Text String Extraction in Delaunay Triangulation

Huimin Grace Guo[1*], Yi Xiao[1*] and Hong Yan[1,2]

[1]School of Electrical & Information Engineering, University of Sydney, NSW 2006, Australia
[2]Department of Electronic Engineering, City University of Hong Kong, Kowlon, Hong Kong

Abstract This paper establishes a Delaunay triangulation model for detecting text strings in an image. In this method, text regions are represented based on Delaunay triangles that can be extracted easily. A searching mechanism based on the left and right arms of a discrete symmetric axis is developed to locate text in an image. The discrete symmetric axis consists of a group of off-curve-edge adjacent Delaunay triangles that are identifiable. Our experiment shows that the approach can detect multi-orientated texts and is efficient and effective.

Key words: text extraction, connected component, Delaunay triangulation, and discrete symmetric axis

1. Introduction

On-line collections of digital images are becoming increasingly common. Understanding text information in static images and video sequences plays an important role in Web searching, image and video indexing, database organization and automatic annotation, where important text, such as book titles, captions, labels and some key words needs to be located. The first step needed to carry out this task is to locate text strings from a raster image. The variations of text in terms of character font, size and style, orientation, alignment, and complex background make the problem of automatic text location very difficult. In addition to good accuracy of text string detection, high speed of computation is also important.

The rest of this paper is organized as follows: a brief review of previous work is described in section 2; the proposed method is described in section 3; experiment results analysis is given in section 4. Finally, concluding remarks are made in section 5.

2. Previous Work

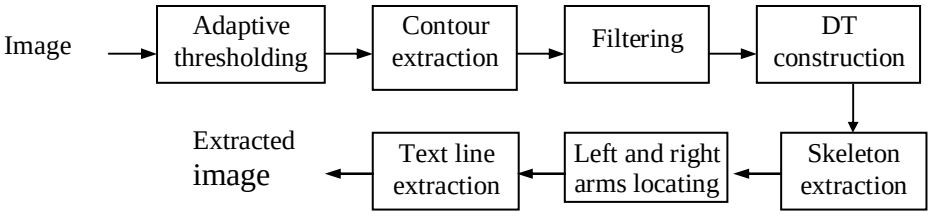
Texture analysis and connected component analysis are the two major methods proposed in the literature for text string location. The first method regards text regions as textured objects and uses well-known methods of texture analysis such as Gabor filtering and spatial variance [1] to automatically locate text lines. This usage of texture for text string location is sensitive to character font size and style. Further,

this method is generally time-consuming and cannot always precisely give text locations. The second method binarizes the images and extracts the connected components of characters and then groups them into character strings [2, 3]. A limitation of this method is that it works only on document images that do not contain many non-text regions. Fletcher and Kasturi [4] develop an algorithm for text string extraction from mixed text and graphics images. It generates connected components and groups the components into logical character strings using the Hough transform. The method is relatively independent of changes in text font style and size, and of string orientation, but cannot be applied in an image containing halftones. To handle various types of images, Jain and Yu [1] employ multi-valued image decomposition for a number of different domains, including classified advertisements, embedded text in synthetic web images, color images and video frames. The technique can be only applied to texts in either the horizontal or the vertical direction. Document images sometimes contain texts with different orientations. To solve this problem, Hase et al [5] recently divided a full color image into several representative binary images. A problem with this method is that pictures may be wrongly extracted as text strings.

This study develops a connected-component-based method that models the image containing texts using the Delaunay tessellation.

3. The Proposed Method

The processing procedure is as follows:



For the precise extraction of connected components, an edge sensing thresholding method is applied to grayscale images [10]. The contour of each connected component is traced by the 8-direction chain code method [11]. Component size and central points are calculated from the sequence of external contours. Big boxes and small boxes are filtered out for the purpose of retaining only the components of the size of character $\left(1 < w / w_m < 3, 1 < h / h_m < 3\right)$, while removing noise and larger non-text components.

Assume that in an image, inter-character space is less than line space, and that every text string consists of at least four characters. By representing the location of the connected components in an image with their component centroids, the image is presented as a set of points in the two-dimensional space. Then, the point pattern has the following distribution features:

a) The points of a text string are on a straight line;

b) For a text string, distance of a point to its adjacent points, is shorter than to other points in the point set.

Thus, we can perceive the point pattern of the image as random points and curves sampled at inter-character distance. To extract text strings, the task becomes reconstructing curves from the samples of points on it.

