Image Edge Enhancement and Segmentation via Randomized Shortest Paths

Ming Xu, Jun Wang, and Zeyun Yu*
Department of Computer Science and Electrical Engineering
University of Wisconsin at Milwaukee
Milwaukee, WI 53211, U.S.A.
* Corresponding author: yuz@uwm.edu

***Abstract*—This paper describes a new method for image edge enhancement and boundary segmentation. Like many interactive graph-based segmentation methods, users are asked to provide some foreground (or object) and background seeds. A set of randomly generated points representing the foreground are paired with another set of random points representing the background. The corresponding shortest paths of all such pairs are found and accumulated. These paths tend to go through the boundaries of the object of interest. Therefore, the accumulated paths can enhance the object edges, from which the final segmentation is obtained. Several experiments are provided to demonstrate the effectiveness of the proposed approach.**

***Keywords*-image segmentation; edge enhancement; shortest paths; Monte Carlo method.**

I. INTRODUCTION

Image segmentation, as a fundamental problem in image processing and computer vision, tries to separates an object of interest from surrounding background in an image and has received tremendous attention from researchers in the computer imaging and vision areas. Many other tasks, such as shape modeling and feature recognition, rely largely on correct segmentation. Traditionally image segmentation has been performed by hand, but manual segmentation is very tedious, inaccurate, and quite often subjective from person to person, even with the help of sophisticated graphical user interface [1,2]. On the other hand, automated segmentation is still considered to be one of the hardest problems in the field of image processing, although various techniques have been described for automatic or semi-automatic segmentation. Commonly used methods include segmentation based on edge detection [3], region growing and/or region merging [4], active curve/surface motion [5], watershed immersion method [6], and normalized graph cut [7] and eigenvector analysis [8].

Among the popular methods used in image segmentation are graph-based methods. In these methods, an image is treated as a graph, where each pixel is a node and an edge is created between two adjacent pixels with 4- or 8-connectivity. There are several core graph-based algorithms, on top of which many other approaches are built. The first method is based on graph cuts [9,10,11], where the goal is to partition the graph into several connected components with certain optimization criteria and each of them corresponds to one segmented feature (or background). This method works in a way similar to data classification – the intra-class difference is minimized while the inter-class difference is maximized. The second graph-based method is the random walker method [12]. This approach also treats the image as a weighted graph but assigns each pixel to a seed (belonging to an object or background) by considering the most likely random walker between the pixel and the seed found. The third widely used graph-based method for image segmentation is the shortest path approach [13]. This approach assigns a pixel to the object if the path from the pixel to the object seed is shorter than paths to the background seed. All the three approaches have been formulated into a common framework in [14,15].

In the present paper we propose a new approach for image edge enhancement and segmentation. Our method also treats an input image as a weighted graph and utilizes shortest paths between two pixels. However, our method differs from other approaches in that the source and sink of a path are randomly generated and likely (but not exactly) represent the object and background respectively. Each pixel is associated with an *edge confidence* (EC) value that is initialized as zero. Whenever a shortest path is found, all pixels on the path are added by one on their EC value. The final EC map of the whole image gives an enhanced edge map of the original image, and thus the segmentation can be performed by simply running the shortest path twice on the object boundaries. As the number of shortest paths executed affects the quality of the EC map (typically, the more the number of paths, the better the EC map), our method can be thought of as a Monte Carlo method.

The rest of the paper is organized as follows. In Section II, we give the detail of our algorithm in several steps with an example of cardiac cell image. Section III shows some edge enhancement and segmentation results, followed by our conclusion in Section IV.

