Foraminifera Image Segmentation- Part 2

***Abstract*—The classification of the species of the Foraminifera is based on visual information about the number of chambers in the intensity map of the marine organism. This project aims to achieve image segmentation with the use of Markov Random Field. The first part of the project will involve using a clustering mechanism to find the pixel clusters corresponding to segments. Once we have the initial segments, we will further process it using Hidden Markov Field's EM algorithm to get a refined segmentation result.**

I. Introduction

The scientific endeavor to understanding life on earth, involves classification of them into the different species. The classification of one particular marine organism, the Foraminifera found in rock and ocean sediment samples, involves visually looking at the images of them. The classification is done based on the number of chambers of the organism. This can, however, be automated with the use of machine learning and image segmentation, to make the whole process a lot less tedious. This project aims to do the same using a Markov Random Field based Image Segmentation.

II. Block diagram and Methodology

The block diagram of our implementation is shown in figure 1. We initially pre-process the image and make it ready for the segmentation. Pre-processing includes edge detection and noise removal. We have used canny edge detector to detect edges efficiently. For removing noise from the image, we blur the image by passing it through filters. We have chosen Gaussian filter for this sake. More details about these are provided in further sections.

Actual segmentation is done with Hidden Markov Random Fields by feeding them with processed probability maps. More information about out model and algorithm is provided in subsection E.

We get the class labels from segmentation model. We post-process the image to eliminate smaller segments and fill the regions to visualize them. We describe the functions that we used for these processing in further sections.

A. Image Dataset

The input data set consists of a set of 230x230 pixel probability maps that encodes the data regarding the probability of finding an edge at any particular pixel. This is expressed in the range of $\{0,255\}$, and hence could potentially be interpreted as the intensity of the pixel. We could also potentially pre-process the image to help identify the edges better.One of the probability maps is shown in figure 2.

B. Pre-Processing

The probability maps are pre-processed to obtain better results. We first extract the edges from the probability map. For this, we used Canny edge detection algorithm. The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images [8]. We got motivated for using this algorithm due to the fact that it fulfills the main criterion for edge detection, namely, (i) high recall on number of edges, (ii) low error in displacement of edge-center, and (iii) low number of false positive edges. Thus extracted edges are shown in figure 3.

We then filter the image by passing it through Gaussian filter by selecting filter width equal to 3. This number was selected by experimenting with various values and then using the best fit. The filtered image is shown in figure 4.

C. Coarse to Fine Edge Probability Map Extraction

We need to refine coarse edge probability maps to obtain high precision edges. Coarse edge map could be noisy to clearly differentiate among chambers and apertures. Hence we need to process this to obtain a refined map. This cannot be achieved by simple thresholding process. We use suitable classifiers to refine this.

D. Clustering

After we have a filtered edge map from the gaussian filter, we see the non-edge points still forming a cloud. We need to refine it to cluster the pixels near the edges which have high probability values. We use k-means clustering for the same, and since we are doing this in an unsupervised way, we use a binary clustering which identifies only if the pixel has to be identified or not. So, k=2 is set. The output obtained is shown in Figure 5.

E. Markov Random Field

Markov random fields (MRFs) have been widely used for computer vision problems, such as image segmentation, surface reconstruction and depth inference. Much of its success attributes to the efficient algorithms, such as Iterated Conditional Modes, and its consideration of both data faithfulness and model smoothness.

The HMRF-EM [9] framework was first proposed for segmentation of brain MR images. Given an image $\mathbf{y} = (y_1, ..., y_N)$ where each y_i is the intensity of a pixel, we want to infer a configuration of labels $\mathbf{x} = (x_1, ..., x_N)$ where $x_i \epsilon L$ and L is the set of all possible labels. In a binary segmentation problem, $L = \{0, 1\}$. According to the MAP criterion, we seek the labeling x^* which satisfies

