Artistic representation of Ink and Paper Pictures from Gray-Scale Photographs

G. Arroyo, D. Martín

Software Engineering Department, University of Granada, Granada, Spain
arroyo|dmartin@ugr.es

Abstract

In this article we introduce a path-finding and reconstruction strokes algorithm applied as ink picture. Also, first we apply filters to photographs to extract characteristics from a scene, then we show how to modify strokes like an artist could do it. We define a stroke structure in which for every stroke we have vertexes and every vertex depends from the previous one. Finally we use smooth strokes with functions to simulate different kinds of brushes and papers.

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

1. Introduction

Book pictures illustration is a complex and completely manual technique at the present time, increasingly it is tried to do this kind of illustrations in an automatic way from 3D models. Paradoxical though it may seem, process is just manual when we make the model and it is desirable that to be completely automatic. For all that, we must try to generate this pictures from photographs.

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 true problem is recognizing contours from a bidimensional image, and his transformation to strokes like a human being could do it.

2. Overview and Related Work

First main idea is to make series of transformations from the original image in two main steps:

1. Contours detection
2. Strokes detection

Once stroke are detected, we do a detection process based on artist drawing way, it splits very long strokes, simulating brush drawing process. A human being almost never draws straight strokes, or perfect angles, but he usually opens or closes them, in others words, he changes them.

2.1. Contour detection: Canny's Algorithm

First of all, we smooth origin image, for that, we discretize Gaussian function given by expression: $S(x,y,\sigma) = ke^{\frac{-(x^2+y^2)}{2\sigma^2}}$

Where σ is 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), his value is: $k = \frac{1}{2\pi\sigma^2}$

Next, we make a convolution mask with discretized values, and we applied to origin image. Then we get the smooth image I.

Next, we apply Canny's algorithm. Let $G_x(I)$ be a gradient in x from image I and let $G_y(I)$ be the gradient in y from image I. We do a convolution for calculating 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 im-

© The Eurographics Association 2003.

ages: $I_x = G_x(I)$ and $I_y = G_y(I)$, that we use them 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 magnitude image, and let $D = atan(\frac{G_y(I)}{G_x(I)})$ be applied to every pixel, where D is direction image. Finally, we can suppress from image M non-maximums values, for that, we define two threshold values given by t_l y t_h. In this case, $\forall p_{x,y} \in M$ we just get in a new gradient image (called A) pixels that comply with $t_l < p_{x,y} < t_h$.

We suppress less significant borders in a process called hysteresis. Algorithm we have used is the following:

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

Main problem is that contours with Y shape are usually splitted. But this isn't a big problem because we can recover them in a following step.

2.2. Stroke detection

Because Canny's algorithm permits to obtain only one pixel borders we can do an algorithm which permits getting larger than possible strokes. Our algorithm for stroke detection is mainly based in introduce pixel P_i into a list. This pixel is selected because it is surrounded by less pixels than others into a window with size given by t_v. Next pixel is similarly chosen with an exception, if it is just in the list it cannot be selected, but otherwise we introduce it into the list.

The first algorithm we could think about is the following:

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 we haven't taken yet
5. If P_{i+1} doesn't exists go to 1 (we have a new stroke), else take it.

Where *Window* is a function which chose a pixel hasn't been chosen yet and it is surrounded by less pixels than others into a window with size given by t_v, which is the maximum distance permitted between this pixel and other one to make an unique stroke.

Main problem in this algorithm come determined by random in the point P_0 for every stroke, in this way, we could get very short strokes when they are only one actually. To avoid this, we don't end algorithm in step 2, we invert the list where we are stocking pixels and we begin again from step 2 with P_0 as the first point 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 other with distances from one pixel to the following. In this way, every pixel depends from the previous one.

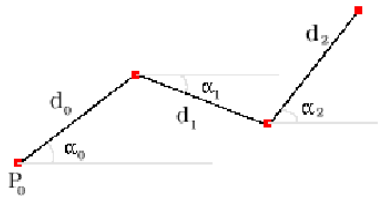

Figure 1: *Stroke path parameters and his representation. P_0 is the initial point of the stroke, while α_i and d_i are angles and distances between two consecutive points*

3. Stroke stylization

As we have said, human being never make perfect strokes, then we apply different modifications to strokes we get with filters. Next, let us try to understand how an artist draw with ink over paper:

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

Changing angles in strokes is trivial because structure we have got. It is easy to change angles given by α_i (see Fig. 1), these changes open or close the stroke from the origin. Also, noise in angles can change straight lines in curve strokes.

In the same way, changing size in stroke is trivial too, because if 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.

We can make bigger changes too, because every stroke belong to a list and it is made by points, we can split this list according to criteria like angle or stroke size. Also, we can modify angles and position of every stroke, in a way to appear be two differents strokes.

If we want to make different sizes of brush we need define a new function w. It takes as parameter the stroke. This function mainly depends from stroke size, and it can be any, but we recommend following functions: $w(\phi) = \frac{\phi}{2} k_{max} \quad if \ \phi < \frac{1}{2}$ and $w(\phi) = (\frac{\phi}{2} + \phi^2) k_{max} \quad if \ \phi \geq \frac{1}{2}$, where ϕ is between

© The Eurographics Association 2003.

0 and 1, and k_{max} is maximal stroke width. This function defines how stroke begins and ends, and how is its width for every point in it.

For different kinds of paper absorption we can use other function t, whose values will be used as inputs to an alpha channel when we draw different strokes. In the same way, t can be take any value, we recommend $t(\phi) = \frac{acos(\phi)}{\pi} k_{max}$, where $k_{max} \in [0,1]$.