II. METHOD

Our method is based on the graph constructed from the original image. As most of other graph-based methods, we treat each pixel as a node and the 4-connectivity between pixels as

The work described was supported in part by an NIH Award (Number R15HL103497) from the National Heart, Lung, and Blood Institute (NHLBI) and by the UWM Research Growth Initiative.

an edge in a graph. First, we define a gradient-dependent value on every pixel in the image is as follows:

$$G(p) = \exp(-0.1 * \|\nabla I\|), \quad (1)$$

where ∇I is the gradient of the input image *I*. Then the edge e_{ij} connecting two vertices p_i and p_j is assigned with the weight as:

$$e_{ij} = \frac{G(p_i) + G(p_j)}{2}. \quad (2)$$

With the edge weight defined above, any shortest path between two pixels in the image would try to go through the image edges as the gradient magnitudes are high and hence the edge weights are low.

Different from other graph-based methods, the idea of our method is to use a number of randomized shortest paths to enhance image edges. Every pixel p is associated with a so-called *edge confidence* (EC) value, which is initialized as zero for every pixel. Whenever a pixel p is visited by one shortest path, the corresponding EC value is accumulated by one. After all the shortest paths are found and applied to the visited pixels, the EC value of a pixel will give a clue of how strong this pixel lies on the image edges, hence producing enhanced edge map of the original image. The key in this process is to determine the end points (source and sink) of the shortest paths, as finding the shortest path between two given points is a standard procedure in graph theory. Below we shall give the details of our algorithm in several steps.

A. Initial Seed Selection by Users

In this step, users should manually pick some seeds representing foreground (object) and background (non-object) using different colors. Usually the number of the object seeds is less than that of the background seeds. Typically just a few (< 10) object seeds are necessary but they should be distributed evenly inside of the object of interest. The background seeds should be distributed around the object. Figure 1 illustrates an example of initial seed selection on an electron microscopic image of cardiac cells. As explained below, these seeds only have high probability to be chosen as the object and background. They may not be used as the sources or sinks of the shortest paths in the segmentation algorithm.

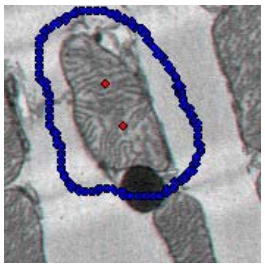


Figure 1. Initial seeds selection on an electron microscopic image of cardiac cells (image size: 177*180 pixels).

B. Generating Marching Distance Maps

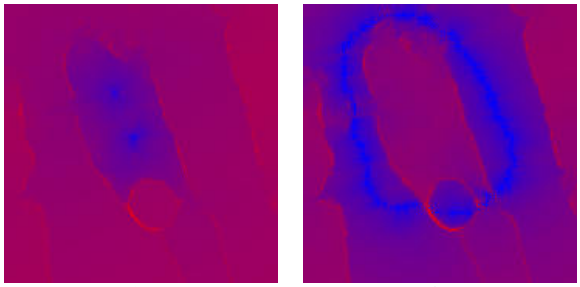
The user-selected seeds mentioned above roughly tell us where the foreground and background are in the image. The pairs formed between them, however, may not be statistically sufficient to enhance the edges for accurate segmentation. To add new pairs for the shortest paths, we must know where the end (source or sink) points are and whether they are in foreground or background. The classification of a position (i, j) is achieved by using the marching distance as defined below:

Definition: Given a 2D scalar image *I*, the *marching distance* between two points *A* and *B* in the function domain is defined by [16]:

$$MD_I(A,B) = \min\{\int_{A\to B} e^{\|\nabla I\|} ds\} \quad (3)$$

where $\int_{A\to B}$ is the integral along a path from *A* to *B*. The minimization is conducted over all the paths from *A* to *B*. Apparently the marching distance favors a path that goes through low-gradient (or non-edge) regions. In other words, if two points are in the same feature, the marching distance between them should be small.

Given a seed pixel *S* in an image, the marching distance from S to all other pixels is so-called marching distance map, which is similar to the shortest path distance from S to all other pixels in the graph representation of the image. The marching distance map can be efficiently computed by using the fast marching method [17].



