15th IAPR International Conference on Machine Vision Applications (MVA)
Nagoya University, Nagoya, Japan, May 8-12, 2017.

Automatic Pencil Sketch Generation by using Canny Edges

Ryota OKAWA Hiromi YOSHIDA Youji IIGUNI
Graduate School of Engineering Science, Osaka University;
1-3 Machi-kane-yama, Toyonaka, Osaka 560-8631, JAPAN
okawa@sys.es.osaka-u.ac.jp
{ yoshida, iiguni }@sys.es.osaka-u.ac.jp

Abstract

This paper presents a system that automatically converts 2D raster images to sketch style. The proposed method first extracts edges at different resolutions. Then, these shapes and brightness are varied and merged. This process expresses trial and error in the actual sketch. Experimental results showed that the proposed method produces images with natural appearance.

1. Introduction

In recent years, many NPR (Non-Photorealistic Rendering) methods have been proposed. NPR is a technique for generating images with artistic expressions, not realistic representation. Since artistic images are visually appealing and impressive, NPR is applied to advertisements, entertainment such as animation and video games. Among the art style, sketch painting is a style that everyone familiar with and has a unique softness. Sketch-like NPR methods take 3D model [1], 2D vector image [2], or 2D raster image [3,4,5] as input. Although 3D models and 2D vector images contain rich information on object shapes, it is relatively difficult to prepare these. Therefore, it is required to generate sketch-like images from 2D raster images.

For research focusing on line drawings, there are research by Son et al. [3] and research by Lu et al. [4]. Son et al. proposed a method for extracting perceptually important lines by constructing likelihood of true contour and tracking the ridges. Lu et al. focused on dividing into short strokes when humans draw long lines and proposed an approach that imitated them.

In this paper, we propose a system that automatically converts 2D raster images to sketch style. The proposed method distorts the edge of the input image with several patterns, adds fluctuation of brightness, and combines them. This process imitates the variation in line strength and shape seen in handwritten sketches.

2. Pencil Sketch

Since it is difficult to draw accurate lines, people draw lines with trial and error. In particular, the important contours are drawn several times with slightly distorted lines [1]. In addition, thickness and density of the lines vary. Lu et al. [4] reported that long lines are drawn divided into short strokes. In summary, the following features can be seen in hand drawn sketches:

A) Shape of lines are shaken
B) Variation in line brightness
C) Main contours are drawn with multiple lines
D) Long lines are drawn divided into short strokes

An example of actual sketch is shown in Figure 1. We can see the above features in the image.

Figure 1: An example of hand-written sketch

3. Proposed Method

In the proposed method, the edges of the input image are distorted by several patterns, and the variation of the blackness is added before combining. An overview of proposed method is shown in Figure 2. Firstly, edges are extracted from an image. The main edges are distorted strongly, and the texture edges are weakly distorted. Next, the brightness of each layer is varied. Then synthesize them and apply the texture. Less important lines are drawn only once and the main contour lines are drawn 2-3 times.

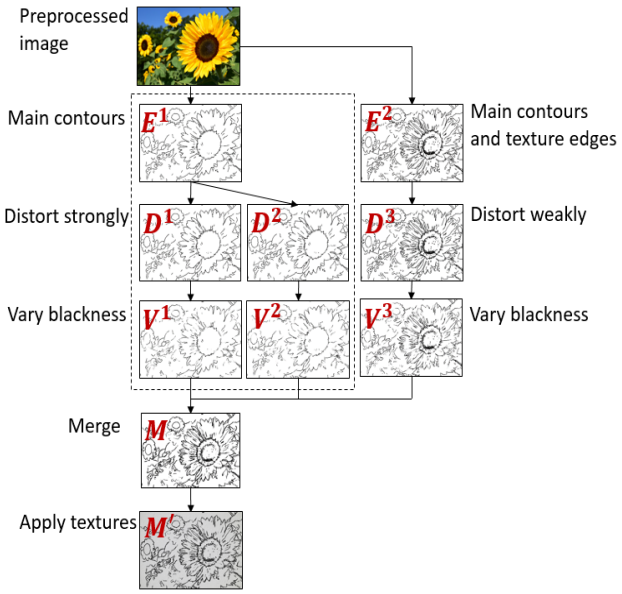


Figure 2: An overview of the proposed method

3.1 Pre-processing

We applied median filter and bilateral filter [6] to the input image as a pre-process. This process has two aims: The first aim is to reduce noise in the image. The second aim is to smooth texture edges while preserving the main outline. This makes it easier to extract main contours in the next step. We applied median filter with 3×3 window. The standard deviation of the color space and the coordinate space of the bilateral filter are 30 and 5, respectively.

