Foram Segmentation

Abstract—The species identification of single-celled marine organisms, foraminifera, poses a challenge as the process can become quite tedious. So, there is a need to create an automatic visual identification system. The foraminiferal species are differentiated from each other from their shell characteristics like aperture location and chamber shape and arrangement. So, several 3D and visual based features are extracted from a dataset of foram images and are used to obtain the edge probability map. This edge probability map provides the basis for segmenting the image. We propose a Markov Random Field (MRF) and computer vision based approach for segmentation using edge probability maps as input.

Index Terms—Graph-Cut algorithm, Segmentation

1 Introduction

Foraminifera are single-celled marine organisms, which are usually less than 1 mm in diameter. One of the most common tasks associated with foraminifera is the species identification of thousands of foraminifera contained in rock or ocean sediment samples, which can be a tedious manual procedure. Thus an automatic visual identification system is desirable. The foraminiferal species are differentiated from each other from their shell characteristics like aperture location and chamber shape and arrangement.

In this paper, we use a Markov Random Field (MRF) [1] and computer vision based technique for foraminifera image segmentation on a foraminifera dataset given to us. Segmentation is the partitioning of an image into multiple regions. It is used to represent an image in a form that is more meaningful and easier to analyze. Precisely, image segmentation assigns a label to every pixel in an image such that the pixels with the same label share certain common characteristics.

A graph is a collection of data-points in the form of edges and nodes. In traditional graph cut [2] approach, the image is segmented into foreground and background (which are previously specified by the user). Each of the pixels of the image is considered as a node which can be either 4-connected or 8-connected to its neighbors. In addition, 2 more nodes are added: the source and target, which are connected to all the pixels in the image. The weights from the source and the target nodes to all other nodes in the image determines the segmentation region. We use the min-cut/max-flow technique for segmenting the coarse edge probability map. Once we get the binary segmented edge probability map, we use morphology operations like closing to make the image even finer and also to connect broken segments. This refined map is then passed to watershed algorithm for thresholding.

The entire pipeline used in our experiment is illustrated in Fig. 8 and some sample segmentation results are shown in Fig. 1. The experiments demonstrate our approach is able to segment chambers and apertures of foraminifera correctly and has the potential to provide useful features for species identification. The remainder of this paper is organized as follows: Section II gives an overview of the related work; Section III explains our methodology. Section IV gives a detailed description of our segmentation approach; Section V explains the parameter selection and validation. Experiments of our approach as well as the analysis of the results are discussed in section VI; Section VII summarizes the report and provides a conclusion based on the overall analysis described in the report.

2 Literature Review

Felzenszwalb et al., in [3] address the problem of segmenting an image into regions. They define a predicate for measuring the evidence for a boundary between two regions using a graph-based representation of the image. They then develop an efficient segmentation algorithm based on this predicate, and show that although this algorithm makes greedy decisions it produces segmentations that satisfy global properties. They apply image segmentation algorithm using two different kinds of local neighborhoods in constructing the graph and illustrate the results with both real and synthetic images. The algorithm runs in time nearly linear in the number of graph edges and is also fast in practice.

Shepherd et al., in [4] compare the largest cohort to date of established, emerging and proposed PET contouring methods, in terms of accuracy and variability. They emphasize spatial accuracy and present a new metric that addresses the lack of unique ground truth. Thirty methods are used at 13 different institutions to contour functional volumes of interest in clinical PET/CT and a custom-built PET phantom representing typical problems in image guided radiotherapy. Contouring methods are grouped according to algorithmic type, level of interactivity and how they exploit structural information in hybrid images. Method-wise evaluation identifies the danger of over-automation and the value of prior knowledge built into an algorithm.

These works show good performance in image segmentation using a Markov Random Field (MRF) approach along with contouring. We aim to explore such techniques for the task of identifying foraminifera.

3 Methodology

A Foraminifera species is identified by its shell characteristics such as aperture location and chamber shape and

arrangement. There is a dearth of accurate segmentation methods for foraminifera segmentation. This is mainly due to the coarseness of the edge probability map as shown in figure 1.