(a) (b)

Figure 2. (a) Marching distance map from the user-picked object seeds. (b) Marching distance map from the user-picked background seeds. The seeds are shown in Figure 1. The blue color means smaller marching distances and by contrast the red color indicates larger marching distances.

Figure 2 shows the marching distance maps by treating the initial object seeds and the initial background seeds as the seeds for the fast marching method [17], where the object and background seeds are picked by a user as shown in Figure 1. The colors in both maps are defined as: blue color means small marching distance and red color means larger marching distance. In other words, the blue pixels have higher chance to be classified as objects while red pixels are more likely to be the background in the given image.

C. Random Seed Selection

The marching distance maps shown in Figure 2 are then utilized to randomly generate source and sink points that will be used to form the shortest paths. Basically the value of each pixel in the marching distance map tells us the probability of that pixel being the source or sink point. The lower the value is, the higher the probability would be. Below we briefly explain the steps of generating source points (representing the object).

1. Randomly choose a pixel from the image. We denote this random pixel as P. Let M (P) be the value of P in the marching distance map (see Fig. 2).
2. We then generate a random number V (P)$\in$ [0, T], where T is a pre-defined threshold. If M (P) $\leq$ V(P), then the pixel P is chosen as a source point (representing the object). Otherwise, ignore P.
3. Repeat (1-2) until sufficient number of source points are chosen.

A similar procedure is applied to generate a set of sink points (representing the background). Figure 3 shows an example of source (in red) and sink (in blue) points. Note that because of the random number generation, some source points are in the background and some sink points in the foreground (object) regions. These wrong classifications would affect the segmentation. But as we will see below, our final segmentation is a total contribution of all the shortest paths formed by the source and sink points. Just a small number of wrong classifications would not make much difference to the final results. To that sense, our method is a Monte Carlo based method.

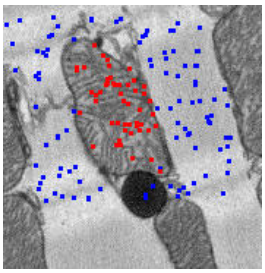


Figure 3. The generation of source (in red, representing object) and sink (in blue, representing background) points by using random number generation.

D. Image Edge Enhancement by Shortest Paths

The source and sink points generated above (see Figure 3 too) are paired and used to find the shortest paths by using the edge weight as defined in Equation (2). For example, if we have n source points and m sink points, then $n \times m$ pairs will be considered and hence $n \times m$ shortest paths will be detected in the graph. The algorithm we use to generate a shortest path is the Shortest Path Faster Algorithm (SPFA).

Figure 4 shows the shortest path between one of source (object) point and one of sink (background) points. We can see that this path favors passing through the boundary of the object because according to equation (2), the edge weights are small on the boundary of the object.

Figure 4. The shorest path between one of the source (object) points and one of the sink (background) points. Note the path favors going through the boundary of the object due to the small edge weights on the boundary.

Figure 5 shows m shortest paths between one of the source points and all the m sink points chosen by the random method described above. Although we see many branches on the paths, all paths do prefer to go through the boundary of the object.

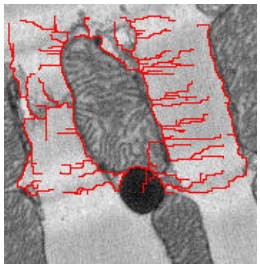


Figure 5. The shorest paths between one of the source points (representing the object) and all the sink points (representing the background).

We then create a new image EC(i, j), representing the edge confidence of the pixel (i, j), initialized as zero. For every one of the $n \times m$ shortest path, if a pixel (i, j) is on the path, we add one to its edge confidence value EC(i, j). After all $n \times m$ shortest paths are checked, we will have a gray-scale image EC(i, j), in which a large value indicates a high probability of the pixel being on the object boundary. Figure 6 shows such an example after the EC(i, j) image is rescaled to [0, 255]. From this figure we can see that the image edges are enhanced due to the preference of the shortest paths going through the object boundary. It is obvious now that our algorithms works like a Monte Carlo method: the result becomes more and more accurate as the number of samples (in our case , the shortest paths) increases. Also, the more uniform the random numbers are, the better the result would be.

