A Novel Graphical Model Approach to Segmenting Cell Images

Shann-Ching Chen*, Ting Zhao*, Geoffrey J. Gordon† and Robert F. Murphy*†‡
Center for Bioimage Informatics, *Department of Biomedical Engineering, †Department of Machine Learning and ‡Department of Biological Sciences, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract— **Successful biological image analysis usually requires satisfactory segmentations to identify regions of interest as an intermediate step. Here we present a novel graphical model approach for segmentation of multi-cell yeast images acquired by fluorescence microscopy. Yeast cells are often clustered together, so they are hard to segment by conventional techniques. Our approach assumes that two parallel images are available for each field: an image containing information about the nuclear positions (such as an image of a DNA probe) and an image containing information about the cell boundaries (such as a differential interference contrast, or DIC, image). The nuclear information provides an initial assignment of whether each pixel belongs to the background or one of the cells. The boundary information is used to estimate the probability that any two pixels in the graph are separated by a cell boundary. From these two kinds of information, we construct a graph that links nearby pairs of pixels, and seek to infer a good segmentation from this graph. We pose this problem as inference in a Bayes network, and use a fast approximation approach to iteratively improve the estimated probability of each class for each pixel. The resulting algorithm can efficiently generate segmentation masks which are highly consistent with hand-labeled data, and results suggest that the work will be of particular use for large scale determination of protein location patterns by automated microscopy.**

I. Introduction

The first step in biological image analysis is often to segment each image into regions of interest. For example, in order to carry out automated analysis of protein subcellular location patterns [1], it is desirable to segment images into single cell regions. However, the variations of image patterns resulting from different specimen types and imaging techniques make segmentation a very difficult task [2]. In the cell segmentation problem, different types of segmentation methods can be applied, such as thresholding, region growing, edge-based segmentation, and seeded watershed segmentation [2].

When only an image of a nuclear marker is available, Voronoi segmentation is widely used to define cell regions [3]. However, the resulting segmentation masks could crop pieces off of individual cells when the cells are close to one another. When evidence about both cell centers and cell boundaries is available (such as from nuclear staining and staining of total protein or plasma membrane proteins), the seeded watershed algorithm [4] usually provides good cell boundaries [5]. However, this algorithm usually produces loose contours which cover irrelevant background regions in addition to the cell bodies [6]. In addition, it can be tricky to compute a good initial seeding for the watershed algorithm, and poor seeding can produce unsatisfactory results. Another class of methods is active contours, which numerically optimize an energy function which measures both the quality of the current boundary and the difference between image features inside and outside the contour [6][7]. This type of segmentation method can have excellent performance on images whose foreground and background regions have different statistical properties, but it can be quite computationally expensive.

In this paper, we propose a novel graphical model approach to tackle the cell segmentation problem. Graphical models have been extensively applied to many classification problems, such as hypertext classification [8] and image segmentation [9], where we need to infer values for many interdependent random variables. Our proposed graphical models are undirected Bayes nets, also called Markov networks or Markov random fields. The relationships between the random variables in these models are described by potential functions associated with groups of nodes called cliques. A classic potential function is the Potts potential [10], which is a *2-way* (pairwise) potential and can be used to represent multiplicative influences between random variables (e.g., each additional neighbor of the background class multiplies my odds of being a background pixel by 1.2). Here we instead use a voting potential [11], which is a *k-way* potential function that integrates information in an additive way (e.g., if two thirds of my neighbors are background pixels, my odds of being background go up by 50%). For inference we use the factor graph representation and the sum-product algorithm [12]; since our graphs contain cycles, this combination is often called loopy belief propagation. A similar voting potential has been demonstrated to significantly improve performance on a multi-cell image classification problem [13].

This paper is organized as follows. In Section II, we briefly explain the relationship between Bayes Net and the factor graph representation using different potential functions. We demonstrate two approximate inference algorithms based on the voting potential function and the factor graph representation. In Section III, we provide the derivations of message calculation in belief propagation using DNA and edge potentials. In Section IV, we compare our algorithm to the seeded watershed method on a collection of yeast cell images [14]. Finally, in Section V we discuss conclusions and future work.

