AUTOMATIC GENERATION OF PENCIL-SKETCH LIKE DRAWINGS FROM PERSONAL PHOTOS

Jin Zhou and *Baoxin Li*
Center for Cognitive Ubiquitous Computing
Department of Computer Science and Engineering
Arizona State University, Tempe, AZ, U.S.A.
{jinzhou, baoxin.li}@asu.edu

ABSTRACT

In this paper, we present an algorithm for automatically generating pencil-sketch like drawings from personal photos. On top of the core step of gradient computation, some proper transformations are introduced to achieve the best visual effects. It is found that the popular Sobel operator and Laplacian operator are both not desirable for this task, and thus we propose a computationally simple algorithm for gradient estimation. Experimental results show that the proposed method can generate visually appealing pencil-sketch like images from personal photos.

1. INTRODUCTION

Pencil sketch drawings are a very popular form of art. In a typical pencil sketch image, only the most characteristic lines of the underlying subject are drawn, using a dark color (pencil) on a white background (paper). Also, certain degree of variation in the darkness of the pencil is typically used to depict various types of transitional boundaries (e.g., edges) and shadows in the original scene. Pencil sketch drawings are in some sense similar to pen-and-ink drawings. In computer graphics, many pen-and-ink rendering techniques have been proposed [1-5], which generate pen-an-ink images by 3D geometry or through human interaction. Recently, an automatic method [6] was reported, which generates pen-and-ink drawings directly from photos. Advantages of this type of automatic approaches, such as no need for a 3D model and no requirement for human interaction, make them appealing in various applications targeting at the general public.

Unfortunately, directly applying the method of [6] to the generation of pencil sketches is not practical. First of all, a fundamental assumption in [6] is that the pen-and-ink drawings are binary. And thus the major focus there is to represent the texture information via binary pen strokes. This is not necessarily true and useful in the case of pencil sketch, which may allow (ideally) some degree of variation in the darkness of the pencil color (we confined ourselves to black pencils only). In particular, there are no examples of human photos given in [6]. It is very likely that in the case of human photos, such as portraits, binary texture information alone is unlikely to capture the essence of the photo. In addition, the method of [6] does not seem to keep the continuity of edges, as evident in the scenery-only examples listed in the paper, which may be important in processing portrait-type photos.

In this paper, we focus on photos with human subjects. Unlike in the case of scenery images used in [6], where people may simply judge if the generated image is beautiful, the generation of a human photo demands the maintaining of the key features of the image, otherwise the obtained image may not resemble the original, rendering the generated sketch undesirable. We also believe that processing human photos may be deemed more appealing to a user as people may be more interested in converting a photo contain themselves.

In Section 2, we introduce a general method for converting a photo into a pencil sketch drawing, illustrated with examples. Then, in Section 3, we experiment the proposed method using different gradient estimation methods, and then propose a computationally simple algorithm, which generates the most appealing visual results. Experiments with various images are then presented in Section 4.

2. PROPOSED METHOD

2.1 Problem Description

The essential properties of a pencil sketch image include:

1. Objects are depicted by contours, which are long and significant edges.
2. Edges are drawn in dark, with the darkness roughly proportional to the local gradient, while the background is kept purely white.
3. Textured regions can be depicted by a collection of short lines/edges.

Therefore, if we want to generate pencil sketch image from a photo, first we need to extract edges from the image, and then draw the edges in black stroke, with the darkness roughly proportional to the local gradient. If the edge detector is so designed that it can extract short edges in textured regions, then the above point 3 will be handled naturally.

Note that, while the above task looks simple and similar to edge detection, it is not a pure edge detection problem, as illustrated in Fig. 1, where we apply the well-known Canny's edge detector [7] to an input image (a) to get the edge map (b). Obviously, the image in (b) does not constitute what people would normally call a pencil sketch drawing.