3.2 Contour extraction

In hand-drawn sketches, main contours are drawn several times, but texture edges are drawn only once. From such observations, we generate two images, the edge image E^1 which including only the main outline and the edge image E^2 which including the texture edges. These images are processed separately in later process.

We use Canny operator [7] as an edge detector. By setting the standard deviation and the thresholds appropriately, we can extract only main edges. Also, the Canny edge has a nearly constant line width and is easy to handle. If the input is a colored image, use the color version of the Canny edge [8]. The gradient magnitude used for calculating the Canny edge is obtained by the following equation:

$$\nabla_x I \coloneqq \mathrm{MaxAbs}\{\,\mathrm{Sobel}_x[R], \mathrm{Sobel}_x[G], \mathrm{Sobel}_x[B]\,\} \quad (1)$$

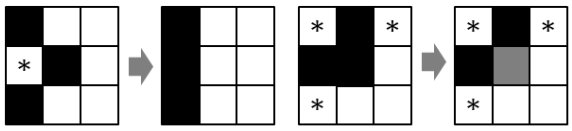
$$\nabla_y I \coloneqq \mathrm{MaxAbs}\{\,\mathrm{Sobel}_y[R], \mathrm{Sobel}_y[G], \mathrm{Sobel}_y[B]\,\} \quad (2)$$

$$\|\nabla I\| \coloneqq \frac{1}{\sqrt{2}}\sqrt{(\nabla_x I)^2 + (\nabla_y I)^2} \quad (3)$$

Here, Sobel $[\cdot]$ is a Sobel operator, and MaxAbs $(\cdot)$ is a function that returns the maximum absolute value among arguments. We can also use FDoG filter [9] instead of Sobel filter to compute Canny edges.

In order to simplify edges, simple mask processing was performed on the detected binary Canny edges. The mask is shown in Figure 3. * Represents an arbitrary pixel value. We also apply the masks which rotated Figure 2 by 90, 180 and 270 degrees. The unnatural wiggling of the pixel is eliminated by the mask process showed in Figure 3(a). By mask processing showed Figure 3(b), lines becomes thinner and anti-aliasing is applied to the corner. The effect of mask processing is shown in Figure 4. It can be confirmed that the lines are smoothed by the mask processing.

We use Canny operator with $\sigma = 2.0$ as main contour E^1. $\sigma = 0.6$ is used as the edge image E^2 with texture. Figure 5 shows the results of edge detection. Figure 5 (a) is the input image, (b) is the Canny edge E^1 of the main contour, and (c) is the Canny edge E^2 including the texture.



(a) Simplification (b) Anti-aliasing

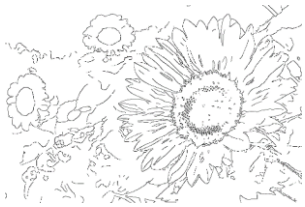
Figure 3: Masks

(a) without mask processing (b) after mask processing

Figure4: An example of mask processing

(a) A preprocessed image

(b) Main contours (E^1)



(c) Main contours and texture edges (E^2)

Figure 5: Canny edges

3.3 Contour distortion

In this section, we describe a method to distort lines. This process imitates the feature B described in Section 2. Although there is a study that adds fluctuation by sinusoid function to vectorized strokes [1], it is difficult to vectorize raster images and apply these methods. Therefore, in the proposed method, we don't vectorize the line but distort the line in the raster form. Let T_x and T_y are the movement amounts of the pixels in the x and y directions, respectively. Distorted edge D is computed by the following equation:

$$D(\mathbf{p}) = E\left(x - T_x(\mathbf{p}), y - T_y(\mathbf{p})\right) \quad (4)$$