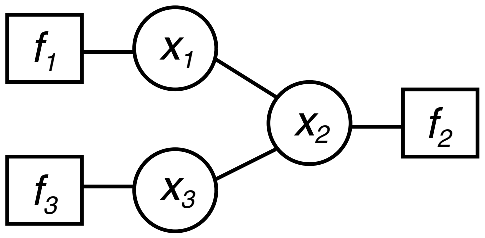

Fig. 1. A collection of three random variables x_1, x_2, x_3 arranged in a graph. Each circle represents the random variable x_i, and each square represents the feature vector f_i of the random variable x_i. An edge connecting two nodes means that their values are directly related to one another.

II. Graphical Model Representation

A. Problem Statement

We formalize the problem by assuming that two sources of information of the multi-cell image are available. One source describes the DNA content at each pixel; the other source provides an initial guess of the cell outlines. Each pixel in the image is considered as a random variable of either the background or foreground labels. We observe that one pixel is likely to belong to the foreground if its DNA intensity is relatively high. In addition, two neighboring pixels are likely to have the same labels if there is no edge between them. The segmentation task is: given an image of a field containing a number of cells with the above information, assign each pixel either to the background environment or to the foreground belonging to a specific cell in the field.

B. Bayes network

A Bayes net is composed of nodes and edges. Nodes represent random variables, and edges between nodes indicate the dependence between the random variables. We can arrange three random variables x_1, x_2, x_3 with associated feature vectors f_1, f_2, f_3 in a Bayes net as shown in Fig. 1.

The feature vector f can provide a probability distribution $P(x)$ over possible labels x as the observed evidence of the random variable. To encourage the labels of connecting random variables to be the same, we can use the following pairwise potential function:

$$\varphi(x_i, x_j) = \begin{cases} \omega & x_i = x_j \\ 1 & \text{otherwise} \end{cases}$$

where $\omega > 1$ is a free parameter which expresses how strongly we believe that x_i and x_j have the same label. The overall probability of a vector of labels x is:

$$P(x) = \frac{1}{Z} \prod_{\text{nodes } i} \varphi^{loc}(x_i) \prod_{\text{edges } i,j} \varphi(x_i, x_j) \tag{1}$$

where Z is a normalizing constant and $\varphi^{loc}(x_i)$ represents the evidence (derived from the feature vector f_i) of node i for every possible label. The above potential is called the Potts potential, and the Bayes net with this potential is called the Potts model [10] or Ising model [15].

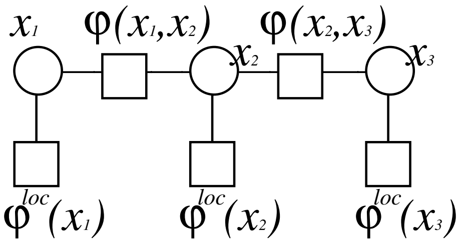

Fig. 2. Three random variables x_1,x_2 and x_3 in a Bayes net using the factor graph representation with pairwise potential function.

C. Potts Potential and Voting Potential

Although the Potts model can represent the dependence of the random variables in a simple and intuitive way, it does not perfectly capture the property about inference from labels of neighboring classes. The problem is that, in Eq. 1, the evidence combines several pair-wise potentials multiplicatively. While the evidence from different neighbors acts through separate potentials, an exponential dependence between the number of neighbors of a given class and the probability of that class results. If one use a partially ordered Markov model (POMM) [16], which is a directed version of the Potts model, the estimation of parameters and sampling from the model are easier than they would be for an undirected model. However, for inference, POMMs suffer from exactly the same problem that the Potts model does: the evidence combines the neighborhood potential functions multiplicatively. When this is not desirable, we can instead use an approach which combines the evidence from the neighboring variables additively. For example, one more intuitive model is to update the posterior probability of the random variable according to the proportion of its neighbors of each class:

$$\varphi_j(x_1, x_2, \ldots, x_k) = 1 + \sum_{i=1, i\neq j}^{k} \alpha I(x_i, x_j)(1 - \varphi_{boundary}(i)) + \sum_{i=1, i\neq j}^{k} \beta \overline{I}(x_i, x_k) \varphi_{boundary}(i)$$

where x_j is the pixel of interest and φ_j is the potential function centered at x_j. x_1, x_2,...x_k are x_j's neighbors, $\varphi_{boundary}(i)$ is proportional to the probability of the existence of a boundary between two neighboring pixels x_j and x_i. I is an indicator function which is 1 when x_i=x_j and 0 otherwise. The voting potential consists of two important terms which encourage different types of behavior. The first is when two pixels have the same labels $I(x_i, x_j)$ and low likelihood of a boundary between them $(1 - \varphi_{boundary}(i))$. This behavior is represented in the first summation term. The second summation term encourages differently labeled pixels if they have high boundary potential between them. α and β are scale factors for these two terms, and the larger they are the more strongly x_j's neighbors will influence its classification. This voting potential function combines the evidence from all

