Computers in Biology and Medicine 63 (2015) 277–286

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

Finding splitting lines for touching cell nuclei with a shortest path algorithm

Xiangzhi Bai [a,*], Peng Wang [a], Changming Sun [b], Yu Zhang [a], Fugen Zhou [a], Cai Meng [a]

[a] Image Processing Center, Beijing University of Aeronautics and Astronautics, Beijing 100191, China
[b] CSIRO Computational Informatics, Locked Bag 17, North Ryde, NSW 1670, Australia

ARTICLE INFO

Article history:
Received 15 April 2014
Accepted 11 October 2014

Keywords:
Splitting line
Touching cell nuclei
Shortest path
Segmentation

ABSTRACT

A shortest path-based algorithm is proposed in this paper to find splitting lines for touching cell nuclei. First, an initial splitting line is obtained through the distance transform of a marker image and the watershed algorithm. The initial splitting line is then separated into different line segments as necessary, and the endpoint positions of these line segments are adjusted to the concave points on the contour. Finally, a shortest path algorithm is used to find the accurate splitting line between the starting-point and the end-point, and the final split can be achieved by the contour of the touching cell nuclei and the splitting lines. Comparisons of experimental results show that the proposed algorithm is effective for segmentation of different types of touching cell nuclei.

1. Introduction

Splitting of touching cell nuclei is a crucial technique in image analysis [1–9]. Many algorithms have therefore been proposed for this purpose [1–9]. Some of the basic tools used by these algorithms are ellipse fitting [4,5], shape information of the contour [2,6,7], watershed [1,8–10], and so on. The splitting line is important in this context. A good splitting line should be located at the edge inside the touching area. The splitting line shows the possible complete contour of a single nucleus. This is very important for biological applications such as cell analysis and cell measurement. Ellipse fitting-based algorithms use ellipses to represent each separated nucleus of the touching cell nuclei. However, the ellipses cannot be considered as the actual boundary of each nucleus. Shape information-based algorithms usually give straight lines as the splitting lines, which are not accurate. Watershed-based algorithms can give the contour of each nucleus, but the contour is usually affected by the positions of markers [9]. Therefore, different markers of the same region may generate different contours [9], and the position of the splitting line will change if the markers are different. These algorithms cannot always give the same splitting line for different sets of markers.

To find good splitting lines, a shortest path-based algorithm is proposed in this paper. First of all, we apply an ellipse fitting-based algorithm [11] to obtain the ellipse centers as the markers. Then, the distance transform of the marker image is used to find the initial splitting lines. These lines can be separated into several line segments, which are used for splitting the multiple nuclei. Furthermore, the starting-point and the endpoint of each line segment are obtained and adjusted. Finally, with each pair of endpoints we apply the shortest path algorithm in a local area to obtain the final splitting lines. Therefore, the final segmentation result can be obtained by combining the contour of the nuclei and the splitting lines. Because of the endpoint searching strategy and the shortest path algorithm, the splitting line runs along the edge inside the touching cell nuclei and it is easy to obtain the complete contour of each nucleus. To evaluate the performance of the proposed algorithm, other algorithms are used for comparison using quantitative and qualitative methods. Experimental results show that our algorithm is effective.

A preliminary version of this paper was presented in [12]. Compared to [12], the extension mainly includes (1) proposing a novel concave point-finding method, (2) adjusting the initial splitting line by rules, (3) adding quantitative evaluation of the experimental results.

2. Algorithm

The procedure employed by our algorithm is illustrated in Fig. 1. Details of the algorithm are given below.

2.1. Ellipse fitting-based marker finding

Each pair of touching cell nuclei can be roughly separated by any line between the two nucleus centers. We refer to the nucleus

* Corresponding author. Tel.: +86 1082338578.
E-mail address: jackybxz@buaa.edu.cn (X. Bai).

http://dx.doi.org/10.1016/j.compbiomed.2014.10.008
0010-4825/

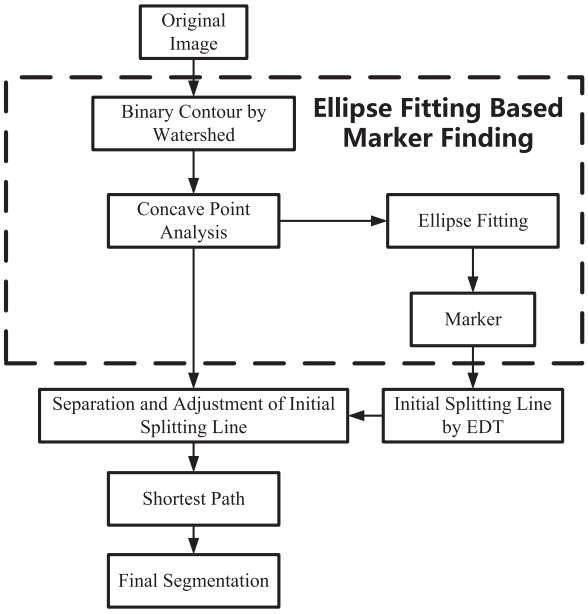

Fig. 1. Procedure of the proposed algorithm.

center as a marker in this paper. As the contour of the nucleus is similar to the ellipse, the ellipse center can be regarded as the marker of the nucleus. Therefore, we use the ellipse fitting-based algorithm [11] to extract the nucleus markers. The algorithm adopted from [11] contains four steps.