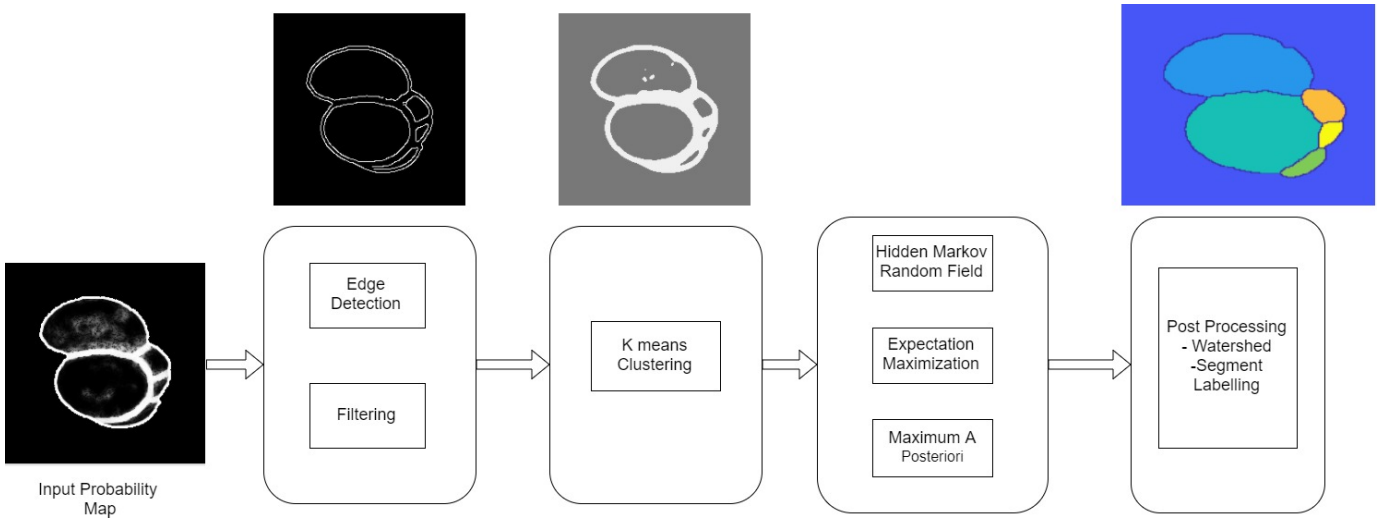

Fig. 1: Block Diagram

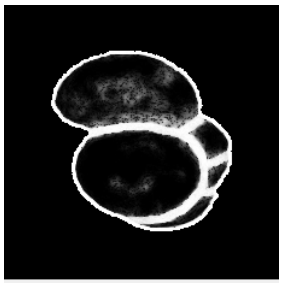

Fig. 2: Probability map

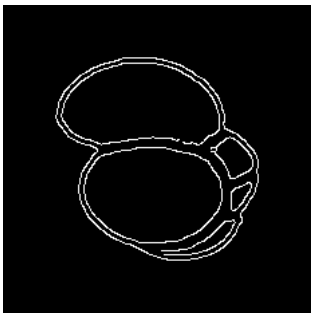

Fig. 3: Edges extracted from probability maps

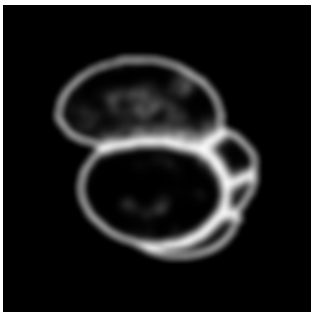

Fig. 4: Gaussian blurred probability map

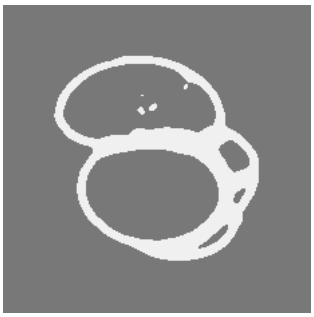

Fig. 5: Clustering Using Kmeans

$$x^* = argmax_x\{P(y|x,\Theta)P(x)\} \tag{1}$$

The prior probability P(x) is a Gibbs distribution, and the joint likelihood probability is

$$\begin{aligned} P(y|x,\Theta) &= \prod_i P(y_i|x,\Theta) \\ &= \prod_i P(y_i|x_i,\theta_{xi}), \end{aligned} \tag{2}$$

where $P(y_i, x_i, \theta_{xi})$ is a Gaussian distribution with parameters $\theta_{xi} = (\mu_{xi}, \sigma_{xi}).\Theta = \theta_l|l\epsilon L$ is the parameter set, which is obtained by the EM algorithm.