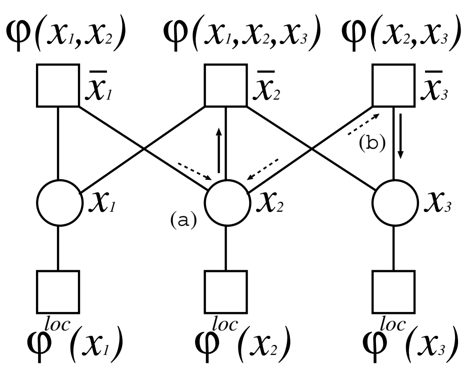

Fig. 3. Three random variables x_1,x_2 and x_3 in a Bayes net using the factor graph representation with voting potential function. (a)Message passing from a variable to a factor node. (b)Message passing from a factor node to a variable.

of node x_j's neighbors into a summary vote to influence its classification. The voting potential is defined in such a way that the message passing can be calculated efficiently by the sum-product algorithm described next.

D. Factor Graphs and Belief Propagation

A factor graph is a graph representation which can split a complicated global function of many variables into a product of several local functions, each of which only depends on a subset of the variables [12]. To calculate the posterior distribution of the random variables, a message passing procedure called the sum-product algorithm is used in a factor graph. A wide variety of algorithms based on the graphical model representation can be derived as the instances of the sum-product algorithm. The Bayes net in Fig. 1 can be represented as a factor graph with the pairwise potential function (Fig. 2).

In a factor graph, the circle nodes represent random variables and the square nodes represent the factor nodes. For example, $\varphi(x_1, x_2)$ is a factor node, a pairwise potential function of two random variables x_1 and x_2. $\varphi^{loc}(x_i)$ is the local evidence from the observation, which is only a function of the variable itself and is usually derived from some set of features describing the object represented by the variable.

The same Bayes net can be represented as another factor graph using the voting potential (Fig. 3). For each variable, we define an affiliated factor node associated with the random variable. This affiliated factor node connects to the centered variable and all the neighbors of the centered variable. We use a bar above the random variable to denote the affiliated factor node. For example, $\bar{x_2}$ connects to x_2 as well as x_1 and x_3 (all the neighbors). This factor node can be explained by collecting the message from the neighbors of x_2, and it is exactly the voting potential function $\varphi_2(x_1, x_2, x_3)$ centered at x_2. We termed x_2 as the centered variable and all other x_i as non-centered variables in all the following descriptions.

There are two types of messages we have to calculate: from a variable to a factor node and from a factor node to a variable. The message calculation from a variable to a factor node is simple. As shown in Fig. 3(a), we just collect all the messages from the neighboring factors (except the one we are sending to) and use the component-wise dot product to combine them with the local evidence. (In the usual implementation of this approach, the messages are initially set to equal probability for all classes.) Generally, the message from a variable x_j to a factor node $\bar{x_i}$ can be calculated as:

$$m_{j\rightarrow\bar{i}}(x_j) = \varphi^{\text{loc}}(x_j) \prod_{l=1,l\neq i}^{k} m_{\bar{l}\rightarrow j}(x_j) \tag{2}$$

The message calculation of the other type of message, from a factor node to a variable, is slightly more complicated (Fig. 3(b)). We have to collect messages from all the neighboring variables (except the one we are sending to) and marginalize out all the variables except the one are we interested in. In general, the message from a factor node $\bar{x_j}$ to a variable x_i can be calculated as:

$$m_{\bar{j}\rightarrow i}(x_i) = \sum_{\sim\{x_i\}} \varphi_j(x_1, \ldots, x_k) \prod_{l=1,l\neq i}^{k} m_{l\rightarrow\bar{j}}(x_l) \tag{3}$$

where $\sum_{\sim\{x_i\}}$ means that we marginalized out all the variables except x_i to calculate the marginal distribution of x_i. All these messages can be calculated very quickly by the sum-product algorithm [12], which decomposes the whole joint distribution into several smaller joint distributions of subsets of random variables so that the marginalization can be calculated easily. When the message passing converges, the belief of a random variable can be calculated in Eq. 4. All the message passing procedures are termed belief propagation (BP).

$$belief(x_j) = \varphi^{\text{loc}}(x_j) \prod_{l=1}^{k} m_{\bar{l}\rightarrow j}(x_j) \tag{4}$$

E. Loopy Belief Propagation and Prior Updating

BP can calculate the exact posterior distribution of the random variables on a graph with no loops in linear time. The same algorithm can work if there are loops, but each message may have to be updated several times before the network converges. The procedure to apply the standard belief propagation rules to a graph with loops is called Loopy Belief Propagation (LBP), an approximate inference method to BP.