The closed contours of touching cell nuclei are extracted by a marker-based watershed algorithm, following the procedure shown in Fig. 2. We take the eroded result of Otsu binary image as the marker of the foreground, and take the distance transform of the foreground as the marker of the background. After the foreground and the background are both marked as the regional minimum, the contour with the largest gradient between the foreground and the background can be generated by watershed transform.

After that, each contour is segmented into several arcs by concave points, each fitting one ellipse. Finally, the fitted ellipses are combined on the basis of their shape information. Details can be found in [11].

We propose a simple and efficient method to find concave points, rather than the method presented in [11]. Firstly, polygon vertices from a contour are saved in a sequence by a polygon approximation algorithm [7,11]. Then, for each point like A with two neighboring points B and C (see Fig. 3), the cross product of two vectors AB and AC is calculated. If the cross product is negative, A is saved as a concave point, such as the example shown in Fig. 3(a). Otherwise, A is not a concave point (see Fig. 3(b)). With this method, it is not necessary to calculate the concavity of every point. The concave points along the contour can be found more efficiently. Finally, the concave angle of each concave point is compared with a pre-set threshold. If the concave angle is larger than the threshold, this concave point should be removed. In practice, if the concave angle of a point is greater than 150°, this point is usually on a smooth line. Therefore, we choose 150° as the threshold of the concave angle in this paper. The experimental results also show the effectiveness of this threshold.

Fig. 4 shows the procedure for generating markers. Fig. 4(a) shows the original touching cell nuclei for segmentation. Fig. 4(b) is the fitted ellipses for the nuclei and Fig. 4(c) is the extracted markers of the nuclei. With these markers, we can find the initial splitting lines by a proper method.

2.2. Initial splitting line obtained by Euclidean distance transform

As the markers of touching cell nuclei have been obtained, we only need to find a proper method to generate the splitting line. Distance transform is an effective method for separating points according to their distance distribution. However, the position of the splitting line varies with different markers. Therefore, the splitting lines obtained by distance transform are regarded as initial splitting lines only.

Different types of distance transforms have been proposed to improve the performance of segmentation algorithms [3,13]. In our algorithm, we use the Euclidean Distance Transform (EDT) [13] on the complementary marker image.

Fig. 5(b) shows the result of distance transform and Fig. 5(c) shows the initial splitting lines obtained by EDT. We can see that the touching cell nuclei can be separated into different regions with only a single nucleus inside. Obviously, the endpoints of the splitting line should be located at the concave points. Thus, the initial splitting line should be adjusted.

2.3. Separation and adjustment of initial splitting line

The initial splitting lines obtained by EDT separate the whole image into areas based on markers. As we do not take the contour information of cell nuclei into consideration, the initial splitting line is not accurate. Generally, a reasonable splitting line should start and end at the concave points on the contour. Therefore, we separate the initial splitting line into endpoints and adjust these endpoints to the nearest concave points.

If there are some line segments touching (see Fig. 6(a)), they can be separated by tracing the points on each of them (see Fig. 6(b)). Suppose there are N (in Fig. 6(a), $N=3$) line segments touching each other. S_i is the starting-point of line segment L_i ($1 \le i \le N$). The separation procedure is given below:

(1) Take any starting-point (S_i) of one line segment (L_i) as the initial tracing point pt_s: $pt_s = S_i$.
(2) Set the value of pt_s as 0 and check the eight-neighbor of pt_s.
(3) If there is one pixel pt_j with a value of 1, let $pt_s = pt_j$, go to (2).
Else if there is no pixel with a value of 1, set the endpoint of the line segment L_i: $E_i = pt_s$, set the value of pt_s as 0.
Else if there is more than one pixel with a value of 1, which means L_i is one of the touching line segments, set the endpoint of the line segment L_i: $E_i = pt_s$, set the value of pt_s as 0.

After all the line segments have been checked, each pair of touching cell nuclei is represented by a pair of starting-point (S_i) and endpoint (E_i). These make up the original coordinate array of starting-points and endpoints (an SE array).

Naturally, the concave points segment the closed contour of the nuclei into different arcs. In the SE array, each endpoint on the contour must belong to a definite arc ending at a pair of concave points. Therefore, the endpoint can be adjusted to the concave point with less distance within this arc. In this paper, we utilize the contour point index and Euclidean distance to adjust the coordinates of the endpoints on the contour. In the implementation of our algorithm, the points of the contour are saved in an anticlockwise array, which starts at the leftmost point. We define the index of each point in this contour array as contour point index (*CPI*). For any point A on the contour, $CPI(A)$ represents the index of the point A stored in the contour point array. Fig. 7 shows a sample adjustment of endpoints. If the endpoint is not on the contour like E_2, it remains the original coordinates. Otherwise, the endpoint on the contour like E_1 should be adjusted to the nearest concave point.

For any endpoint pair assumed as E_1 and E_2, E_1 is first adjusted by the following procedure. E_2 is adjusted in the same way.