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

We don't evaluate shortest strokes, only them which are longer than a threshold value given by l. We recommend for ink color 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 from evaluate t in point i.

But, even after correcting angles and distances it is difficult to a human being draw something like this, because most of angles are straight angles or corners. To avoid that we define strokes from Bèzier curves, whose control points come determined by lists of angles and distances with an origin point (see Fig. 1).

Output is a polygon blend generated by lists of strokes (with changes in angles and distances) and Bèzier curves (which are applied with a width function defined by: $w(\phi) = \frac{acos(\phi)}{\pi} k$).

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

4. Results and Conclusions

Even thought ink human person pictures or, in general, life beings aren't usually used and almost all ink jobs are used in buildings, we have tried our algorithm in organic models (see Fig. 3), getting pretty good results.

Parameters σ, t_l and t_h affect basic sketch from the picture, a hight value of σ will smooth image in the beggining 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.

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).

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.*

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* $f_t(x) = x k_{max}$ *in the other.*

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

2. M. Salisbury, C. Anderson, D. Lischinsky, D.H. Salesin, "Scale-dependent reproduction of pen-and-ink illustrations", *Proceedings of SIGGRAPH 96, 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

© The Eurographics Association 2003.

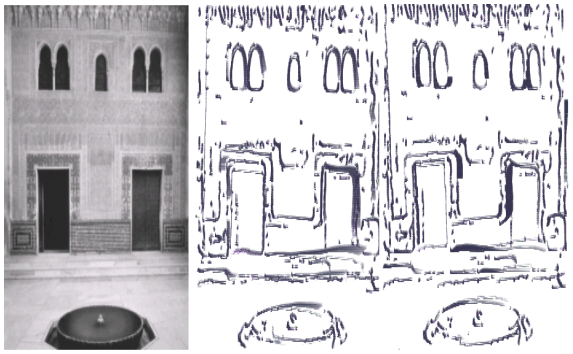

Figure 4: *We have changed parameters in Bèzier control points and some angles. Result is similar to change shadows in a picture. Parameters used in both images are:* $\sigma = 1$, $t_l = 0$, $t_h = 0.1$, $t_v = 4$, $l = 3$, $k_r = 0.8$

4. D. Martín, J.D. Fekete, J.C. Torres "Flattening 3D objects using silhouettes", *Eurographics'02, pp. 239-248, Saarbrucken, Germany, 2002*
5. D. Marr, "Vision - A Computational Investigation into the Human Represent" *1982*
6. J.A. Aznar, M. Moreno, "Simulación Computacional del Procesamiento Visual Biológico de Bajo Nivel" *Valencia, 2001*

Figure 5: *Differents kinds of parameters and functions in draw process, the first of strokes uses following values:* $\sigma = 1$, $t_l = 0$, $t_h = 0.07$, $t_v = 3$, $l = 4$, $k_r = 0.6$ *and as width functions:* $f_w(x) = \frac{x}{2} k_{max} \quad if \quad x < \frac{1}{2}$ *, and* $f_w(x) = (\frac{x}{2} + x^2) k_{max} \quad if \quad x \geq \frac{1}{2}$*, and for transparency functions* $f_t(x) = \frac{acos(x)}{\pi k_{max}}$ *in both kinds of stroke. Second one use following values:* $\sigma = 1$, $t_l = 0$, $t_h = 0.1$, $t_v = 3$, $l = 4$, $k_r = 0.8$ *and as width functions* $f_w(x) = \frac{acos(x)}{\pi k_{max}}$*, and for transparency functions* $f_t(x) = \frac{acos(x)}{\pi k_{max}}$ *in a kind of stroke and* $f_t(x) = x k_{max}$ *in the other. Results are similar.*

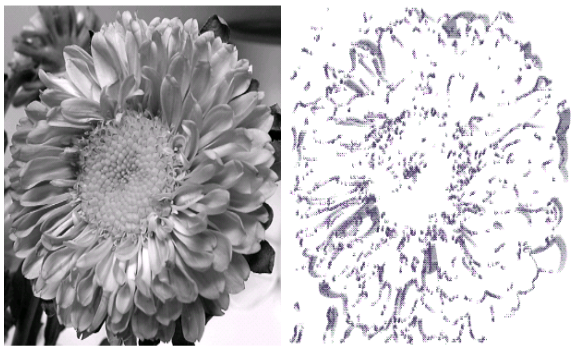

Figure 6: *For elements with too much details it is useful use* t_h *between 0.2 and 0.4, for this photograph we have used:* $\sigma = 1$, $t_l = 0$, $t_h = 0.2$, $t_v = 4$, $l = 3$, $k_r = 0.2$

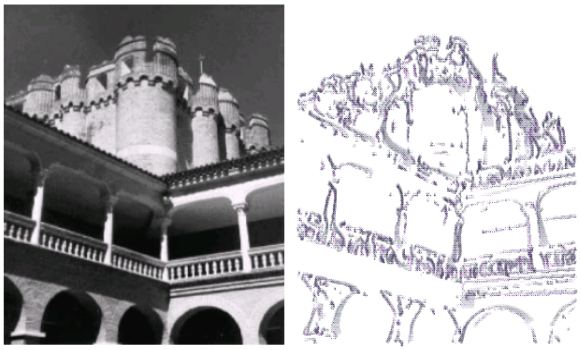

Figure 7: *Photograph with too much noise are smoothed because parameter* σ*, but details aren't supressed (thanks to parameter* t_h*), parameter used are:* $\sigma = 1$, $t_l = 0$, $t_h = 0.1$, $t_v = 4$, $l = 3$, $k_r = 0.2$ *width and transparency functions are same as before.*

© The Eurographics Association 2003.