We can run loopy belief propagation on a factor graph that includes voting potentials. However, we might expect the message from a factor to a non-centered variable to be fairly weak: this suggests an even simpler algorithm for inference: we can run LBP but ignore all of the messages from factors to non-centered variables. (Ignoring a message means considering it to be uniform.) We will call this algorithm Prior Updating (PU) with Voting Potential. The name PU comes from the usage of the current classifications of a node's neighbors to update the prior for the node's own classification [13]. PU has shown good approximate inference performance in the cell image classification problem [11]. PU will be noticeably faster than LBP, since there will usually be many more non-centered variables than there are centered ones in each factor. We therefore used PU for the work described here.

$$
\begin{aligned}
& m_{\bar{j}\to k}(x_k) \\
={}& \sum_{\sim\{x_k\}} \left(1 + \sum_{i=1}^{k-1} \alpha I(x_i, x_k)(1-\varphi_{boundary}(i)) + \beta \bar{I}(x_i, x_k)\varphi_{boundary}(i)\right) \prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'}) \\
={}& \sum_{\sim\{x_k\}} \prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'}) + \sum_{\sim\{x_k\}} \sum_{i=1}^{k-1} \alpha I(x_i, x_k)(1-\varphi_{boundary}(i)) \prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'}) + \sum_{\sim\{x_k\}} \sum_{i=1}^{k-1} \beta \bar{I}(x_i, x_k)\varphi_{boundary}(i) \prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'}) \\
={}& 1 + \sum_{i=1}^{k-1} \sum_{\sim\{x_k\}} \alpha I(x_i, x_k)(1-\varphi_{boundary}(i)) \prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'}) + \sum_{i=1}^{k-1} \sum_{\sim\{x_k\}} \beta \bar{I}(x_i, x_k)\varphi_{boundary}(i) \prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'}) \\
={}& 1 + \sum_{i=1}^{k-1} \sum_{\sim\{x_k\}} \alpha I(x_i, x_k)(1-\varphi_{boundary}(i)) \cdot m_{i\to\bar{j}}(x_i) \prod_{i'=1, i'\neq i}^{k-1} m_{i'\to\bar{j}}(x_{i'}) \\
+{}& \sum_{i=1}^{k-1} \sum_{\sim\{x_k\}} \beta \bar{I}(x_i, x_k)\varphi_{boundary}(i) \cdot m_{i\to\bar{j}}(x_i) \prod_{i'=1, i'\neq i}^{k-1} m_{i'\to\bar{j}}(x_{i'}) \\
={}& 1 + \sum_{i=1}^{k-1} \sum_{x_i} \alpha I(x_i, x_k)(1-\varphi_{boundary}(i)) \cdot m_{i\to\bar{j}}(x_i) + \sum_{i=1}^{k-1} \sum_{x_i} \beta \bar{I}(x_i, x_k)\varphi_{boundary}(i) \cdot m_{i\to\bar{j}}(x_i) \\
={}& 1 + \sum_{i=1}^{k-1} \alpha(1-\varphi_{boundary}(i)) \cdot m_{i\to\bar{j}}(x_k) + \sum_{i=1}^{k-1} \beta\varphi_{boundary}(i) \cdot (1 - m_{i\to\bar{j}}(x_k)) \qquad (5)
\end{aligned}
$$