2.2 The Proposed Four-step Approach

From the example in Fig. 1, it is obvious that, to achieve the desired visual effects of an actual pencil sketch as summarized above, more processing is needed. In the following, we propose a four-step method for this purpose, as outlined in Fig.2. The method is first described in a general setting with the Laplacian operator as an example. We will then explore more for a better gradient estimation algorithm in Sect. 3.

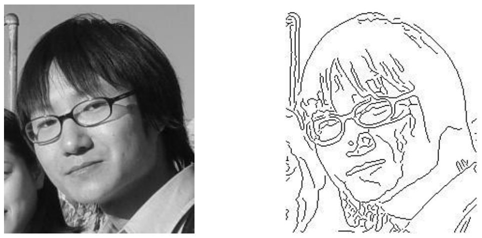


Figure 1. Edge detection does not generate a pencil-sketch type of drawing. (a) Original image. (b) Detected edges.

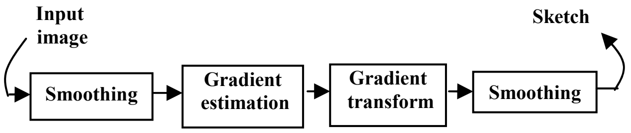


Figure 2. Outline of the steps in the proposed method.

Step 1: Smooth the input picture

In general, an unprocessed image may contain excessive noise, which reacts strongly to the subsequent gradient estimation. Therefore, as the first step, the input image is smoothed by a Gaussian low-pass filter, just as the pre-processing of the Canny's edge detector.

Step 2: Gradient estimation

With the noise subdued by the smoothing step, the next task is to detect points of significant gradient (roughly speaking, the edges). As noted above, not only the locations of the edges but also the gradient at those locations should be kept. For illustration purpose, we use the following Laplacian operator for gradient estimation. (The performance will be discussed in Sect. 3.)

0	-1	0
-1	4	-1
0	-1	0

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

That is, the output after applying the mask (left) to an image block shown on the right is given as $g = 4z_5 - (z_2 + z_4 + z_6 + z_8)$.

Since the Laplacian operator produces both positive gradient and negative gradient, to roughly detect the edge, we can simply keep either of the two gradients (a more accurate way is to detect the zero-crossing). We choose the negative gradient. To this end, the following simple thresholding is used:

$$g = \begin{cases} |g| & \text{if } g < 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Step 3: A proper gradient transform

To achieve the objective of linking darker pencil color to edges of larger gradient, we apply the following transform,

$$g = \begin{cases} 120 - g & \text{if } g > 0 \\ 255 & \text{otherwise} \end{cases} \tag{2}$$

where *120* is an empirically-chosen parameter, which can be user-adjustable in the software. With the image of Fig. 1(a) as the input, Steps 1 and 2 will generate the image given in Fig. 3 (a).

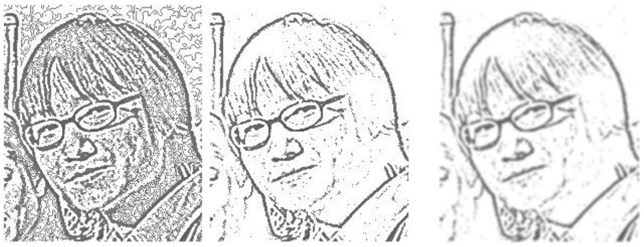


Figure 3. Intermediate results of the steps.

Note that, even with the smoothing operation in Step 1, one can still find that, from Fig. 3(a), there are excessive noisy details still visible in the image, making it dissimilar to a pencil sketch. This is especially the case since we used the Laplacian operator, which is very sensitive to noise. To get a succinct pencil sketch image, we need to eliminate most of the details. To this end, in practice we threshold the gradient image before applying the transformation of Eqn. (2). Thus in Eqn. (1), g will be set to zero if $|g|$ is less than a threshold. Using a threshold *4* gives us the image of Fig. 3 (b).

