Pen and Ink Pictures from Gray-Scale Photographs

G. Arroyo, D. Martín

Dpto. de Lenguajes y Sistemas Informáticos, University of Granada, Spain
arroyo|dmartin@ugr.es

Abstract

A new method for producing pen and ink pictures from gray-scale photographs is introduced in this article. It is based on path-finding and reconstruction of strokes. First we apply filters to extract contours from photographs which are converted to strokes, then we show how to modify these strokes in the same way an artist could do it. The strokes are not pixel-based but bspline curves, which are rendered simuling different kinds of brushes and papers.

Categories and Subject Descriptors (according to ACM CCS): 1.3.3 [Computer Graphics]: Non-Photorealistic Rendering

1. Introduction

In many books, the illustration are hand made, which is costly and complex. There is an increasingly interest in doing that in an automatic way from 3D models. In many cases, the construction of 3D models is a complex and lengthly process. Otherwise, photographs are very easy to obtain. So, we are interested in producing a completely automatic process, based on photographs, to obtain pictures that look like handmade ones.

As stroke simulation is almost done with deformable models of 3D brushes[1], and ink brush simulation for 3D models is just surprising[2, 3], we could think that the big problem is recognizing contours from a bidimensional image, and its transformation to strokes, like a human being could do.

2. Overview

The method applies 3 steps to obtain the final results:

1. Contours detection
2. Strokes creation
3. Strokes stylization

The main contributions of these paper are in the creation and stylization of strokes, because the contours detection is based in very common techniques.

Now, these steps are better explained.

2.1. Contour detection: Canny's Algorithm

The process to obtain the images is as follows. First of all, we smooth the original image, discretizing it with a Gaussian function given by expression:

$$S(x,y,\sigma) = ke^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Where σ is the standard deviation of the Gaussian function. Value given by k is useful to make all area of the function equal to 1, in this way we have all function values normalized (and we don't loose information), and its value is:

$$k = \frac{1}{2\pi\sigma^2}$$

Next, a convolution mask with discretized values is applied to the original image. Then we get the smooth image I.

We apply Canny's algorithm. Let $G_x(I)$ and $G_y(I)$ be the gradients in x and y from image I. We do a convolution for computing these gradients with discretized and normalized values of partial difference in x and y in the Gaussian function:

$$\frac{\partial S(x,y,\sigma)}{\partial x} = \frac{-x}{2\pi\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}} \;,\; \frac{\partial S(x,y,\sigma)}{\partial y} = \frac{-y}{2\pi\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

If we apply both masks to image I we get two news images: $I_x = G_x(I)$ and $I_y = G_y(I)$ which are used to calculate angles and directions of contours.

Let $M = G_x(I)^2 + G_y(I)^2$ be applied to every pixel, where M is the magnitude image, and let $D = atan(\frac{G_y(I)}{G_x(I)})$ be ap-

plied to every pixel, where D is the direction image. Finally, we can suppress from image M non-maximums values. For that, two threshold values given by t_l y t_h are defined. In this case, $\forall p_{x,y} \in M$ a new gradient image(called A) is obtained, with pixels that comply with $t_l < p_{x,y} < t_h$.

The less significant borders are suppressed in a process called hysteresis. The algorithm used is as follows:

For each point P_1 given by (x,y):

if $A(x,y)$ haven't visited yet:

- if $A(x,y) < t_l$ then:
  - $F(x,y) = non-border$
  - mark P_1 as visited
- if $A(x,y) > t_h$ then:
  - $F(x,y) = border$
  - mark P_1 as visited
  - follow the direction given by $D(x,y)$ from P_1 to both sides, while $A(i,j) > t_l$:
    - mark point P_2 given by (i,j)
    - mark P_2 as visited

The main problem is that contours with Y shape are usually splitted. But this is not a big problem because they can be recovered with the following step.

2.2. Stroke creation

The main problem is that Canny's algorithm only produces one pixel borders, while we are interested in obtain wider strokes.

Our algorithm for stroke creation is based on obtaining a list of points, which depend on the obtained contours.

The list is created introducing pixels that comply with a condition. The pixel P_i will be include if it is surrounded by less pixels than others, given a window with a size of t_v. The next pixel is chosen in the same way with an exception: it will be included in the list only if it is not in the list previously; and so on.