To calculate the voting potential, the messages from the variables to the factor nodes can be calculated easily using Eq. 2. Eq. 5 shows how we can calculate the messages from a factor node to the centered variable. In the derivation, we permute the index of the random variables in the potential function φ_j in Eq. 3 so that the index of the centered variable is k at the right hand side of the equation. We further assume that the message $m_{l\to\bar{j}}(x_l)$ is normalized so that $\sum_{x_l} m_{l\to\bar{j}}(x_l) = 1$. The first equation is the definition of the desired message. The second equation distributes multiplication over addition. The third equation uses the fact that all terms in the product $\prod_{i'=1}^{k-1} m_{i'\to\bar{j}}(x_{i'})$ are independent, along with our assumption $\sum_{x_{i'}} m_{i'\to\bar{j}}(x_{i'}) = 1$, to compute the summation. The fourth equation factors $m_{i\to\bar{j}}(x_i)$ out of the product. The fifth equation uses again the facts that all terms in the product are independent and $\sum_{x_{i'}} m_{i'\to\bar{j}}(x_{i'}) = 1$. The last line uses the fact that $I(x_i, x_k)$ is nonzero *iff* $x_i = x_k$. All the computation only involves summation and dot product between two vectors.

III. SEGMENTATION USING VOTING POTENTIAL

A. Dataset

To test our approach, we have used a collection of images of yeast cells [14]. It contains images of proteins encoded by more than four thousand open reading frames (ORFs). A three channel image was acquired for each ORF, consisting of DIC (differential-interference contrast), GFP (green fluorescence protein), and DAPI (a DNA dye) channels. Each image has around 20 to 50 cells, often clustered together. These ORFs have been categorized into 23 location patterns by visual examination. Automated segmentation of these images into single cell regions will faciliate automated analyzis of the location patterns. We selected a small subset of the images for our experiments.

B. DNA potential and boundary potential

We defined two potential functions, *DNA potential* and *boundary potential*, which can be derived from a DNA content image and an image with noisy information about the positions of cell boundaries. From the DNA intensity at each image pixel, we define a DNA potential, the local evidence that each pixel is likely to be in the foreground or background. The DNA potential is a column vector of dimension $n+1$, where n is the number of possible foreground labels (e.g., different cells). A pixel with a high DNA intensity is more likely to belong to the foreground, so its foreground potential should be set high (and background potential should be set low). We use a sigmoid function to transform each DNA intensity value to a value for the background potential, and set the potential for each foreground class to equal shares of the difference between the background potential and one. The threshold and slope of the sigmoid function are automatically chosen using two reference points in the image: a pixel with the 95^{th} percentile of the DNA intensity (which should have very low background potential), and a pixel on the initial cell outline (which should have about equal foreground and background potential). The second pixel was chosen as the pixel with the median DNA intensity out of all pixels on the initial cell outline. The potentials for the two references points were set to 0.11 and 0.273, respectively. The second value was chosen empirically as providing better results than using one-third (i.e., equal probabilities for background and two foreground classes).

Images such as total protein images, membrane protein

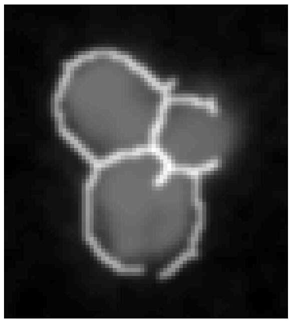

Fig. 4. An example of a boundary contour superimiposed on the DNA potential of the foreground label.

images, and DIC images usually have good information about cell boundaries. From these images, an initial cell contour can be extracted using basic image processing procedures, such as edge-finding or morphological operations. In our experiments, we first converted the DIC image into an edge image based on the assumption that where an edge is located usually has high intensity variance in the image. The variance of an image I at a certain position (x, y) is calculated as follows: $V(x, y) = \sum_i \sum_j g(i - x, i - y)[I(i, j) - \mu(i, j)]^2$, where g is a bivariate Gaussian distribution with standard deviation σ and $\mu(x, y) = \sum_i \sum_j g(i - x, i - y)I(i, j)$. We tried different values for σ and found that $\sigma = 2$ gave the best results. Because the local variance is dependent on the average local intensities in the images, we normalized the edge image as $s(x, y) = \frac{\sqrt{V(x,y)}}{\mu(x,y)}$, which is much less sensitive to the average local intensities. We further applied the Ridler-Calvard thresholding [17], morphological thinning, and low-pass filtering with a 7×7 gaussian window to the edge image. For each pixel x_i in the edge image, we define the neighborhood using a $W \times W$ square window, where x_i is at the center. For each pair of pixels x_i and x_j in the neighborhood, we calculated the sum of the intensity values of the pixels between x_i and x_j in the blurred images, then converted the sum to a contour potential by a sigmoid function (with empirically chosen slope of 1 and threshold of 5). The contour potential describes the probability of existence of a contour between two neighboring pixels. Pixels are likely to have different labels if there is a contour between them. An example of a contour image superimposed on the DNA potential of the foreground label is shown in Fig. 4.