In many cases, the referenced coordinates are non-integers. In that case, the pixel value is interpolated by bicubic interpolation. The vector fields T_x, T_y are generated by the sum of K Gaussian kernels:

$$T_x(\mathbf{p}) = \sum_{i=1}^{K} u_i \exp\left(-\frac{\|\mathbf{p} - \mathbf{p_i}\|^2}{2\sigma^2}\right) \quad (5)$$

$$T_y(\mathbf{p}) = \sum_{i=1}^{K} v_i \exp\left(-\frac{\|\mathbf{p} - \mathbf{p_i}\|^2}{2\sigma^2}\right) \quad (6)$$

Since the generated vector field is the sum of the Gaussian kernel, it is smooth. The kernel center p_i are set randomly. The weights u_i and v_i are sampled from uniform distributions. For major edges, we set σ and weight large. For texture edges, we set σ and weight smaller.

We denote images that distorted the main edge E^1 in two ways as D^1, D^2. Let D^3 be the image distorting the edge E^2 including the texture. Figure 6 shows an example of a distorted edge.

Figure 6: An example of line distortion (black lines: E^1, red lines: D^1)

3.4 Blackness variation

In this section, we describe a method to vary blackness of lines. The purpose of this process is two-fold: one is to break long lines by saturating the luminance values and divide them into short strokes (the feature D described in Section 2). The other is to reproduce the fluctuation of the strength of the line drawn by a person by the weak fluctuation of blackness (the feature B in Section 2). Generate a map F of intensity of luminance fluctuation by the following equation:

$$F(\mathbf{p}) = \sum_i w_i \exp\left(-\frac{\|\mathbf{p}-\mathbf{p}_i\|^2}{2\sigma^2}\right) \quad (7)$$

Centers of kernel $\mathbf{p}_i$ are set randomly. In order to make the distance between centers more than a certain value, we use Poisson Disk Sampling (PDS)[10,11] to placing centers of kernel. The weights w_i are sampled from uniform distributions. F is used to vary the luminance of the distorted edge image $D \in [0,255]^N$ by the following equation:

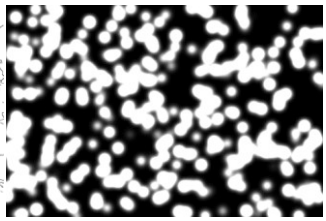
$$V(\mathbf{p}) = \mathrm{MAX}\big(0, D(\mathbf{p}) - F(\mathbf{p})\big) \quad (8)$$

An example is shown in Figure 7. Let V^1, V^2, V^3 are blackness fluctuated image of D^1, D^2, D^3, respectively. Let M is the merged image of them:

$$M := V^1 V^2 V^3 \quad (9)$$

This merging process is imitation of the feature C described in Section 2. An example is shown in Figure 8. Figure 8(a) (b) are main contours, (c) is main contours and texture edges, and (d) is the merged image of them.

(a) D^1



(b) F

(c) V^1

Figure 7: Blackness variation

3.5 Texture generation and applying

As a final step, we apply the pencil texture to the generated image. Hereinafter, a simple pencil texture generation method is described. It is an easy method but can generate texture of sufficient quality.

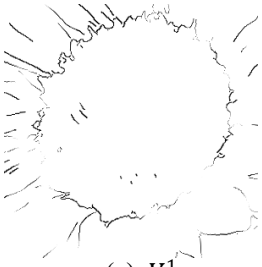
The pencil texture T is generated by the following equation:

$$T_1(x,y) = \text{sampled from } U(0,255) \quad (10)$$

$$T_2(x,y) = \begin{cases} 255, & \text{with probability } 0.5 \\ \text{sampled from } U(0,255), & \text{with probability } 0.5 \end{cases} \quad (11)$$

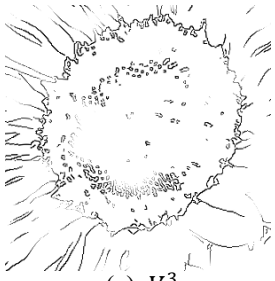
$$T(y,x) = 255 - \big(T_1(x,y) + T_2(x,y)\big) \quad (12)$$

Figure 9(a) shows the generated pencil texture.