We propose a Delaunay triangulation method to cluster the points in a text line, which is one of the well-known proximity graphs. Impose Delaunay triangulation on the point set of the centroids, and then a Delauqany edge is the connection of a point to its packed neighbors. The points of a text line lie on the shortest edge of the corresponding Delaunay triangles. To sort the points into an order compatible with the natural trace of the curve (as perceived by a human), the discrete symmetric axis is found by ordering the specific Delaunay triangles. And thereby the left and right arm of the discrete symmetric axis can be extracted. Text lines are then detected from the arms by a collinear and a closeness test.

As the Delaunay triangulation computation does not depend on the image orientation, the method can extract text lines in any orientation.

3.1 The definitions and Lemma

Definition: An *on-curve-edge* is a Delaunay edge that is on the curve, and *an off-curve-edge* is a Delaunay edge that is not on the curve.

Definition: A Delaunay triangle is a *Side triangle* if one of its edges is an on-curve-edge.

Definition: The center of a circumscribed circle of a Delaunay triangle is the *Delaunay center*.

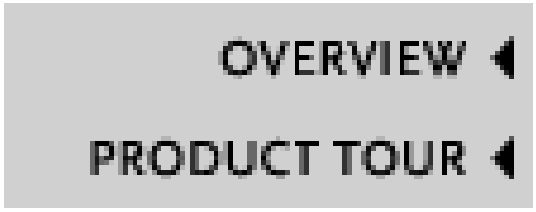
Lemma. If any point from a curve Γ has its adjacent point on Γ closer than to other points off Γ, then a pair of points adjacent along Γ lies on the shortest edge (SS) of a Side triangle.

Proof: (1) Any pair of points adjacent along Γ forms a Delaunay edge.
According to a property of Voronoi diagram/Delaunay triangulation, any point and its nearest neighbor point forms a Delaunay edge [8]. As any point on Γ has its adjacent point on Γ as its nearest neighbor, the pair forms a Delaunay edge.

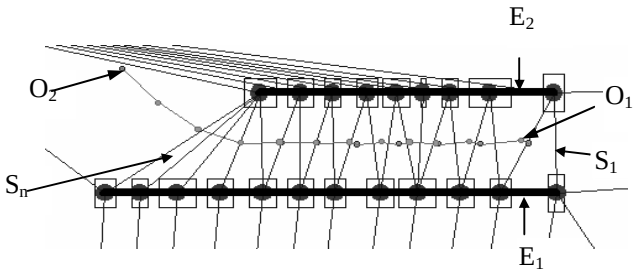
(2) A Delaunay edge formed by a pair of points adjacent along Γ is also the shortest edge in a Side triangle.
If this Delaunay edge is not the shortest edge in a Side triangle, then we have at least one of the points in the pair that is closer to the vertices off the curve than to its adjacent point on Γ. This is contradictory to the sampled curve assumption. So any pair of points adjacent along Γ forms a shortest edge of a Side triangle. □

3.2 Extraction of the left and right arms in a discrete symmetric axis

Applying the above lemma, when constructing the Delaunay triangulation on the centroids of the connected components, the points on a text line will lie on the *SSs* of the Side triangles



(a). An image containing text



(b). Text in the Delaunay triangulation.
The branch associated to the text is from O_1 to O_2.

Figure 1. Delaunay triangulation of an image containing text.

As shown in Figure 1 an example of the Delaunay triangulation of an image containing text. The points in the text lie on the *SSs* of the triangles. They form left and right arms of the discrete symmetric axes. For the branch (O_1O_2) associated to the text , O_1 and O_2 are its end points. Denoting $S_i(E_i, left_i)$ as a Side triangle having an on-curve-edge ' E_i ' and a Booler value ' $left_i$ '. If E_i is on the same side of the curve as E_1, we say it is on the *left arm*, called *left edge*. Then its $left_i$ = true. Otherwise, E_i is on the *right arm*, called *right edge*. Then its $left_i$ = false. The branch O_1O_2 consists of a set of Delauny centers of side triangles in $\{S_i\}$. $i = 1,2,\mathrm{K}\ n.$ Both S_1 and S_n have off-curve-edges adjacent to a Delaunay triangle's *SS* .

Trace sets of Side triangles $\{S_i\}$ and its left and right arm with the input: a set of Delaunay triangles $\{T_k\}\ k = 1,2,\mathrm{K}\ m.$, and the output: m_1 sets of Side triangles $\{S_i^{\ j}\}\ j = 1,2,\mathrm{K}\ m_1, i = 1,2,\mathrm{K}\ m_{2,} m_1 < m, m_2 < m.$

3.3 Collinear and closeness text line searching

To separate text lines from the other extractions, a collinear and closeness test is conducted. We propose an algorithm to find collinear text lines by checking the angle difference of any two adjacent left (right) edges. The points of a text line may not be strictly in a straight line if they are descending or ascending. When denoting the smallest positive angular difference between two adjacent left (right) edges e_{i-1} and e_i as θ_i, $0 \le \theta_i \le 90^0$., θ_i must be less than a threshold θ_{th}. θ_{th} represents the descent/ascent tolerance. The choice of θ_{th} depends on the character size and the text line space.