The algorithm is as follows:

1. $P_0 = Window(P, t_v)$
2. If P_0 doesn't exists end
3. Otherwise:
4. Get the nearest pixel P_{i+1} that is not processed
5. If P_{i+1} doesn't exists go to 1 (we have a new stroke), else take it.

Window is a function that chooses a pixel not chosen before and that is surrounded by less pixels than others into a window with a size given by t_v, which is the maximum distance allowed between this pixel and another one to make an unique stroke.

The main problem in this algorithm is based in randomness in the point P_0 for every stroke, producing that only very short strokes are obtained when they are only one actually. To avoid this, the algorithm does not stop in step 2, but the list is inverted where there are stocking pixels, and the process starts again from step 2 with P_0 as the first point to put into the path.

Then path is ordered in the same way that Canny's algorithm. From this point, we make a list with angles and another one with distances from one pixel to the following. In this way, every pixel depends on the previous one.

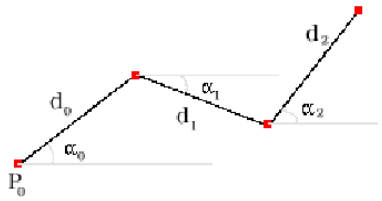

Figure 1: *Stroke path parameters and his representation. P_0 is the initial point of the stroke, while α_i and d_i are angles and distances between two consecutive points*

2.3. Stroke stylization

Once strokes are created, a process is applied based on knowledge of how artists draw, splitting very long strokes, and simuling brush drawing process. An artist almost never draws straight strokes, or strokes with perfect angles, but he/she usually draws with curve paths, opens or closes them, etc [5].

As a previous step, some characteristics of how an artist draws with pen and ink are commented:

1. He/she changes angles and size in strokes
2. He/she uses different sizes of brush
3. He/she uses different kinds of paper
4. He/she uses more or less number of stroke to do some picture

Changing the angles of strokes is trivial using our structure. It is easy to change angles given by α_i (see Fig. 1). These changes open or close the stroke from the origin. Also, some added noise in angles can change straight lines to curve strokes.

In the same way, changing size of strokes is easy too. When we decrease distances given by d_i (see Fig. 1), we will have got shorter stroke. Also, we can use noise to make strokes with different lengths, shorter or larger, in an arbitrary way.

Other changes are easy to obtain because every stroke belong to a list, which is made by points. We can split this list according to criteria like angle or stroke size. Also, angles

and position of every stroke can be modified, in a way it appears to be two differents strokes.

To obtain different sizes of brush we need define a new function w that takes the stroke as a parameter. This function mainly depends on stroke size, and it can be any, but the next one produce good results: $w(\phi) = \frac{\phi}{2} k_{max} \quad if \ \ \phi < \frac{1}{2}$ and $w(\phi) = (\frac{\phi}{2} + \phi^2) k_{max} \quad if \ \ \phi \geq \frac{1}{2}$, where ϕ is between 0 and 1, and k_{max} is maximal stroke width. This function defines how stroke begins and ends, and the width for every position in it.

We have provided another level of flexibility including the simulation of the paper absorption. For different kinds of paper absorption we use the function t, whose values will be used as inputs to control the alpha channel when we draw different strokes. In the same way, t can be take any value, although the next one produce good results:

$$t(\phi) = \frac{acos(\phi)}{\pi} k_{max}, \text{ where } k_{max} \in [0,1]$$

Let g_i be the output of the width function and let t_i be the output of the transparency function in a point given by i. Then, to draw a stroke two points translated to p_i and p_{i+1} respectively are generated, and rotated $\frac{\pi}{2}$ radians in relation to α_i (called p_j and p_{j+1} respectively), generating a polygon formed by following points: p_i, p_{i+1}, p_{j+1} and p_j.

Only those strokes which are longer than a threshold value, given by l, are evaluated.

But even after correcting angles and distances, it is difficult to obtain a good simulation of a hand-made drawing, because most angles are right angles or corners. To avoid that the strokes are obtained from Bèzier curves, whose control points are determined by lists of angles and distances with an origin point (see Fig. 1).

Output are polygons which are blended in the way of several strokes with changes in angles and distances and Bèzier curves, which are applied with any width function, we have tested many of them and the following produces good results:

$$w(\phi) = \frac{acos(\phi)}{\pi} k)$$

Finally, a smooth correction is done (simuling paper absorption), drawing a half-transparent picture with soft rotation angles (from $-\beta$ to β).