Step 4: Final smoothing for enhanced visual effects

The results from the previous three steps typically give the visual effect of a pencil sketch, except that there may be many broken contours that appear to be unnatural. Also, the thresholding step may make the difference between the contours and the background too abrupt, rendering the contours less similar to a pencil stroke. To alleviate these problems, we adopt another smoothing step to further blend the contours with the background and to link the broken contours. The effect of this step is illustrated in Fig. 3(c).

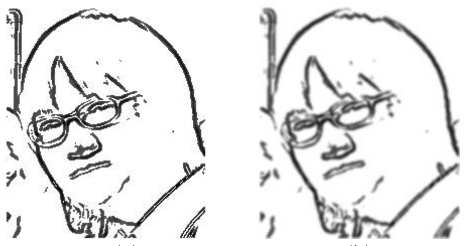
3. A BETTER GRADIENT ESTIMATION METHOD

The core step of the proposed method in Sect. 2 is the gradient estimation. In this section, we show, with experiments, that both of the commonly-used Sobel operator and Laplacian operator are not very effective for our purpose. Then we discuss what consists of a good gradient estimator for the purpose, and propose an algorithm for this task.

The examples shown in Fig. 3 are based on the Laplacian operator. Here we give an example of using the Sobel operator to perform the same task. The results are given in Fig. 4. Note that, although Fig. 4 (a) seems very pencil-drawing like, the details of the original image (e.g. the eyes and the mouth) are damaged, rendering the final results less appealing than that of Fig. 3(c).

3.1 The Proposed Gradient Estimator

In estimating the gradient from the original image, it would be desired to consider a larger support in order to extract true contours rather than noisy segments. One of the drawbacks of the Laplacian operator as well as other conventional edge detectors is that only a small neighborhood is used to compute the gradient. To address this issue, we propose a simple algorithm which makes use of a larger support in gradient computation. For a given point, along the horizontal axis, this method first searches for a local maximum in the 1-D gradient in both directions (towards the left and towards the right of the current point). The two gradients from the above are added to get one horizontal gradient for that point. The process is repeated for the vertical axis, and the larger of the two gradients will be assigned as the final gradient of the current point. With the input image smoothed first, this search effectively utilizes a larger support in gradient computation.



(a) (b)

Figure 4. (a) Results of the Sobel operator, after inversion and thresholding.. (b) After applying the transformation of Eqn. (2).

Formally, we define the gradient between two points p_1 and p_2 as

$$g(p_1, p_2) = (v_1 - v_2) / \text{Distance}(p_1, p_2)$$

For each point p, we define four search directions, to its left, right, up, and down, respectively. For each direction, we use a greedy search algorithm to search for the local maximum in gradient for point p, as illustrated in the following pseudo code:

```
Function g_d( $p$ )  // $d \in$ {left, right, up, down}
  maxg = 0;
  For each point $p_i$ in the direction $d$ ,from near to far:
      $g = g(p, p_i)$;
      if (|$g$| > |maxg|) maxg = $g$;
      else return maxg;
```

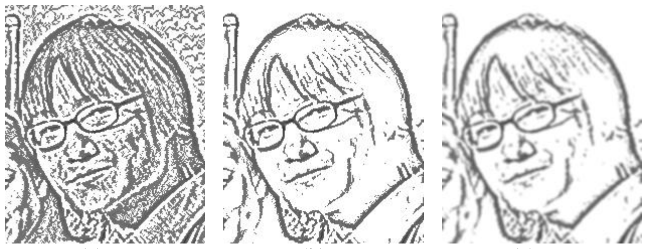
On each of the horizontal (x) and vertical (y) axes, the gradient for point p is obtained by a simple addition:

$$g\_x(p) = g\_left(p) + g\_right(p)$$
$$g\_y(p) = g\_up(p) + g\_down(p)$$

and the final gradient for point p is given by

$$g(p) = \begin{cases} g\_x(p) & \text{if } |g\_x(p)| \geq |g\_y(p)| \\ g\_y(p) & \text{otherwise} \end{cases}$$

With the above simple algorithm, we process the image of Fig. 1(a) again, and Fig. 5 shows the obtained results, corresponding to those of Fig. 3. A close look reveals that, in the final results in Fig. 5, the lines are darker and thicker by the proposed gradient computation; also more true details are detected without adding too much noise. Overall, it was found that the simple gradient estimation method proposed above generates outputs that are visually more appealing. Additional examples of comparison between the different gradient estimation methods will be presented in the next section.