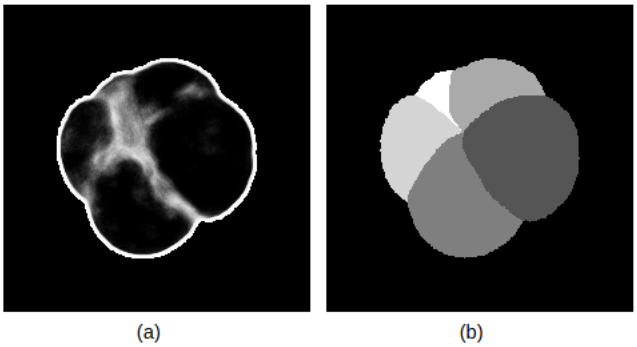

Fig. 1. (a) Edge Probability Map (b) Labeled Image

In this paper, we discuss a solution to this problem which uses an MRF approach, called as the graph cuts technique along with morphological image transformation techniques and multi-segment segmentation with watershed approach.

3.1 Graph cuts Binary Segmentation

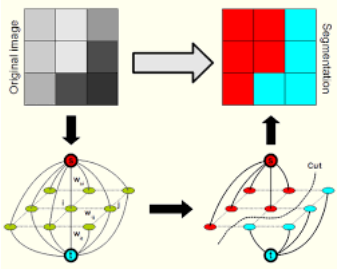

Fig. 2. Binary Segmentation using Graph-Cut [5].

3.1.1 Theory

In binary segmentation [6], Fig. 2, the image is partitioned into two different regions, one belonging to the foreground and the other belonging to the background. We introduce P as the set of pixel indices and a binary variable for each pixel such that $x_p \in B^p$ for every $p \in P$. When $x_p = 0$, pixel p is a background pixel and when $x_p = 1$, the pixel is in the foreground. The intensity of the pixel p is given by $I(p)$. The segmentation problem can be formulated in the form of minimizing the energy function which is given in Equation 1.

$$E(x) = \sum_{p \in P} D_p(x_p) + \lambda \sum_{p,q \in N} V_{p,q}\,(x_p, x_q) \qquad (1)$$

The **Data term** D_p adds a cost dependent on how different $I(p)$ is compared to the expected value of either the background E_b or the object E_o. $D_p(x_p)$ is computed as in Equation 2.

$$D_p(x_p) = \begin{cases} I(p) & \text{if } x_p = 0 \\ 255 - I(p) & \text{if } x_p = 1 \end{cases} \qquad (2)$$

D_p is low when assigning label 0 to dark pixels or when assigning 1 to bright pixels and high otherwise.

The **regularization term** $V_{p,q}$ adds a cost if neighboring pixels are similar but have different labels.

The parameter $\boldsymbol{\lambda}$ [7] controls the relative weight of the data term D versus the smoothness term V.

3.1.2 Priors

If we know the labels of some of the pixels a priori, then we can use this information to calculate an estimate of the expected value of the foreground E_o or the background E_b. The information can also be added to the graph by adding edges from the source and the target to the pixel under consideration with a cost of infinity based on whether it belongs to the foreground or the background. By adding these edges, we prohibit cuts which will contradict our a priori knowledge.

3.1.3 Graph Construction

For a min-cut on any graph, we set $x_p = 0$ for all vertices connected to the source and $x_p = 1$ for all vertices connected to the sink.

The energy function $E(x)$ can be reformulated into a min-cut problem where the minimal cut leads to the same labeling as the global minimum of $E(x)$.

3.2 Morphological Technique

There are two basic morphological operators: erosion and dilation. These operators are usually applied in tandem.

