A NEW APPROACH TO THINNING BASED ON TIME-REVERSED HEAT CONDUCTION MODEL*

Xinhua Ji, Jufu Feng

School of Electronics Engineering and Computer Science, Peking University
Beijing 100871, P.R.China
Email: jixinhua@pku.org.cn, fjf@cis.pku.edu.cn

ABSTRACT

In this article a novel thinning algorithm based on time-reversed heat conduction model is proposed. The image is viewed as a thermal conductor and thinning task is then considered as an inverse process of heat conduction. Given an image, the direction map of heat conduction is first computed, and then time-reversed heat conduction is simulated. The result turns out to be a thinned pattern. The algorithm can be applied to gray-scale or binary images.

1. INTRODUCTION

Thinning algorithms play an important role in image processing tasks as it simplifies object representation, feature extraction and pattern analysis. The main goal is to find skeletons of object, while preserving topological and geometrical properties.

Most of the research has been done on binary images. A binary image consists of objects (pixels valued 1) and background (pixels valued 0). Thinning algorithms on binary images iteratively remove pixels from the object edge, until one-pixel skeleton is obtained [1][2].

Thinning on gray-scale image is sometimes required. Some algorithms define connectivity for gray-scale image, and then remove edge pixels under the constraint of preserving connectivity [3][4].

The algorithm to be proposed, however, treats gray-scale and binary images indiscriminatingly. For gray-scale image, the gray-level is mapped into real values between [0,1]. Pixels valued 0 are assumed to be background, while non-zero pixels form objects to be thinned. Binary images can be considered the same way, only that pixels take value of either 0 or 1. This way we do not make a difference between gray-scale image and binary image.

* The work is supported by the National Natural Science Foundation of China (60175004).

The article is organized as following: Section 2 introduces heat conduction model and the idea to make a time-reversed simulation. Section 3 discusses the direction map of heat conduction. Section 4 proposes the whole thinning algorithm. Section 5 shows experimental results. The last section is conclusion.

2. TIME-REVERSED HEAT CONDUCTION

2.1. Heat conduction model

The basic idea here is to view the image as a 2D thermal conductor that consists of pixels, where pixel intensity represents the temperature. Pixels with higher value have higher temperature, while the background pixels have the lowest temperature. The function of intensity is now a function of temperature, noted by $T(x, y)$. Considering the model to be discussed, it is necessary to add a time variable, so we have $T(x, y, t)$.

In a thermal conductor, heat will flow from warmer area to cooler area as time passes. According to Fourier's Law of heat conduction, the heat flux Φ at a point, which is the flow of heat per unit area and per unit time, is proportional to the gradient of temperature at that point,

$$\Phi = -k\nabla T(x, y, t) \qquad (1)$$

k is called thermal conductivity. The minus in (1) suggests that heat flows in the opposite direction of temperature gradient.

The change of temperature over time at each point can be described using "heat conduction equation":

$$\frac{\partial T(x, y, t)}{\partial t} = \kappa \nabla^2 T(x, y, t) \qquad (2)$$

where κ is thermal diffusivity, which is related to k.

Given an initial temperature $T(x, y, 0) = T^0(x, y)$, the solution to (2) is:

t = 0 t = 0.5 t = 1.5

Figure 1. Heat conduction of a one-pixel width line

$$T(x,y,t) = \frac{1}{4\pi\kappa t}\iint T^0(x,y) \cdot \exp\left[-\frac{(x-\xi)^2+(y-\eta)^2}{4\kappa t}\right] d\xi d\eta \qquad (3)$$
$$= T^0(x,y) * \left[\frac{1}{4\pi\kappa t}\exp\left(-\frac{x^2+y^2}{4\kappa t}\right)\right]$$

where * is convolution .