Points in a text line are not only aligned in a line but also close to each other. Closeness is also a feature to test. The closeness test is conducted by checking L_i, the length of E_i. When $L_i > L_{th}$, we say e_i is not in the same text line as e_{i-1}. For a character of size $S(w_m \times h_m), L_{th}$ is chosen as $2S \sim 3S$.

If $\theta_i < \theta_{th}$ and $L_i < L_{th}$ we say e_i is in the same text line as e_{i-1}.

Detect the text line with the input: a set of edges $\{e_i\} i = 1,2,\mathrm{K}\ n_0$, which forms a left(right) arm; and the output: n_1 sets of edges $\{e_k^j\} j = 1,2,\mathrm{K}\ n_1, k = 1,2,\mathrm{K}\ n_2, n_1 < n_0, n_2 < n_0$.

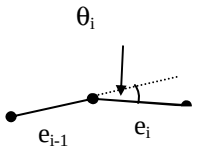


Figure 2. Adjacent left (right) edge

4. Experiment results analysis

To evaluate our method, experiments are conducted on a variety of images including web pages, book coverings, and videos. They consist of text strings, pictures and halftone images. The size, font and orientation of characters may vary. The experiments were computed on a 700MHz Pentium-III computer. The Programs were implemented in C++.

The Delaunay triangulation is generated with the conventional incremental method. As the triangulation is generated on only the points of the components of the size of a character, the processing time is considerable. The information from each triangle, including: vertices, longest side, shortest side and adjacent triangle, is stored

for further processing. For text line extraction, $\theta_{th} = 25^0$ gives satisfactory results. L_{th} is font associated. We set $L_{th} = 2w_m$, for evaluating the effect of character touching or spaces between words.

The experiments results are generated using SSs, the shortest sides of Side triangles. The correct extractions are from the points in text lines that are arranged to meet the assumption. The reasons for failure are: characters are spaced sparsely, characters are touching and some random points have the same size as the font's and are distributed on a line.

The extractions are filtering dependent. With different values for w_m and h_m, we obtain different results. Figure 3 shows successful text location using the proposed method. In Figure 3 (b), The text line 'The University of Sydney' is the result for $h_m = 15$ pixel, $w_m = 10$ pixel and the other text lines are the result for $h_m = 3$ pixel, $w_m = 3$ pixel. Figure 4 is the result for an image containing a skewed text. The approach isn't affected by the orientation. Figure 5 shows the extracted result from a web page. It contains two text strings with different font orientations and size. The results were obtained in two steps. When setting $h_m = 30$ pixel, $w_m = 20$ pixel, the titled strings were located. The horizontal string was extracted when setting $h_m = 6$ pixel, $w_m = 4$ pixel. A few points in the 'flower' region wrongly extracted as text. The reason is that their connected components have the same size as the fonts and are located as text. Figure 6 shows the extracted result from a video sequence. In Figure 6(a), Character "A" and "D" are missed. The reason is that their inner-word distance is larger than the line space. Consequently, its point pattern does not meet the assumption for using this method. In Figure 7, the small text line 'The' is missed. It is regarded as random points in non-text regions because it has less than 4 characters. The text 'ces' is also missed because the characters touch each other causing the distance between the centroids of the characters to be larger than the distance from the centroids to other points. The other untouched text strings are extracted correctly. Figure 8 shows the extracted result from a book cover. The text strings are extracted correctly.
Resolution significantly affects the location accuracy. In low-resolution images, characters may not be separated by thresholding techniques.

Table 1. Correct location rate

Image type	Web page	video	Book cover
Correct location rate R	92%	80%	98%

Table 1 shows the tested correct location rate of different resolution images. Correct location rate R is defined as

$$R = \frac{\text{Number of characters located}}{\text{Total number of characters}}.$$

Video images have the lowest resolution, therefore causing touching characters that cannot be extracted by this approach. In web page images, some random points in a picture or halftone areas may be similar to text lines therefore being wrongly located as text strings. Thus, 'book cover' has the highest correct location rate.

Thresholding computation is the most time-consuming part of the whole processing time. Thresholding computation time depends on the image size.