F. EM and MAP

To estimate the above parameter set Θ, we use the EM algorithm. The details of the algorithm similar to explained [9] are as below:

1) Start: Assume we have an initial parameter set $\Theta^(0)$
2) *E-Step*: At the *t*th iteration, we have $\Theta^(t)$, and we calculate the conditional expectation:

$$\begin{aligned} Q(\Theta|\Theta^t) &= E[lnP(\mathbf{x},\mathbf{y}|\Theta)|\mathbf{y},\Theta^t] \\ &= \sum_{\mathbf{x}\epsilon\chi} P(\mathbf{x}|\mathbf{y},\Theta^t) lnP(\mathbf{x},\mathbf{y}|\Theta), \end{aligned} \tag{3}$$

where χ is the set of all possible configurations of labels.

3) *M-step*: Now mamximaize $Q(\Theta|\Theta^t)$ to obtain the next estimate:

$$\Theta^{t+1} = argmax_{\Theta} Q(\Theta|\Theta^t). \tag{4}$$

Then let $\Theta^{t+1} -> \Theta^t$ and repeat from the E-step.

Let $G(z;\theta_l)$ denote a Gaussian distribution function with parameters $\theta_l = (\mu_l, \sigma_l)$:

$$G(z;\theta_l) = \frac{1}{\sqrt{2\pi\sigma_l^2}} exp(-\frac{(z-\mu_l)^2}{2\sigma_l^2}). \tag{5}$$

We assume that the prior probability can be written as

$$P(x) = \frac{1}{Z} exp(-U(x)), \tag{6}$$

where $U(x)$ is the prior energy function. We also assume that

$$\begin{aligned} P(y|x,\Theta) &= \prod_i P(y_i|x_i,\theta_{xi}) \\ &= \prod_i G(y_i;\theta_{xi}) \\ &= \frac{1}{Z'} exp(-U(y|x)). \end{aligned} \tag{7}$$

In the above algorithm, we need to solve for x^* that minimizes the total posterior energy

$$x^* = argmin_{x\epsilon\chi}\{U(y|x,\Theta) + U(x)\} \tag{8}$$

with given y and Θ, where the likelihood energy is

$$\begin{aligned} U(y|x,\Theta) &= \sum_i U(y_i|x_i,\Theta) \\ &= \sum_i [\frac{(y_i-\mu_{x_i})^2}{2\sigma_{x_i}^2} + ln\sigma_{x_i}]. \end{aligned} \tag{9}$$

The prior energy function U(x) has the form

$$U(x) = \sum_{c\epsilon C} V_c(x), \tag{10}$$

where $V_c(x)$ is the clique potential and C is the set of all possible cliques.

In the image domain, we assume that one pixel has at most 4 neighbors: the pixels in its 4-neighborhood. Then the clique potential is defined on pairs of neighboring pixels:

$$V_c(x_i, x_j) = \frac{1}{2}(1 - I_{x_i,x_j}), \tag{11}$$

where

$$I_{x_i,x_j} = \begin{cases} 0 & \text{if } x_i != x_j \\ 1 & \text{if } x_i = x_j \end{cases} \tag{12}$$

We use the iterative algorithm from [9] section 3, to solve 8

Similarly, we use the HMRF-EM algorithm as described in the paper [9] to finally apply the MAP algorithm as described above, and arrive at the parameters.