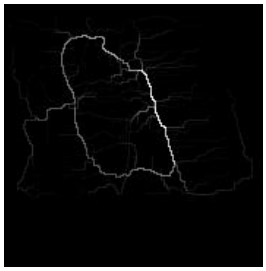


Figure 6. The edge confidence (EC) map formed by adding one to pixels where a shortest path goes through.

E. *Image Segmentation from the Enhanced Edges*

After the image edges are enhanced by using the randomized shortest paths, we detect the object boundary by running the shortest path one more time. First, we search the edge confidence (EC) image to find out the brightest pixel (denoted by A) and we assume that this point be on the boundary of the object. Then we search the neighborhood of this pixel and find out the brightest neighboring pixel (denoted by B). Then we form another graph by using the EC image. In this graph, the edge weight is the inverted gray-scale value in the EC image, such that the edges on the object boundary have smaller edge weights. In particular, an edge where one of the end points has zero EC value is assigned the weight infinite. Also, the edge between A and B mentioned above is also assigned the weight infinite. With these specifications, the shortest path between A and B in this new graph would result in the segmentation as shown in Figure 7 below.

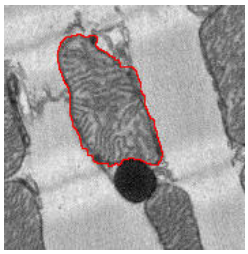


Figure 7. The final segmentation result by using the randomized shortest paths, an algorithm based on the Monte Carlo method.

III. RESULT

In this section we will test the proposed algorithm on several biomedical images to show the effectiveness of our method. Figure 8 shows a noisy image of cardiac cell with 264*235 pixels. In Figure 8 (a) we choose 3 initial object seeds (in red) and a number of initial background seeds (in blue). With these seeds, we generate the marching distance maps for both the object and the background, as shown respectively in Figure 8 (b) and (c). These maps are then used to randomly generate the source (in red) and sink (in blue) points, representing the object and background respectively, as seen in Figure 8 (d). In this example, there are a total of 50 source and 100 sink points. These points are paired and the corresponding 5,000 shortest paths are found in about 1.6 seconds. Figure 8(e) and (f) show the rescaled edge confidence image and the final segmentation result of our method.

Figure 9 shows another example of noisy cardiac cell image with 125*145 pixels. Figure 9(a) shows the initial seeds (one object seed and a number of background seeds) picked by the user. Figure 9(b) and (c) show the marching distance maps by treating the user-picked seeds as the seeds for the fast marching method. Figure 9(d) shows the randomly generated source (red) and sink (blue) points by using the marching distance maps as the probability. The computed edge confidence map that combines all the shortest paths is shown in Figure 9(e). Figure 9(f) shows that the segmentation result. In this example, we generates 50 source points and 100 sink points, and it takes about 0.64 second to find all the 5,000 shortest paths.

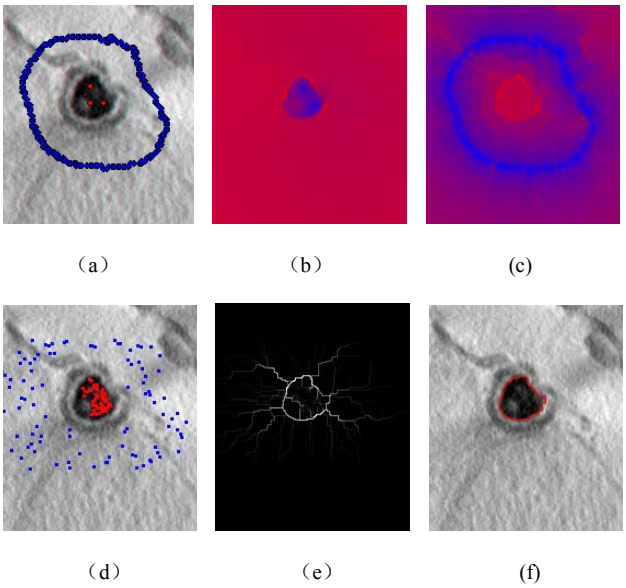


Figure 8. Illustration of our algorithm on a cardiac cell with 264*235 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The computed edge confidence image by adding all shortest paths together. (f) The final segmentation result.