Erosion of a grey-level image F by another structuring element B, denoted $F \ominus B$, is defined in the following equation 3:

$$F \ominus B(m,n) = \min(F(m+s, n+t) - B(s,t)) \qquad (3)$$

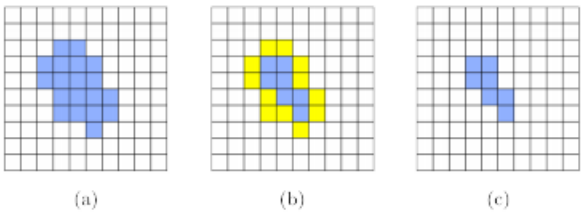

Fig. 3. Erosion

Erosion is a shrinking operator in the values of $F \ominus B$ are always less than or equal to the values of F. Dilation of a grey-level image F by another structuring element B, denoted $F \oplus B$, is defined by the following equation 4:

$$F \oplus B(m,n) = \max(F(m+s, n+t) + B(s,t)) \qquad (4)$$

Dilation is a expansion operator in the values of $F \oplus B$ always larger than or equal to the values of F.

Closing operations can be thus defined in terms of erosion and dilation. Closing of a data sequence by a structuring element is defined as dilation followed by erosion.

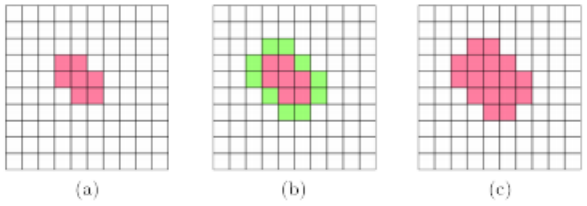

Fig. 4. Dilation

Closing of a grey-level image F by another structuring element B, denoted $F \bullet B$, is defined in following equation 5:

$$F \bullet B = (F \oplus B) \ominus B \tag{5}$$

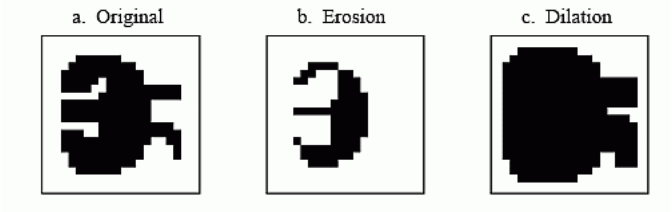

Fig. 5. a. Original b. Erosion c. Dilation

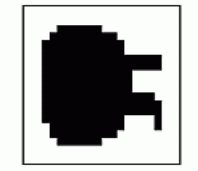

Fig. 6. Closing

Closing of a data sequence can be interpreted, as shown in Fig. 6 [8], as sliding a flipped-over version of the structuring element along the data sequence from above and the result is the lowest points reached by any part of the structuring element. Generally, closing can fuse narrow breaks, fill up small holes, and as well as gaps in the sketch.

3.3 Distance Transform

The Distance Transform [9] is an operator normally only applied to binary images. The result of the transform is a gray level image that resembles the input image, except that the gray level intensities of the points inside foreground regions are changed to show the distance to the closest boundary from each point. These form the prior boundaries for the Watershed [10] algorithm.

3.4 Watershed Algorithm

Any grayscale image can be viewed as a topographic surface where high intensity denotes peaks and hills while low intensity denotes valleys. The watershed transformation treats the image it operates upon like a topographic map,

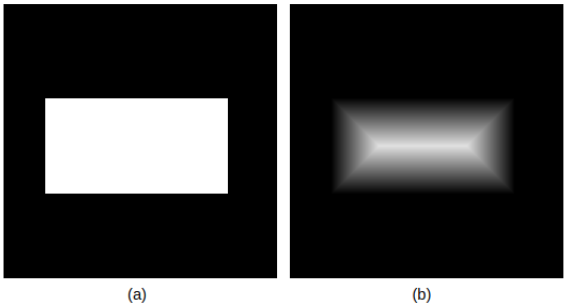

Fig. 7. Distance Transform (a) Original Image (b) After applying Distance Transform

with the brightness of each point representing its height, and finds the lines that run along the tops of ridges. In graphs, watershed lines may be defined on the nodes, on the edges, or hybrid lines on both nodes and edges.

4 Implementation

The Image segmentation approach is divided into 3 stages: Segmentation of edge pixels, Morphological refining of the edge map and segmentation into multiple chambers. We depict a pipeline of these stages in Figure 8. We discuss these stages in the following sub-sections.