Several colours have been tested and the best results are obtained with the following RGBA vector: (0.2, 0.2, 0.3, $k_r t_i$), where $k_r = 0.6$ or $k_r = 0.8$, and t_i is the output of evaluating t at point i.

3. Results and Conclusions

Although it is no very usual to see pen and ink pictures of humans and life beings, we have tried our algorithm with organic models (see Fig. 3), getting pretty good results.

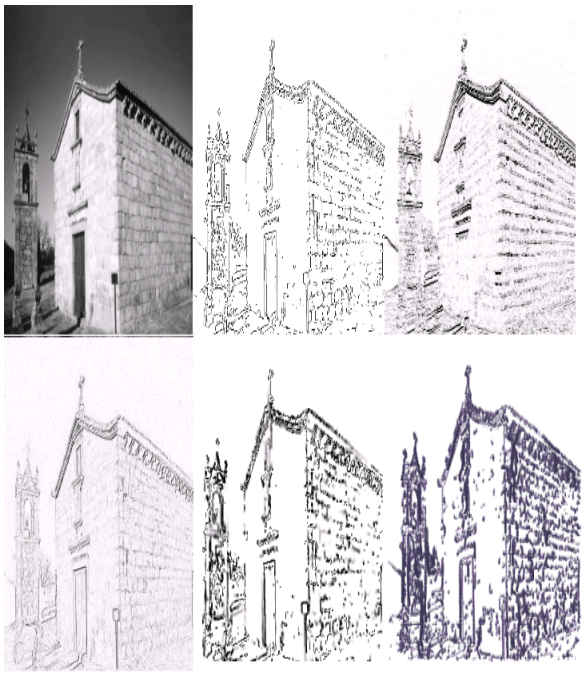

Figure 2: *From left to right and from top to bottom, comparison to original image with output images from: Canny's algorithm, Sobel's filter, Robberts's filter, Canny's algorithm with path-finding algorithm, and finally, our algorithm with followings parameters:* $\sigma = 1$, $t_l = 0$, $t_h = 0.1$, $t_v = 4$, $l = 3$, $k_r = 0.6$ *and as width function* $f_w(x) = \frac{acos(x)}{\pi k_{max}}$, *and as transparency function* $f_t(x) = \frac{acos(x)}{\pi k_{max}}$ *in a kind of stroke and* $f_t(x) = x k_{max}$ *in the other.*

Parameters σ, t_l and t_h affect basic sketch from the picture, a hight value of σ will smooth image in the beginning of the algorithm. A lower value of t_h or higher of t_l will soil picture with more details. But, mainly parameters in basic lines are t_v y l, because this values affect to decision of what lines are valid and what are only points.

In the process of testing, we have seen that some values produce better results than others. These values are shown in the examples.

We have used two kinds of strokes for all images, one use a Bèzier curve and other do minimal changes in angles to the original path (see Fig. 4).

References

1. B. Baxter, V. Scheib, M.C. Lin, D. Manocha "DAB: Interactive Haptic Painting with 3D Virtual Brushes", *Proceedings of SIGGRAPH 01, Computer Graphics Proceedings, Annual Conference Series, pp. 461-468, 2001* 1

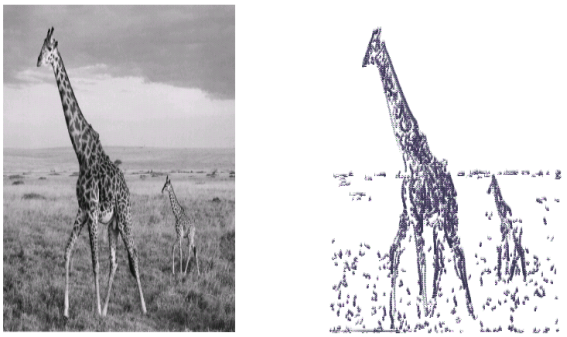

Figure 3: *Our algorithm applied to an organic model with the following parameters:* $\sigma = 1$, $t_l = 0$, $t_h = 0.2$, $t_v = 4$, $l = 6$, $k_r = 0.6$ *and as width function* $f_w(x) = \frac{acos(x)}{\pi k_{max}}$, *and for transparency function* $f_t(x) = \frac{acos(x)}{\pi k_{max}}$ *in a kind of stroke and* $f_t(x) = xk_{max}$ *in the other.*