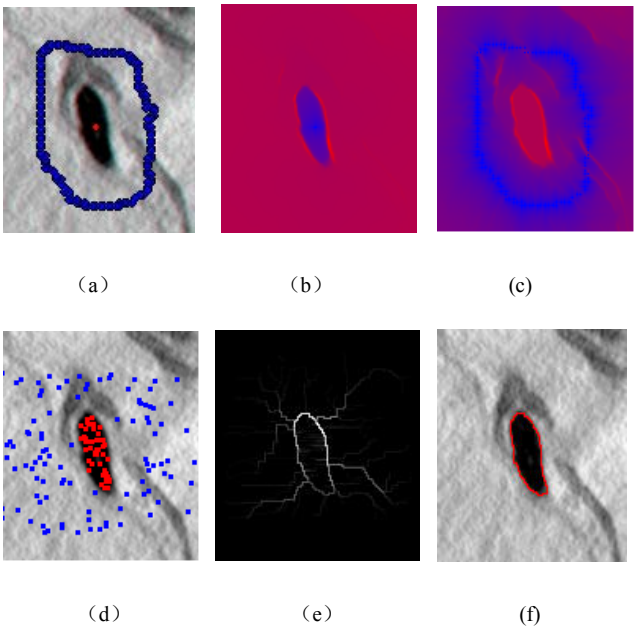


Figure 9. Illustration of our algorithm on a cardiac cell with 125*145 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The computed edge confidence image by adding all shortest paths together. (f) The final segmentation result.

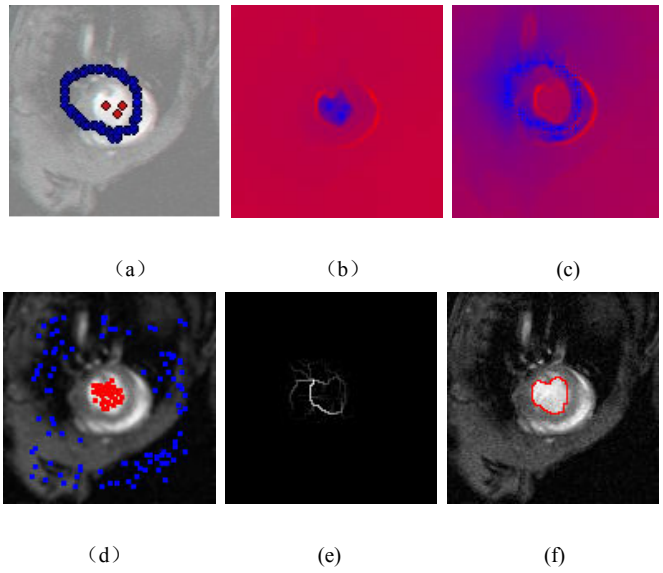


Figure 10. Illustration of our algorithm on an MRI image of the heart with 128*128 pixels. (a) Initial object (in red) and background (in blue) seeds picked by the user. (b) The marching distance map from the object seeds. (c) The marching distance map from the background seeds. (d) The randomly generated source (in red) and sink (in blue) points by using the marching distance maps. (e) The enhanced edge confidence map. (f) The final segmentation result of the left ventricle.

