. 17-24, 2006. Article (CrossRef Link)
[21] D. Xie, Y. Zhao, D. Xu, and X. Yang, "Convolution filter based pencil drawing and its implementation on GPU," LNCS Vol. 4847, pp. 723-732, 2007. Article (CrossRef Link)
[22] Y. Kim, J. Yu, H. Yu, and S. Lee, "Line-art illustration of dynamic and specular surfaces," ACM Trans. on Graphics, Vol. 27, No. 5, Article No. 156, 2008. Article (CrossRef Link)
[23] Z. AlMeraj, B. Wyvill, T. Isenberg, A. Gooch, and G. Richard, "Automatically mimicking unique hand-drawn pencil lines," *Computers & Graphics*, Vol. 33, No. 4, pp. 496-508, 2009. Article (CrossRef Link)
[24] H. Kang, S. Lee, and C. Chui, "Flow-based image abstraction," *IEEE Trans. on Visualization and*

Computer Graphics, vol. 15, no. 1, pp. 62-76, 2009. Article (CrossRef Link)
[25] Y. Kwon and K. Min, "Texture-based pencil drawings from pictures," *Communications in Computer and Information Science*, Vol. 206, pp. 70-77, 2011. Article (CrossRef Link)
[26] H. Yang and K. Min, "Feature-guided convolution for pencil rendering," *KSII Trans. on Internet and Information Systems*, Vol. 5, No. 7, pp. 1311-1328, 2011. Article (CrossRef Link)
[27] H. Matsui, J. Johan, and T. Nishita, "Creating colored pencil images by drawing strokes based on boundaries of regions," in *Proc. of CGI 05*, pp. 148-155, 2005. Article (CrossRef Link)
[28] D. Xie, Y. Xuan and Z. Zhang, "A Colored pencil-drawing generating method based on interactive colorization," in *Proc. of ICCCIE*, pp. 166-169, 2010. Article (CrossRef Link)
[29] M. Hata, M. Toyoura and X. Mao, "Automatic generation of accentuated pencil drawing with saliency map and LIC," *Visual Computer*, Vol. 28, pp. 657-668, 2012. Article (CrossRef Link)
[30] M. Wang, R. Hong, X. Yuan, S. Yan and T. Chua, "Movie2Comics: Towards a lively video content presentation," *IEEE Trans. on Multimedia*, Vol. 14, No. 3, pp. 858-870, 2012. Article (CrossRef Link)
[31] D. Xie, H. Hu and Z. Zhang, "An Improved method for generating colored pencil drawing," *Advanced Materials Research* Vol. 433-440, pp. 1555-1560, 2012. Article (CrossRef Link)
[32] C. Lu, L. Xu, J. Jia, "Combining sketch and tone for pencil drawing production," in *Proc. of NPAR* 2012, 2012. Article (CrossRef Link)
[33] H. Yang, Y. Kwon, and K. Min, "A stylized approach for pencil drawing from photographs," In *proc. of EGSR 2012*, pp. 1471-1480, 2012. Article (CrossRef Link)

Fig. 19. Results

(a) (b) (c)

Fig. 20. Results