It is seen from (3) that, heat conduction process has a property of low-pass filtering. The temperature at each point tends to be homogeneous. Figure 1 shows the result of heat conduction on an image. The one-pixel width line becomes thick when $t = 0.5$, and even thicker when $t = 1.5$ (κ=1 in both cases)

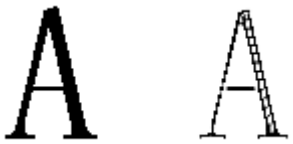
2.2. Time reversed heat conduction

It is intuitive from figure 1 that heat conduction eventually makes a line thicker over time. Thus a question is naturally arisen: can we obtain a thinner line from a thicker one if we "reverse the time"?

If time is reversed heat will flow from cooler area to warmer area, i.e. along the direction of the temperature gradient. We call it "Time-Reversed Heat Conduction", or TRHC for short. TRHC on an image can be simulated in the following process:

1. Calculating gradient map
   Calculate gradient vector of the image at each pixel, and its magnitude and direction.
2. Time-Reversed Heat conduction
   2.1 For each pixel, a small amount of temperature is subtracted. The amount, according to Fourier's Law, is proportional to the magnitude of gradient vector at the pixel.
   2.2 The subtracted temperature of this pixel is then added to one of its adjacent pixels, to which the gradient vector of this pixel points
3. Repeat 1-2 for a certain times.

Simulating the TRHC process on various images show interesting results. Sometimes, the thinned pattern is found.



(i) original image (ii) after TRHC

Figure 2. TRHC applied to letter 'A'

Yet sometimes the process fails its thinning purpose, it detects edges instead, as shown in Figure 2. The left stroke of letter A is thinned, while the right stroke splits into two, as if edges were detected.

Considering a thick line in a binary image, as figure 3(i) shows, the gradient vectors vanish except in edge areas. In the first iteration , those edge-adjacent pixels will receive the heat transferred from the edge pixels, and cause a raise in temperature. These pixels will have higher temperature than any other pixels in the following iterations, and will draw more and more heat from both exterior and interior pixels. This results in the edge-detection behavior of the process.

From above we see that the described process is sensitive to initial state. Though in each iteration the change to the image is small, the gradient map resulted may be very different, especially in orientation. As gradient map is calculated again and again, a small disturbance to the original image (e.g. noise) might make a big difference to the final output.

To counter the problem, Heat Flow Direction Map is introduced. It is computed only once, and acts as gradient at all iterations. It is to be explained in the next section.

3. HEAT FLOW DIRECTION MAP

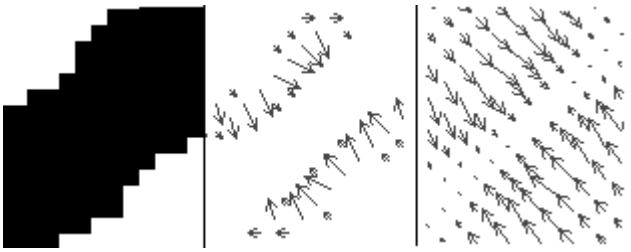
3.1. The motivation

For the purpose of thinning, we hope that heat flows from the edge of the object, to the center area eventually. If we can approximate the direction of heat flow at every pixel, regardless of time, it then could be used as gradient map in the TRHC process, in every iteration. We call it Heat Flow Direction Map, or HFDM.

In figure 3(ii), the gradient map provides heat-flow direction only at edge areas; away from these areas, it vanishes quickly. The HFDM is expected to (a) have the property that the gradient map has in edge areas, (b) but to change slowly in the image.

Assume the gradient map of original image is

$$(T_x, T_y) = \left(\frac{\partial T^0(x,y)}{\partial x}, \frac{\partial T^0(x,y)}{\partial y}\right) \qquad (4)$$



(i) original image (ii) gradient map (iii) HFDM

Figure 3 Heat-Flow Direction Map

We are seeking $[U(x,y),V(x,y)]$ which minimize

$$E = \iint (U - T_x)^2 + (V - T_y)^2 + \mu\left(U_x^{\,2} + U_y^{\,2} + V_x^{\,2} + V_y^{\,2}\right)dxdy \quad (5)$$

the term $(U - T_x)^2 + (V - T_y)^2$ in (5) expresses the need for (a), while $\left(U_x^{\,2} + U_y^{\,2} + V_x^{\,2} + V_y^{\,2}\right)$ speaks for (b), μ is a parameter to balance between them.