C. Segmentation procedures

Each image in the dataset consists of 512×535 pixels. Each pixel is assigned a DNA potential $\varphi^{loc}(x_i)$ and a boundary image is generated by applying a basic edge finding algorithm to the DIC image. The DIC images were also used to identify sub-regions containing whole cells or separated cell clusters by finding connected regions in the blurred DIC image. For computational feasibility, a reasonable size for a sub-region is at most 100000 pixels; sub-regions of larger size were ignored.

In order to save memory and be able to operate on larger image regions, an iterative process is designed so that only one cell region will be identified at each iteration. To do this, we defined three classes, background, the foreground class of the current cell, and the foreground of all other cells. At the end of the first iteration, we simply separate all of the foreground pixels from the background pixels. The local evidence of the background pixels are set to be *[1, 0, 0]*, and the one pixel with the highest foreground confidence is selected. We set the local evidence of this pixel to be *[0, 1, 0]* and run belief propagation again until convergence. All of the foreground pixels with the highest belief in the second class are segmented as one cell. The segmented cell is stored and all of its pixels are considered as background by setting their local evidence to be *[1, 0, 0]*. We iterate the process of picking the most confident pixels in the foreground and segment out the cells one by one until the segmented region is too small to be considered as a cell.

The choice of W is also important because the memory requirement is quadratic to W. W should be large enough so that enough information from the neighbors can be incorporated, but it should be small enough to save computations. We evaluated different Ws and found that $W = 7$ gives the best accuracy and efficiency tradeoff.

IV. EXPERIMENTS

In this section, we compare experimental results for graphical model segmentation and seeded watershed segmentation. We randomly selected 13 images from the yeast image dataset, and manually segmented these images into 434 single cell regions using the DIC image as a guide. An example of a portion of a DIC image and a corresponding hand-labeled mask are shown Fig. 5 and Fig. 6, respectively.

A. Evaluation measurement

Assuming that the hand-labeled regions were correct, we calculated cell-level and pixel-level recall, precision, and F measures to compare the two segmentation methods. We define the area overlap for a hand-labeled mask (HM) as the the number of overlapping pixels between a machine mask (MM) and the HM divided by the total number of pixels in that HM. So that incomplete segmentation can be recognized, we permit only one MM to match with a given HM. Since there may be more than one MM overlapping with a HM, the one with largest area overlap is matched to that HM, and the other MMs are only permitted to match other HMs (some MMs may not end up matched). We further define an area overlap threshold T to distinguish between correctly and incorrectly segmented cells. A MM is considered as a true positive (cTP) if its area overlap with its corresponding HM is greater than T, and considered a false positvie (cFP) otherwise. (We chose a fairly stringent criterion of $T = 80\%$ in anticipation of the need for nearly complete cell regions for automated pattern classification.) The cell-level recall and precision can be calculated as $cTP/(number\ of\ HMs)$ and $cTP/(cTP + cFP)$, respectively. The cell-level F measure can be calculated as $2 \times Recall \times Precision/(Recall + Precision)$;

Based on the correspondence between MM and HM, we can similarly define the pixel-level true positve (pTP) and false positive (pFP). pTP is defined as the number of pixels in all MMs that overlap with the corresponding HM and pFP is defined as the number of pixels in all MM that overlap with all other HM. Note that by this definition pixels in a MM that do not overlap with any MM are ignored. This is donoe to avoid penalizing the watershed method for including additional surrounding background pixels in a mask. The pixel-level F measure is calculated as defined above. The cell-level measurement indicates the ability of the segmentation method to find correct masks, and the pixel-level measurement assesses the quality of the found masks.

B. Seeded Watershed Segmentation

The seeded watershed algorithm uses the nuclei as seeds and splits the reference image into regions equivalent to the drainage regions of the image landscape. Here we used the DNA image both for seeding and as the reference image. An example of a DNA image is shown in Fig. 8. The initial seeding plays a very crucial step for the final segmentation results. Our previously work [5] suggested that proper thresholding after background substraction could usually provide a good seeding. An example image by seeded watershed algorithm is shown in Fig. 9.