4.1 Segmentation of edge pixels

In the first stage, we use the coarse edge probability map and perform a binary segmentation using graph cuts to get a binary edge map with clearly defined edge and non-edge pixels.

We use the maximum flow algorithm [11] which is widely used to minimize energy functions like the one tackled in this problem. We first build a graph representing this energy. This graph is composed of non-terminal nodes which represent all the pixels of the image. The nodes are connected in a grid arrangement, so that the nodes corresponding to neighbor pixels are connected by a forward and a backward edge. The weights of all the non-terminal edges is set as λ and the weights of edges from the source node are set to $D_p(0)$ and edges to the sink node are $D_p(1)$. We then perform the maxflow computation and obtain the segmentation labels.

4.2 Morphological refining of the edge map

In the second stage we perform a thinning operation on the binary edge maps obtained in stage 1 and follow it by performing morphological closing to remove discontinuities in chmaber boundaries.

We use the Zhang-Suen thinning algorithm [12] to reduce the thickness of coarse edge maps. The brief description of the algorithm is explained as follows:

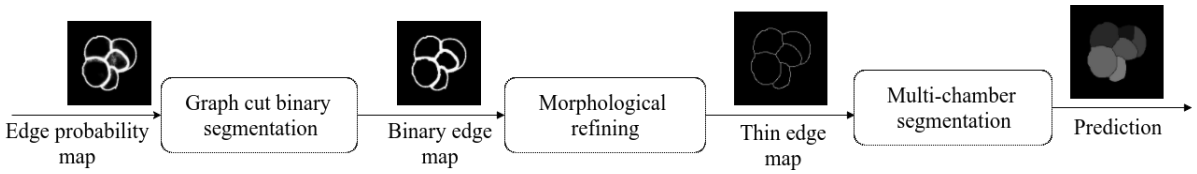

Fig. 8. Pipeline of image segmentation for foraminifera detection.

Algorithm 1 Zhang-Suen thinning Algorithm

Initialization
N(P1) = the number of non-zero pixel neighbours of P1 (= sum(P2 .. P9));
S(P1) = the number of transitions from 0 to 1, ($0 \rightarrow 1$) in the sequence P2,P3....P9,P2;

Step 1:-
Condition 0: The pixel is 1 and has eight neighbors
Condition 1: $2 \leq$ N(P1) ≤ 6
Condition 2: S(P1) = 1
Condition 3: P2 * P4 * P6 = 0
Condition 4: P4 * P6 * P8 = 0

Step 2:-
Condition 0: The pixel is 1 and has eight neighbors
Condition 1: $2 \leq$ N(P1) ≤ 6
Condition 2: S(P1) = 1
Condition 3: P2 * P4 * P8 = 0
Condition 4: P2 * P6 * P8 = 0

Iteration:-
If any pixels are set in either step 1 or step 2, then all the steps are repeated until no image pixels are changed.

The Zang-Suen algorithm has the highest enforcement efficiency. Additionally, a median filter can also be used prior to thinning for removel of salt and pepper noise.

4.3 Segmentation into multiple chambers

In the third stage, we use the thin edge maps to obtain the segmentation of the image into its chambers using the watershed segmentation algorithm.

We first invert and threshold the thin edge maps. We follow this by computing the Euclidean Distance Transform (EDT). The EDT is the distance of each non-zero foreground pixel to the closest background pixel. We then find peaks in the distance map we generated from EDT. We then tune a parameter d which is the minimum pixel distance between each peak.

We then apply a connected component analysis using 8-connectivity, which gives the priors or markers that the watershed algorithm requires to perform segmentation. Watershed algorithm expects local minima in the distance map. We thus pass it the negative value of the distance map. The watershed algorithm returns the labels of all pixels.

4.4 Toolboxes

We use the PyMaxFlow python library to perform graph cut segmentation.

For image processing required for thinning of the image, we used the Image Processing Scikit toolbox Scipy [13] in Python. We use OpenCV [14] for image operations like thresholding and morphological operations to perform thinning of edges.