The idea comes from Xu [5], in [5] a Gradient Vector Flow is computed as an external force for snake deformation. The GVF is based on similar criteria to (a) and (b) , and the solution is calculated using numerical methods. The form adopted here as (5), however, is simpler than that of GVF, and an intuitive solution is found.

3.2. Finding HFDM

Using calculus of variations [6], Solution to (5) satisfies the following equations:

$$\begin{cases} \mu\nabla^2 U - U + T_x = 0 \\ \mu\nabla^2 V - V + T_y = 0 \end{cases} \quad (6)$$

Applying 2D Fourier transform[7] to (6) results in

$$\begin{cases} \widetilde{U} = \dfrac{1}{1 + 4\mu\pi^2(u^2 + v^2)}\widetilde{T}_x \\ \widetilde{V} = \dfrac{1}{1 + 4\mu\pi^2(u^2 + v^2)}\widetilde{T}_y \end{cases} \quad (7)$$

where $\widetilde{U},\widetilde{V},\widetilde{T}_x,\widetilde{T}_y$ is 2D Fourier transform of U,V,T_x,T_y , respectively.

According to (7), (U,V) can be obtained using a low-pass filter on the gradient map through frequency domain, the filter has the form of

$$H(u,v) = \frac{1}{1 + 4\mu\pi^2(u^2 + v^2)} \quad (8)$$

Thus the HFDM can be calculate using the following algorithm:

1. Calculate the gradient of original image, T_x, T_y
2. Perform 2D Fourier transform on T_x, T_y to get $\widetilde{T}_x, \widetilde{T}_y$
3. Apply the filter $H(u,v)$ to $\widetilde{T}_x, \widetilde{T}_y$,
4. Perform reverse Fourier transform to get U,V

Figure 3(iii) shows a HFDM resulted using the above algorithm. It is easily seen that, the direction map has the property that we want: It has vectors pointing from edge area to center area not only in edge area. This HFDM can be used as gradient map in the TRHC thinning algorithm, as to be discussed in the next section.

4. TRHC THINNING ALGORITHM

It is now ready to improve the TRHC process introduced in section 2. Instead of calculating gradient at each iteration, HFDM is now used in all iterations. This yields the TRHC thinning algorithm

4.1. Algorithm outline

1. Preparation. Let $i = 0$, let $T^0(x,y)$ be the original image. The image may be preprocessed using any noise-reduction techniques.
2. Calculating HFDM
   (a) Use the algorithm described in 3.2 to calculate HFDM of T^0, noted by $[U(x,y),V(x,y)]$
   (b) Calculate magnitude and direction of HFDM for each pixel

   $$M(x,y) = \sqrt{U(x,y)^2 + V(x,y)^2}$$

   $$\theta(x,y) = \arctan\frac{V(x,y)}{U(x,y)} + n(x,y)\pi$$

   where $n(x,y) = 0,\ 1,\ 1,\ 2$, if the vector $[U(x,y),V(x,y)]$ residents in 1st, 2nd, 3rd, 4th quadrant, respectively. This makes $\theta(x,y)$ to be a full-circle direction.
3. Time-Reversed Heat conduction.
   (a) For each pixel (x,y), a small amount of temperature is subtracted from $T^i(x,y)$. The amount is proportional to $M(x,y)$.

   $$T^{i+1}(x,y) = T^i(x,y) - \alpha M(x,y)$$

   α is a parameter that controls the overall speed of heat-flow
   (b) The subtracted temperature is then added to an adjacent pixel of (x,y) , say, (x',y')