Table 2 Processing time (r =10 for Bernsen's search window)

Image	**Image size (pixel)**	**Number of connected compnent**	**Font size**		**Computation time(s)**	
			W_m ***(pixel)***	h_m ***(pixel)***	**Thresholding**	**Others**
Sample1	145×242	98	5~10	7~14	0.54	0.04
illu2	394×251	53	11~22	16~32	1.62	0.09
illu4	605×403	318	8~16	10~20	4.06	0.37

Table 2 shows the processing time with different image sizes. Apart from thresholding, the approach has a very short processing time. An improvement in thresholding computation time is needed.

5. Conclusion

In a text string, its centroids of the connected components are distributed not only in a straight line but also close to each other in an innerword's distance. In Delaunay triangulation, the points in a text lie on the SSs of the Side triangles. A Side triangle is a Delaunay triangle that has one edge on the curve. A Delaunay center of a Side triangle can approximate the skeleton point associated with the text line. By joining all the skeleton points between two off-curve-edge adjacent Side triangles, the discrete symmetric axis associated with the text line is obtained.

Discrete symmetric axes can be easily obtained by searching adjacent Delaunay triangles, and furthermore locate the left and right arms on which the text lines lie. As the computation of Delaunay triangulation is independent of orientation, the method can detect multi-orientated texts. Experiments on a variety of images show that the proposed algorithm is also efficient and accurate.

Acknowledgement: This work is partially supported by an Australian Postgraduate Award (Industry) and a grant from the Hong Kong Research Grant Council (CityU 1088/00E).

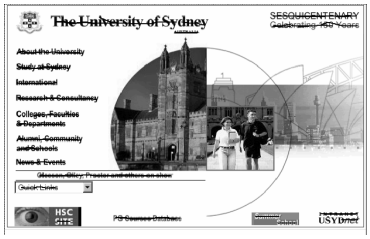
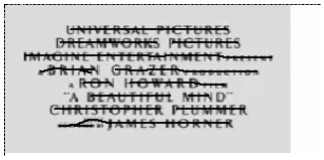


Figure 3. Successful text location. Two different sized fonts, one is h_m=15 pixel, w_m=10pixel; the other is h_m=3 pixel, wm=3 pixel.

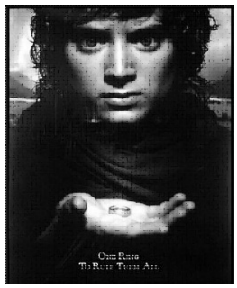
Figure 4. The result for an image containing skewed text.

Figure 5. The extracted result for an image from a web-page in different font and different orientation.



(a)

(b)



(c)

Figure 6. The extracted result from video sequences. In (a) character 'A' and 'D' are missed.

Figure 7. The extracted result from a web page. The small text line 'The' is missed. The touching characters 'ces' are also missed.

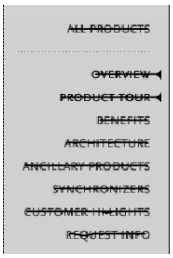


Figure 8. The extracted result from a book cover.

References

1. A. Jain, B. Yu, "Automatic text location in images and video frames", *Pattern Recognition* Vol. 31(12) pp. 2055-2076, Dec. 1998
2. L. O'Gorman, "The document spectrum for page layout analysis", *IEEE Trans. Pattern Anal Machine Intell.* Vol. 15, No. 11 pp. 1162-1172, 1993
3. Y. Xiao and H. Yan, "Text region extraction in a document image based on the Delaunay Tessellation", *Pattern Recognition*, 36(2003) pp. 799-809
4. L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images", *IEEE Tran. On Pattern analysis and machine intelligence*, Vol 10, No. 6, pp. 910-918, Nov. 1988
5. H. Hase, T. Shinokawa, M. Yoneda and Ching Y. Suen, "Character string extraction from color documents", *Pattern Recognition* Vol. 34 (7) pp. 1349-1365, 2001
6. L. H. D Figueiredo and J. D. M. Gomes, "Computational morphology of curves", *Visual Computer*, 1995
7. J. W. Brandt and V. R. Algazi, "Continuous skeleton computation by Voronoi Diagram", *CVGIP: Image Understanding*, Vol 55, No. 3, pp. 329-338, May, 1992
8. A. Okabe, B. Boots and K. Sugihara, "Spatial Tessellations-Concepts and Applications of Voronoi Diagrams", 1992
9. Yi Xiao and Hong Yan, "Symmetry based corner finding", *IEEE Tran. On Pattern analysis and machine intelligence*, to be published
10. J. Bernsen, "Dynamic thresholding of gray-level images", in *Proc. 8th ICR*, Paris, 1986, 1251-1255
11. J. Jimenez and J. L. Navalon Some, "Experiments in Image Vectorization". *IBM J. Res. Develop*. Vol. 26(6), pp. 724-734, 1982.