G. Post-Processing

Once the segmentation model is obtained as the output of HMRF, we process the image further to obtain a visually appealing output. We used a structuring element, a disk of width 1 to close the image. We also remove small objects from the segmented image which are probably not chambers or aperture. We use watershed [10], region filling algorithm to assign different colors to the image segments.

The final segmented image after post-processing is shown in figure 6

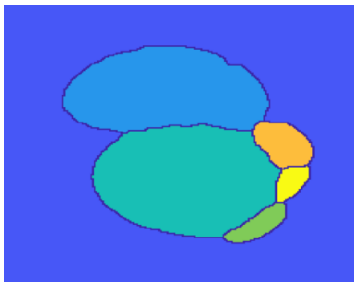

Fig. 6: Image segmented using our model

The actual labelled image is shown in figure 7

Fig. 7: Actual segmented image

III. Performance Metric

We use overlap covering score as the performance metric in evaluating model prediction. We use un-weighted covering score because for species identification purpose, every chamber and aperture have equal importance. To evaluate how well the chambers and apertures are detected, we define the recall of regions for each sample as the percentage of the labeling regions detected in the estimated segmentation, where each region in the human labeling is regarded as detected if there exists an overlapping estimated region with covering ratio greater than 0.5. This is very similar to the approach used in [1].

IV. Hyper-parameter Selection

The probability maps provided to us don't have class labels. But the final output needs to have classes associated with each of the segment. This is an unsupervised model because we don't know how many segments we are going to get in the final result. Hence we would need to do parameter estimation

by learning from the data itself. We are going to adopt Expectation Maximization as described in II-F for the same. EM assigns labels and estimates parameters simultaneously. For the given data, the E step of EM would first compute the probability distribution using the existent parameters. The M step would update the estimates of parameters based on weighted labelling.

A. Gaussian Filter Width selection

During the pre-processing step, we chose Gaussian filter width to be 3 based on some experimentation. The following figures - 8 9 10 show the outputs for various sigma that we experimented with.

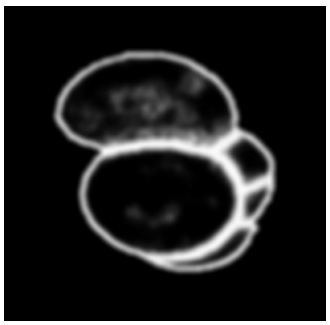

Fig. 8: Probability map passed through Gaussian filter with filter width = 2

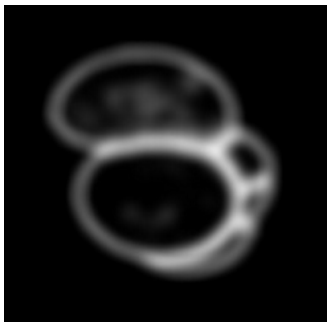

Fig. 9: Probability map passed through Gaussian filter with filter width = 4

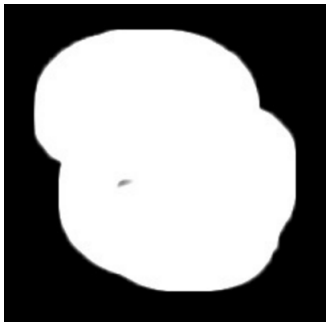

Fig. 10: Probability map passed through Gaussian filter with very high value of sigma

Based on these experimental results, we chose the filter width to be 3.

B. Canny Filter threshold

Similarly, based on some tuning, we selected the Canny filter threshold value to be 0.75. The following figures - 11 12 13 14 show the outputs for various threshold values we experimented with.

Fig. 11: Canny edge detection with threshold 0.1

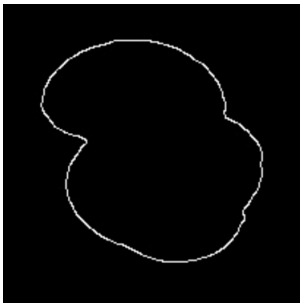

Fig. 12: Canny edge detection with threshold 0.9

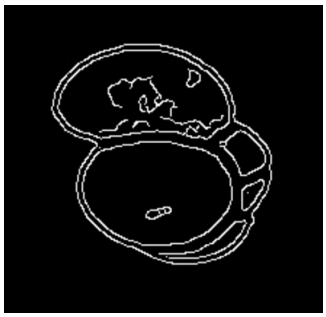

Fig. 13: Canny edge detection with threshold 0.5

V. Toolbox

We are using Matlab tools for doing the task. In matlab, we have used the functions GaussianBlur, Kmeans, Watershed and HMRF-EM by Quan Wang [9]. For the filter, we are using a canny filter, which is made by getting a filter through imfilter function.