5 Parameter Selection and Validation

In the graph-cut algorithm, we perform parameter selection on λ which controls the relative weight of the data term versus the smoothness term. Choosing a small λ leads to over-segmentation while a large λ leads to under-segmentation. We choose the value of λ through trial and error by varying it in the range of 1 to 100 with a step size of 25 [7].

5.1 Kernel size selection

The kernel size used during morphological operations for thinning plays a major role in the performance of the algorithm. The obtained thin edge map must be the closest possible representation of the original probability map. So we test the cross-correlation of all the edge probability maps with their thin edge maps for different kernel sizes. The kernel size which gives the highest cross-correlation ratio is taken as the optimum kernel size for further evaluation. The cross-correlation ratio of the thin edge maps with their probability maps is shown in figure 9

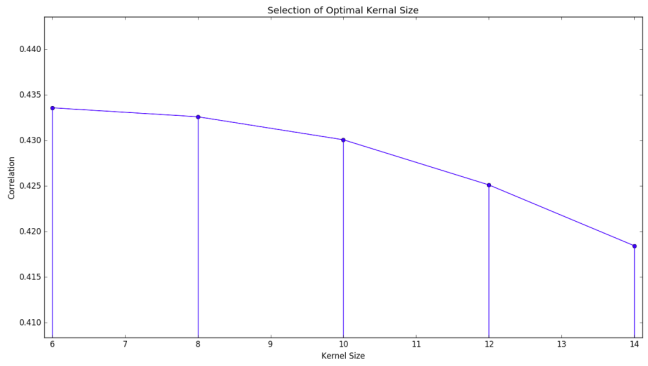

Fig. 9. Cross- Correlation between edge probability map and thin edge map for various kernel sizes

From the graph, it is clear that the thin edge maps are highly correlated with the original edge probability maps for kernel size of 6x6. So we select it as the optimum kernel for further analysis.

5.2 Error Metric

We use the following covering metric which takes the overlap of two regions R and R′ as:-

$$O(R, R') = \frac{|R \cap R'|}{|R \cup R'|'} \quad (6)$$

The covering of a segmentation S (the groundtruth) by a segmentation S′, is given as:-

$$C(S, S') = \frac{1}{N_s} \sum_{R \in S} |R|. \max_{R' \in S'} O(R, R') \quad (7)$$

Here, N_s is the total number of pixels in the groundtruth segmentation S not including the background region. The Covering Score (C(S,S′)) gives a measure of the degree of closeness of original labelling and the implemented segmentation.

6 Experimental Results

6.1 Dataset

The dataset consists of 320 samples. Every sample's coarse edge probability map has been given to us. We are also provided with corresponding labels (segmented image) for the 320 samples for cross validating our techniques. The coarse edge probability map and final segmented image given to us were processed in a complex way. FIrst Each sample consists of 16 images taken under different light source directions through a microscope with 30x magnification. Edge features are extracted from 2D images as well as a 3D reconstructed surface computed from the 16 images and are properly fused. A convolutional neural network (CNN) is trained to extract features from the coarse edge probability map.Another random forest is trained to refine the coarse edge map and the final segmentation is obtained by post-processing on the refined edge map.

6.2 Graph cuts binary segmentation

In this section, we report the results of binary segmentation on the edge probability maps to obtain a binary edge map.

We vary the regularization parameter λ from a small value 1 to a large value of 100 in steps of 25. We show the binary edge maps generated for one of the images for each value of the regularization parameter in Figure 10.

We observe that for λ = 1 and 25, the binary edge map has ragged edges which could potentially lead to a poor chamber segmentation. λ = 50, 75 and 100 produce smooth edges. We observe the same trend on all the images in the training set. We thus set λ = 50 for the next 2 stages.

6.3 Morphological refining of edge map

In this section, we report the results of morphological transformations on images to obtain a refined thin edge map. We first perform a thinning operation as explained in section 4.2 and follow it by a morphological closing operation to close the chambers whose boundaries are discontinuous. We vary the kernel size (window size) from 4×4 to 12×12 and report the resulting thin edge maps for one of the sample images in Figure 11.