(a) (b) (c)

Figure 5. (a) Gradient estimation using the proposed method. (b) After thresholding. (c) Final results; Compare with Fig. 3(b) and Fig. 3(c).

4. EXPERIMENTAL RESULTS

4.1 Comparison of Different Gradient Estimation Methods

In Fig. 6, we present another example for the comparison between results of the proposed method (Sect.2) in combination with different gradient estimation methods. Apparently, this test image is more complex, which contains not only human faces, but also entire human bodies with some complex background. From Fig. 6, it can be seen that the proposed gradient estimation method produces the most visually appealing results, with more details. For example, the face and elbows of the person on the left is very obvious in (b), while it is unclear or partially missing in (c).

4.2 More Examples from the Proposed Method

We tested our method with more than 100 images containing human faces plus some other types of images. Some examples and the corresponding pencil sketch drawings are shown in Fig. 7. It can be seen that our method is good at not only human face images but also other types of images such as the architecture image. For a full-sized version of the examples presented here and for more examples of other images, please visit the following Web site:

http://www.public.asu.edu/~jzhou19/pencil.htm

5. CONCLUSION

This paper presents a method for automatically generating pencil sketch drawings from photos. A gradient estimation method has also been proposed, which was found to serve the purpose better than the conventional methods. Experimental results show that the proposed methods are suitable for face images as well as other types of images.

6. REFERENCES

[1] Georges Winkenbach and David H. Salesin. “Computer-generated pen-and-ink illustration”. In Andrew Glassner, editor, *Proceedings of SIGGRAPH’94*, pp. 91-100, Orlando, Florida, July 1994.

[2] John Lansdown and Simon Schofield. “Expressive rendering: A review of nonphotorealistic techniques”, *IEEE Computer Graphics and Applications*, 15(3), pp. 29-37, May 1995.

[3] Micheal P. Salisbury, Sean E. Andersion, Ronen Barzel, et al. “Interactive pen-and-ink illustration”. In Andrew glassner, editor, *Proceedings of SIGGRPATH’94*, pp. 101-108, Orlando, Florida, July 1994.

[4] Michael P. Salisbury, Micheal T. Wong, John F. Hughes, et al. “Orientable textures for image-based pen-and-ink illustration”. In T. Whitted, Editor, *Proceedings of SIGGRAPH’97*, pp. 401-406, Los Angeles, California, 1997.

[5] Oliver Deussen and Thomas Strothotte. “Computer-generated pen-and-ink illustration of trees”. *Proc. SIGGRPATH* 2000, pp. 13-18, New Orleans, Louisana, July 2000.

[6] Jiatao Song, Zheru Chi, Jilin Liu and Hong Fu, “Automatic generation of pen-and-ink drawings from photos”, *Proc. IEEE International Conf. on Image Processing*, 2004.

[7] J. Canny, A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 8, No. 6, Nov 1986.