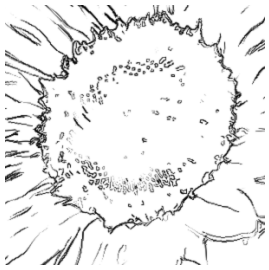


(a) V^1

(b) V^2



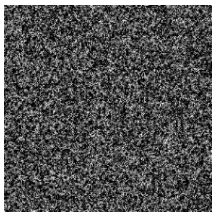
(c) V^3

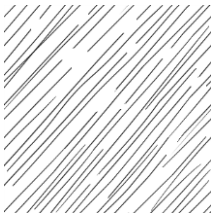


(d) M

Figure 8: An example of variation of blackness

We use a stripe pattern texture for imitation of rough painting. Figure 9(b) shows an example of stripe pattern texture. The input image is binarized by discriminant analysis method [12] and a stripe pattern is applied to the black region.





(a) A generated pencil texture (b) A stripe pattern

Figure 9: Examples of texture

4. Experiments

We compiled and executed the program in Visual C ++. The specifications of the PC used in the experiment is Intel Core-i5 3.20 GHz CPU and memory 8.00 GB. The execution time was about 2.5 seconds for the image of size 960 ×480. It takes about 10 seconds if we use FDoG-based Canny edges.

Figure 10 shows comparison to Lu et al. [4]. The result of [4] looks detailed drawing. On the other hand, our result looks quite rough sketch. Figure 11 shows comparison to Son et al. [3]. In the results of [3], the main contours are drawn only once. On the other hand, our result contains multiple lines for main contours.

(a) Result of Lu et al. (b) Our result (c) Our result (FDoG-based)

Figure 10: Comparison to Lu et al. [4]

(a) Result of Son et al. (b) Our result (c) Our result (FDoG-based)

Figure 11: Comparison to Son et al. [3]

5. Conclusion and Future Works

We proposed a system that automatically converts 2D raster images to sketch style. Our method consists of simple filtering processes. Experimental results showed that our method reproduces features of handwritten sketch well. In the future, we would like to use Saliency and apply our method to video sequences.

References

[1] H. Lee, S. Kwon and S. Lee, "Real-time pencil rendering," NPAR, pp.37—45, 2006.

[2] M. Hagiwara, K. Ushida and S. Tsurumi, "Generating Sketch-like Images from Vector Images Based on the Modeling Approach of Drawing", The Journal of The Institute of Image Information and Television Engineers, Vol. 64, No. 9, pp.1385—1388, 2010.

[3] M. Son, H. Kang, Y. Lee and S. Lee, "Abstract line drawings from 2d images," Pacific Conference on Computer Graphics and Applications, pp.333—342, 2007.

[4] C. Lu, L. Xu and J. Jia, "Combining sketch and tone for pencil drawing production," NPAR '12 Proceedings of the Symposium on Non-Photorealistic Animation and Rendering, pp.65—73, 2012.

[5] X. Mao, Y. Nagasaka and A. Imamiya, "Automatic generation of pencil drawing from 2D images using line integral convolution," Proc. of the 7th International Conference on Computer Aided Design and Computer Graphics (CAD/GRAPHICS'01), pp.240—248, 2001.

[6] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," ICCV '98 Proceedings of the Sixth International Conference on Computer Vision, pp.839– 846, 1998.

[7] J. Canny, "A Computational Approach to Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 8, Issue 6, pp.679—698, 1986.

[8] A. Koschan and M. Abidi, "Detection and classification of edges in color images," Signal Processing Magazine, Special Issue on Color Image Processing, Vol. 22, No. 1, pp. 64—73, 2005.

[9] H. Kang, S. Lee and C.K. Chui, "Coherent line drawing," Proceedings of the 5th international symposium on Non-photorealistic animation and rendering, pp.43—50, 2007.

[10] R. Cook, "Stochastic sampling in computer graphics", ACM Transactions on Graphics (TOG), Vol. 5, Issue 1, pp.51—72, 1986.

[11] R. Bridson, "Fast Poisson Disk Sampling in Arbi-trary Dimensions", SIGGRAPH '07 ACM SIGGRAPH 2007 sketches, No. 22, 2007.

[12] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems, Man and Cybernetics, Vol. 9, Issue 1, pp.62—66, 1979.