We observe that the thin edge map obtained for a kernel size of 12×12 loses information of one of the small chambers while the one obtained with a kernel size of 4×4 has an extra chamber. This could be attributed to the fact that a larger kernel size could potentially miss some of the smaller boundary transitions when they get averaged while a very small kernel size could lead to transitions that do not exist. We observe the same statistics for other images in the training set. We thus choose a kernel size of 6×6 as the optimal size.

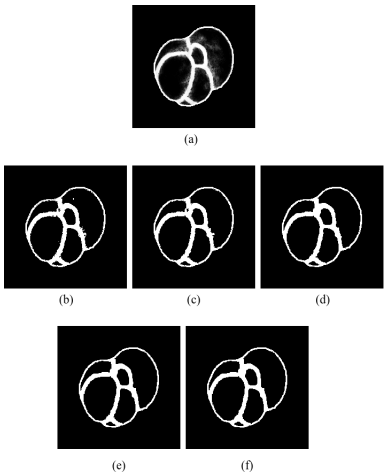

Fig. 10. (a) is the probability edge map and (b), (c), (d), (e) and (f) are the binary edge maps from graph cuts with λ 1, 25, 50, 75 and 100 respectively.

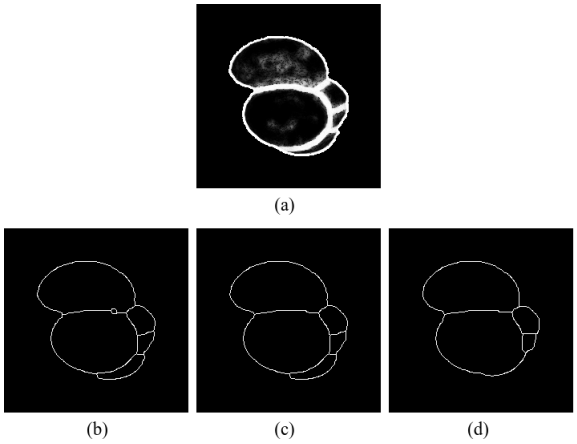

Fig. 11. (a) is the probability edge map and (b), (c) and (d) are the thin edge maps for kernel sizes 4×4, 6×6 and 12×12 respectively.

6.4 Segmentation into multiple chambers

In this section, we report the results of segmentation of the thin edge maps into chambers.

The performance of the watershed algorithm is dictated by the priors. To obtain accurate priors, we vary d, the distance between peaks in the distance map from 5 to 15 and compute the watershed segmentation. We show the results in Figure 12

We observe that a very low d of 5 leads to over-segmentation while a value of 15 leads to under segmentation where some of the chambers are missing. A d of 10 is a good trade-off to obtain a good segmentation with low error. The reason for this performance could be due to a higher d leading to overshooting peaks of some small chambers,

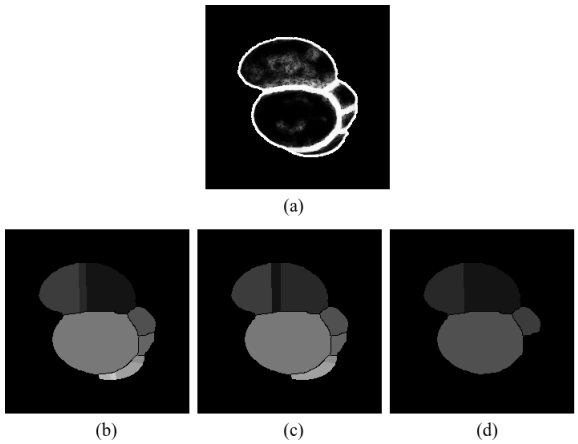

Fig. 12. (a) is the probability edge map and (b), (c) and (d) are the segmentation outputs for peak distances 5, 10 and 15 respectively.