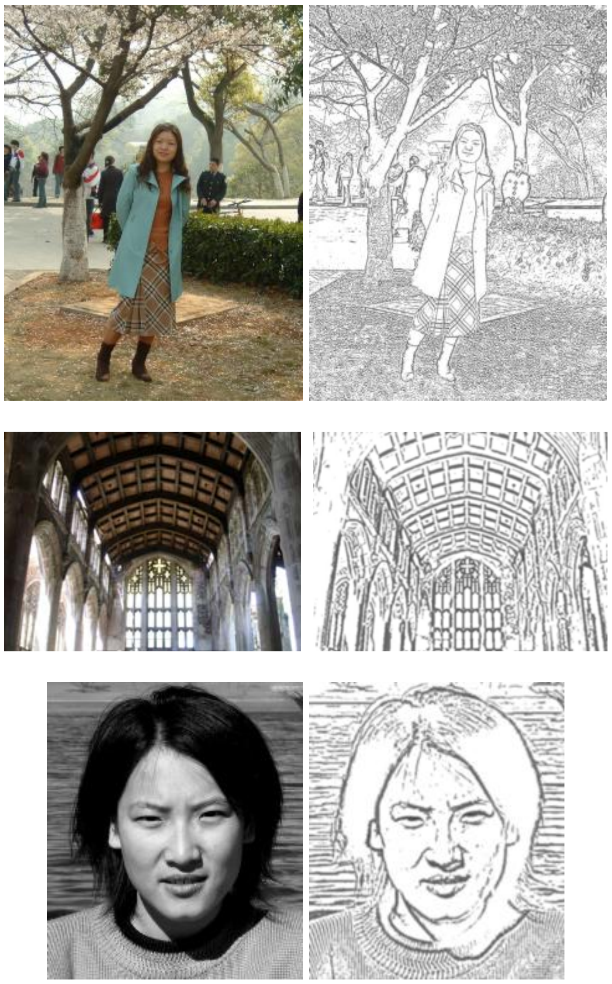
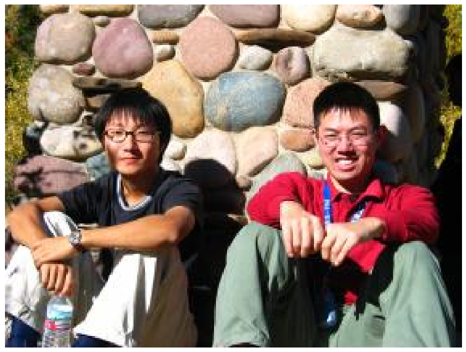
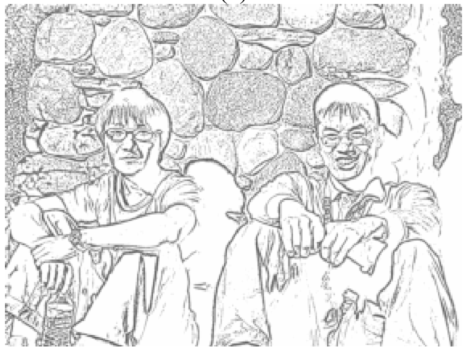


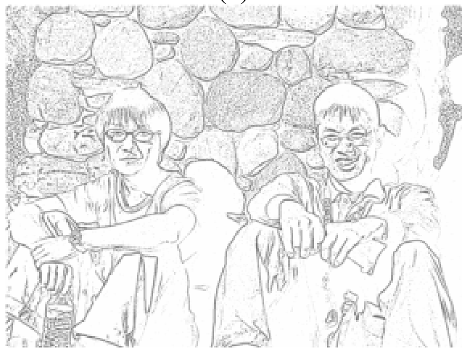
Figure 7. Other examples from the proposed method.



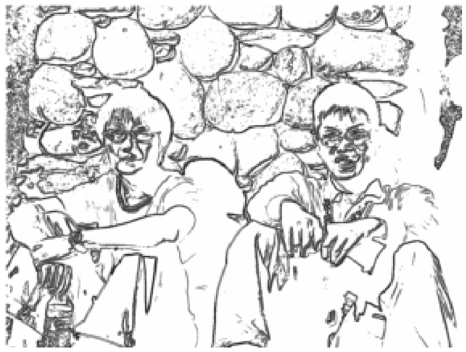
(a)



(b)



(c)



(d)

Figure 6. (a) Original image; (b) Results using the proposed gradient estimation method;. (c) Results based on the Laplacian operator; (d) Results based on the Sobel operator. Note that all the results are generated by the method proposed in Sect. 2, except that the gradient estimation is computed using different approaches.