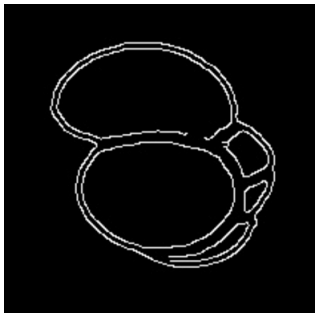

Fig. 14: Canny edge detection with threshold 0.8

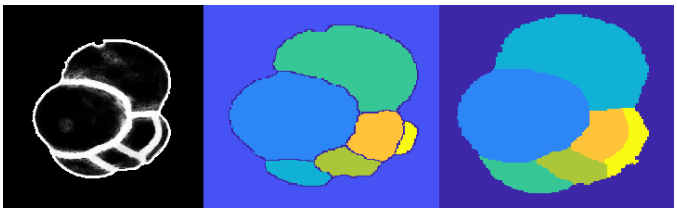

Fig. 15: (a) Probability map (b) Image segmented with our model (c) Labelled image output

VI. Results

The results of our model is shown in this section. Few images whose probability map has clear edges, give proper outputs. Few of the maps have some inconsistency in the edges. These maps do not result in great segmentation. The following figure 15 shows the probability map, predicted segmentation, actual labelled image of a sample.

Some images whose probability map don't give clear edges, have some error-ed segmentation output as shown in fig 16

We ran our model over all the images. The covering score of the training data for all the 320 images that were given is found to be 0.8124. We selected 10 images whose edges were properly defined in the probability map. The covering score for these images is found to be 0.8252. Similarly we ran our model on 10 images whose edges were not well defined and the probability maps were not discrete. The covering score for these images were found to be 0.7811.

References

[1] Coarse-to-Fine Foraminifera Image Segmentation through 3D and Deep Features by Qian Ge, Boxuan Zhong, Bhargav Kanakiya, Ritayan Mitra, Thomas Marchitto and Edgar Lobaton

[2] A Simple Unsupervised Color Image Segmentation Method based on MRF-MAP by Qiyang Zhao

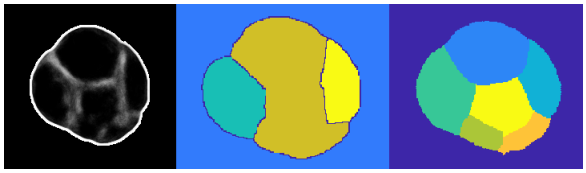

Fig. 16: (a) Probability map (b) Image segmented with our model (c) Labelled image output

[3] Zoubin Ghahramani, An introduction to hidden Markov models and Bayesian networks, Hidden Markov models: applications in computer vision, World Scientific Publishing Co., Inc., River Edge, NJ, 2001

[4] A Markov Random Field Image Segmentation Model Using Combined Color and Texture Features by Zoltan KatoTing-Chuen Pong

[5] https://www.tensorflow.org/

[6] https://github.com/fchollet/keras

[7] https://pystruct.github.io

[8] https://en.wikipedia.org/wiki/Canny_edge_detector

[9] Wang, Quan. "HMRF-EM-image: implementation of the hidden markov random field model and its expectation-maximization algorithm." arXiv preprint arXiv:1207.3510 (2012).

[10] https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html

[11] https://www.mathworks.com/help/images/ref/edge.html

[12] https://en.wikipedia.org/wiki/Sobel_operator

[13] https://www.mathworks.com/help/images/ref/watershed.html