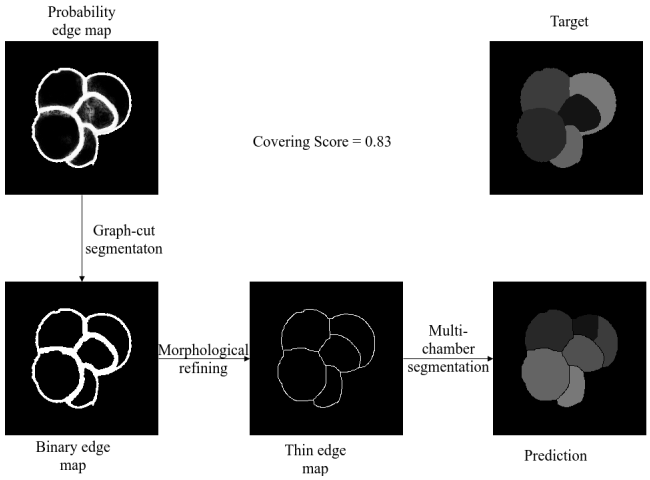

Fig. 13. Edge probability map, targets and output of all the 3 stages.

while a lower d could be finding peaks that are within a chamber.

6.5 Performance of our image segmentation technique

We show the results of image segmentation of a few images from the dataset and also report their covering scores along with the labels. We also show the output of the intermediary stages in Figure 13, Figure 14, Figure 15 and Figure 16. For all the cases, we choose λ = 50, kernel size as 6×6 and distance d = 10.

The predictions show that our segmentation technique gives a high covering score when there are large chambers while over segmenting when there are smaller chambers with vague boundaries.

On scoring all the 320 training predictions, we obtained an average covering score of 0.75 or 75%. We plot a histogram of the covering score for all samples in training set in Figure 17.

From the histogram, we observe that a majority of the samples have a covering score over 0.6. This validates our implementation of image segmentation.

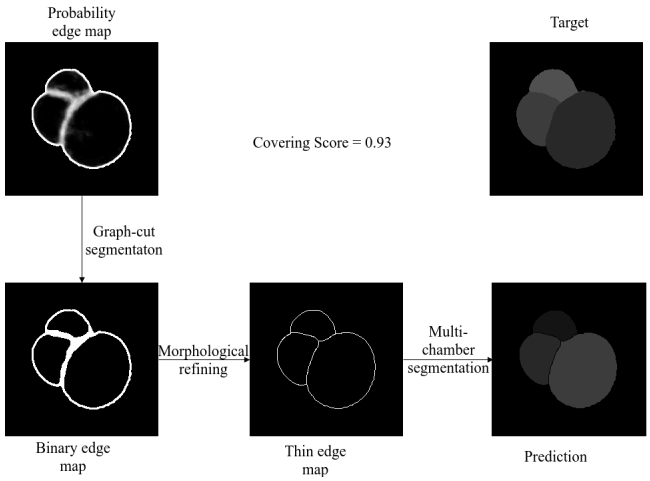

Fig. 14. Edge probability map, targets and output of all the 3 stages.

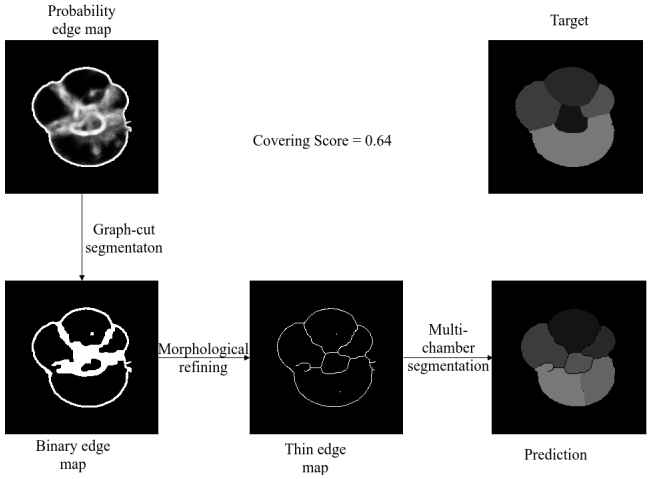

Fig. 15. Edge probability map, targets and output of all the 3 stages.