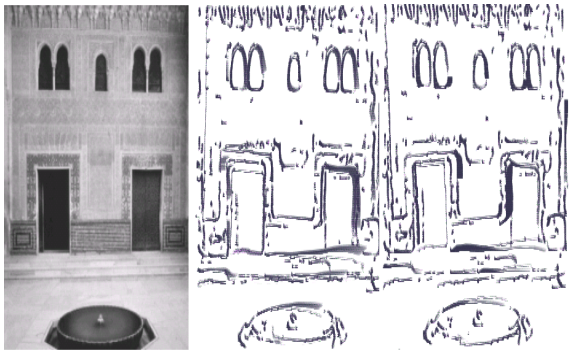

Figure 4: *We have changed parameters in Bèzier control points and some angles. Result is similar to change shadows in a picture. Parameters used in both images are:* $\sigma = 1$, $t_l = 0$, $t_h = 0.1$, $t_v = 4$, $l = 3$, $k_r = 0.8$

2. M. Salisbury, C. Anderson, D. Lischinsky, D.H. Salesin, "Scale-dependent reproduction of pen-and-ink illustrations", *Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, Annual Conferences Series, pp. 461-468, 1996* 1

3. M.P. Salisbury, M.Wong, J.F. Hughes, D.H. Salesin, "Orientable textures for image-based pen-and-ink illustration", *Proceedings of SIGGRAPH 97, Annual Conferences Series, pp. 401-406, 1997* 1

4. D. Martín, J.D. Fekete, J.C. Torres "Flattening 3D objects using silhouettes", *Eurographics 02, pp. 239-248, Saarbrucken, Germany, 2002*

5. A. l. Guptill, "Rendering pen and ink", *Watson-Guptil Pub., 1997* 2

6. D. Marr, "Vision - A Computational Investigation into the Human Represent", *W. H. Freeman & Co., September 1983*

7. J.A. Aznar, M. Moreno, "Simulación Computacional del Procesamiento Visual Biológico de Bajo Nivel", *http://www.ub.es/pbasic/Vision_Binocular.PDF, Valencia, 2001*

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

9. E. Davies, "Machine Vision: Theory", *Algorithms and Practicalities Academic Press, 1990, Chap 5.*

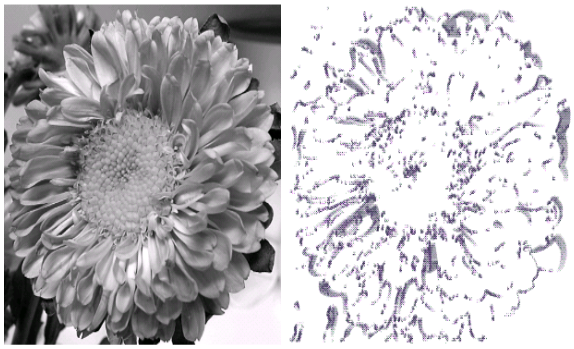

Figure 5: *For elements with too much details it is useful use* t_h *between 0.2 and 0.4, for this photograph we have used:* $\sigma = 1$, $t_l = 0$, $t_h = 0.2$, $t_v = 4$, $l = 3$, $k_r = 0.2$

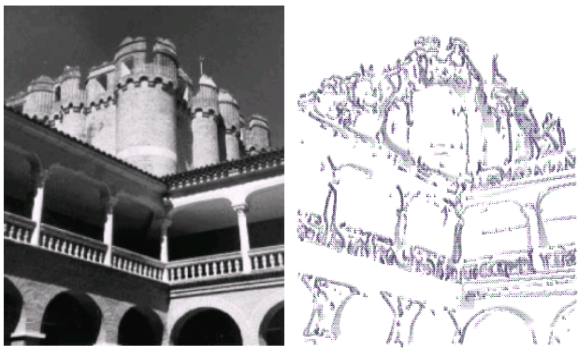

Figure 6: *Photograph with too much noise are smoothed because parameter* σ, *but details aren't supressed (thanks to parameter* t_h*), parameter used are:* $\sigma = 1$, $t_l = 0$, $t_h = 0.1$, $t_v = 4$, $l = 3$, $k_r = 0.2$ *width and transparency functions are same as before.*