Figure 10 shows the performance of our approach on an MRI image of a mouse's heart. In order to separate the two closely-located features (namely, the left ventricle near the center surrounded by the right ventricle), we must carefully choose the initial seeds for the background, as shown in Figure 10 (a). As in the other examples, we show the marching distance maps starting from the object and background seeds in Figure 10(b) and (c) respectively. Figure 10(d) demonstrates the randomly generated source (red) and sink (blue) points, from which a number of shortest paths are found and used to enhance the edges, as shown in Figure 10(e). The final segmentation of the left ventricle is given in Figure 10(f). In this example, we randomly generated 50 source (object) points and 100 sink (background) points, and the total time for calculating shortest paths is 0.29 second.

IV. CONCLUSION

In the present paper we described a new graph-based algorithm to enhance the edges of a feature in an image by using the shortest paths of random source and sink points. The enhanced edge confidence image is then used to perform the segmentation by using the shortest path detection again. Our method is like the traditional Monte Carlo method in that the resulting edge confidence image and hence the final segmentation result become more accurate as the number of random shortest paths increases. Therefore, our approach is very scalable and can be very easily implemented in parallel computing. Our future work will be refinement of the proposed method in images with multiple features of interest.

REFERENCES

[1] E.N. Mortenson and W.A. Barrett. Interactive segmentation with intelligent scissors. Graphical Models and Image Processing, pages 349-384, 1998.

[2] C Rother, V Kolmogorov. Grabcut: Interactive foreground extraction using iterated graph cuts . ACM Transactions on Graphics, vol. 23, pp. 309–314, 2004.

[3] Gonzalez, R.C. and R.E. Woods, *Digital image processing*, Addison-Wesley, 1992.

[4] Z. Yu and C. Bajaj, Image Segmentation Using Gradient Vector Diffusion and Region Merging, in Proceedings of International Conference on Pattern Recognition, pp. 941-944, 2002.

[5] R. Malladi, J.A. Sethian, and B.C. Vemuri, *Shape modeling with front propagation: A level set approach.* IEEE Trans. Pattern Anal. Machine Intell.,. 17(2), pp. 158-175, 1995.

[6] N. Volkmann, A novel three-dimensional variant of the watershed transform for segmentation of electron density maps. J Struct Biol,. 138, pp. 123-129, 2002.

[7] J. Shi and J. Malik, *Normalized cuts and image segmentation.* Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 731-737, 1997.

[8] A.S. Frangakis et al., Identification of macromolecular complexes in cryoelectron tomograms of phantom cells. Proc Natl Acad Sci U S A, 99(22): pp. 14153-14158, 2002.

[9] Y. Boykov, G. Funka-Lea. Graph Cuts and Efficient N-D Image Segmentation, International Journal of Computer Vision, vol. 70, no. 2, pp. 109-131, 2006.

[10] A. Delong, Y. Boykov. A Scalable graph-cut algorithm for N-D grids, Proceeding of Computer Vision and Patten Recognition , pages 1-8, 2008.

[11] O. Juan, Y. Boykov. Active graph cuts, Proceeding of Computer Vision and Patten Recognition, pages 1023-1029, 2006.

[12] L. Grady. Random walks for image segmentation.. IEEE Transactions on Pattern Anal. Machine Intell., 28(11):1768–1783, 2006.

[13] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In Proceedings of International Conference on Computer Vision, pages 1-8, 2007.

[14] A. K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In Proceedings of International Conference on Computer Vision, pages 1-8, 2007.

[15] C. Couprie, L. Najman, L. Grady, and H. Talbot, Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest, In Proceedings of International Conference on Computer Vision, pages 731-738, 2009.

[16] Z. Yu and C. Bajaj, *Automatic Ultrastructure Segmentation of Reconstructed Cryo-EM Maps of Icosahedral Viruses.* IEEE Transactions on Image Processing,. 14(9): pp. 1324-1337, 2005.

[17] R. Malladi and J.A. Sethian. A real-time algorithm for medical shape recovery, in *Proc. of Int'l Conf. on Computer Vision*, pp. 304–310, 1998.