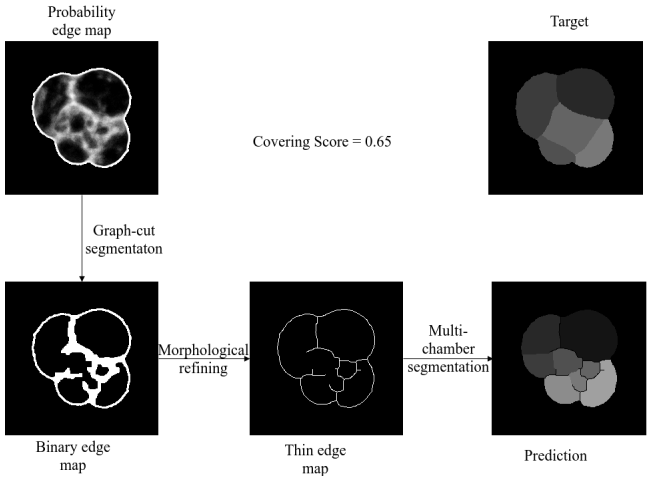

Fig. 16. Edge probability map, targets and output of all the 3 stages.

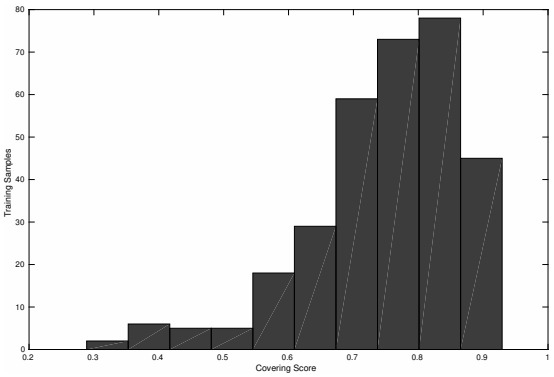

Fig. 17. Histogram of covering score for training set.

7 Conclusion

So, we segmented the chambers of Foraminifera images from their probability maps with decent accuracy. We used Graph cut segmentation to get thin probability maps from the more coarse maps. With some pre-processing we applied Distance transformation and Watershed algorithm to obtain the chamber segmented image of the probability map.

References

[1] S. Z. Li, "Markov random field models in computer vision," in *European conference on computer vision*. Springer, 1994, pp. 361–370.

[2] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 309–314.

[3] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[4] T. Shepherd, M. Teras, R. R. Beichel, R. Boellaard, M. Bruynooghe, V. Dicken, M. J. Gooding, P. J. Julyan, J. A. Lee, S. Lefèvre *et al.*, "Comparative study with new accuracy metrics for target volume contouring in pet image guided radiation therapy," *IEEE transactions on medical imaging*, vol. 31, no. 11, pp. 2006–2024, 2012.

[5] "Graph cuts," http://mygsoc.blogspot.com/2013/07/graph-cuts.html, accessed: 2017-11-30.

[6] A. Delong and Y. Boykov, "Globally optimal segmentation of multi-region objects," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 285–292.

[7] B. Peng and O. Veksler, "Parameter selection for graph cut based image segmentation." in *BMVC*, vol. 32, 2008, pp. 42–44.

[8] "MS Windows NT kernel description," http://www.dspguide.com/ch25/4.htm, accessed: 2017-11-30.

[9] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 529–533, 1995.

[10] K. Haris, S. N. Efstratiadis, N. Maglaveras, and A. K. Katsaggelos, "Hybrid image segmentation using watersheds and fast region merging," *IEEE Transactions on image processing*, vol. 7, no. 12, pp. 1684–1699, 1998.

[11] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[12] W. Chen, L. Sui, Z. Xu, and Y. Lang, "Improved zhang-suen thinning algorithm in binary line drawing applications," in *Systems and Informatics (ICSAI), 2012 International Conference on*. IEEE, 2012, pp. 1947–1950.

[13] E. Jones, T. Oliphant, and P. Peterson, "others. scipy: Open source scientific tools for python. 2001," *URL http://www. scipy. org*, 2016.

[14] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.