$T^{i+1}(x', y') \leftarrow T^{i+1}(x', y') + \alpha M(x, y)$

(x', y') is one of the 8 neighbors of (x, y), which is in the direction of $\theta(x, y)$

4. Repeat 3 for N times. T^N is the output.

4.2. Discussion on the algorithm

In the first step of the algorithm, the noise-reduction preprocessing is not a necessity. However, it will help the algorithm to generate more smoothed skeleton. We use a 5 ×5 Gaussian mask $G(x, y) = \exp\left[-(x^2 + y^2)/\sigma^2\right]/2\pi\sigma^2$ with σ = 1.5 to do the job.

In the second step of the algorithm, by substituting HFDM for the gradient map in TRHC process, we've eliminated the edge-detection behavior, for the heat will always flow, according to HFDM, from the edge to the center area, regardless of time. Unlike the original process, the HFDM is calculated only once, and used for all subsequent iterations, thus the algorithm is no longer sensitive to initial data.

In 3(a) of the algorithm, a pixel's temperature is not allowed be negative. If a pixel's temperature reaches zero, it will not contribute any heat. Also in 3(b), any temperature higher than 1 is cut to 1.

There are 3 parameters in the algorithm. One is μ when calculating HFDM. We use μ = 2 to stress the need of slow-variation. The other is α in 3(a). However this is not an important parameter as it merely affects the overall speed of heat conduction, we use α = 0.2 in our experiments. The last one is the iteration time N. The larger N, the thicker lines that the algorithm can find a skeleton for. For the image with very thick line, the step 1-4 may be required to iterate multiple times.

5. EXPERIMENTS

Experiments on both artificial and realistic images have been done. These images include alphabet letters, Chinese characters, fingerprints, etc. Figure 4 and 5 are two examples. These experiments have shown that, though we have not considered much about it, the algorithm preserves connectivity well. However, there is no guarantee to get a one-pixel width line. This is caused by the nature of discrete computing, the HFDM may have two oppositely aligned vectors with same strength at two adjacent pixels.

6. CONCLUSION

We have introduced a new thinning algorithm based on time-reversed heat conduction. A heat-flow direction map

Figure 4. Example on Chinese character

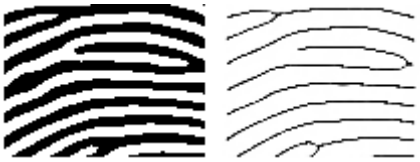


Figure 5. Example on fingerprint images

is calculated first, and heat flow is simulated using the direction map.

The algorithm proposed can be easily extended to higher dimensional space. The corresponding HFDM, however, need to be computed numerically.

Further research should lead to obtaining one-pixel width skeleton while preserving topological properties. Analysis on auto-selecting iteration times is also needed.

7. REFERENCES

[1] Louisa Lam, Seong-Whan Lee, "Thinning methodologies – a comprehensive survey", *IEEE Trans. PAMI*-14(9) pp. 869-885, 1992

[2] Ben-Kwei Jang, and Roland T. Chin, "Analysis of thinning algorithms using mathematical morphology", *IEEE Trans. PAMI*-12(6), pp.541-551, 1990

[3] Keiichi Abe, Fuyuki Mizutani, and Caihua Wang, "Thinning of gray-scale images with combined sequential and parallel conditions for pixel removal", *IEEE Trans. SMC*-24(2) pp. 294-299, 1994

[4] C. R. Dyer and A. Rosenfeld, "Thinning algorithms for gray-scale pictures", *IEEE Trans,PAMI*-1(1), pp.80-90, 1979

[5] C. Xu and J. L. Prince, "Gradient vector flow: a new external force for snakes", *IEEE Proc. Conf. On Computer Vision and Pattern Recognition*, pp.66-71, 1997

[6] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. 1. New York: Interscience, 1953

[7] R. C. Gonzalez, R. E. Woods, *Digital Image Processing* , Addison-Wesley publishing company, 1987