C. Comparision

We applied our graphical model segmentation on 13 yeast cell images with 434 hand-labeled single cell regions. An example resulting segmentation is shown in Fig. 10. We can see that there is a very high consisitency between the hand-labeled masks and the graphical model segmentation masks. We also applied the seeded watershed algorithm on the same images, using the DNA image as both seeding and reference image. Given the importance of the threshold to the seeding process, we used various factors of the automated threshold chosen by the Ridler-Calvard method [17]. The results with the highest cell-level F measure are reported. As shown in Table I, graphical model segmentation clearly outperformed seeded watershed segmentation. The pixel-level measurements indicated the quality of the generated masks, and the cell-level precision represent the ability of the algorithm to retrieve masks. We consider the latter to be the most important important metric to compare the two algorithms. We calculated the 95% confidence interval of the cell-level precisions: the average precision of the graphical model segmentation falls within (71.9%, 80.0%) and that of the seeded watershed algorithm falls within (58.2%, 67.3%). The non-overlapping confidence intervals confirmed that the graphical model segmentation outperformed the seeded watershed segmentation. (Because the seeded watershed segmentation assigns all pixels to a mask, the pixel-level precision, recall, and F measure are all the same.)

D. Post Processing

Although most of the masks generated by the graphical model approach are of good quality, some unsatisfactory masks need to be screened out. To do this, we can rank the resulting masks by some measurements and pick a threshold to select masks which are ranked higher. We extracted a set of features which are considered as good descriptions of good masks. The features are summarized in Table II. The first two features are related to the principal components of the mask. The eccentricity is calculated using the first two major components of the mask, and a good mask should have a low eccentricity value. The area fraction is defined as the ratio of the mask area and area of a fitted ellipse. The fitted ellipse can be found by first finding the two major principal components of the mask, and then set the length of the major axis to be of the largest Mahalanobis distance of all the pixels in the mask. The resulting ellipse would cover all the pixels within the mask. This area fraction is expected to be high for good masks. The next three features are related to the size and the DNA content of the cell. We also synthesized five additional features by taking the squares of each features. For each cell, 10 features were extracted and used for training the weights of logistic ridge regression classifier to identify good masks.

We used a leave-one-out cross validation approach: at each fold of cross-validation, one image is selected as the testing set and the remaining are used as the training set. All the machine generated masks in the training set were divided into positive and negative examples according to their area overlaps with hand-labeled masks and the area overlap threshold T. We also

TABLE I
EVALUATION OF TWO SEGMENTATION METHODS.

	Seeded Watershed	Graphical Model
Cell-Recall	62.9%	75.1%
Cell-Precision	62.8%	76.0%
Cell-F measure	62.8%	75.6%
Pixel-Recall	69.3%	82.9%
Pixel-Precision	69.3%	94.6%
Pixel-F measure	69.3%	88.4%

TABLE II
FEATURES USED FOR POST PROCESSING.

Feature Name	Description
eccentricity	The eccentricity of the mask
area fraction	The area ratio of the mask and fitted ellipse
total pixel	Total number of pixels
DNA per pixel	Average DNA intensity per pixel in the mask
total DNA	Total number of DNA intensity
synthesized	Square of each feature above

TABLE III
EVALUATION OF POST PROCESSING.

	Watershed	Graphical Model	Post Processing
Cell-Recall	61.4%	74.9%	55.3%
Cell-Precision	65.5%	75.9%	89.5%
Cell-F measure	62.6%	75.3%	65.5%
Pixel-Recall	67.4%	83.6%	60.4%
Pixel-Precision	67.4%	94.8%	96.7%
Pixel-F measure	67.4%	88.8%	72.0%

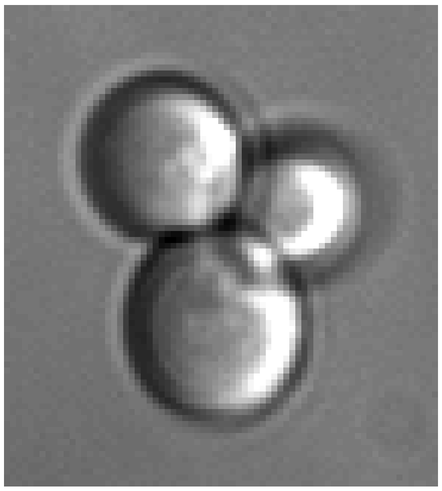

Fig. 5. An example of the DIC image. The DIC image usually gives good information about the cell boundary. The right upper cell is an out of focus cell.

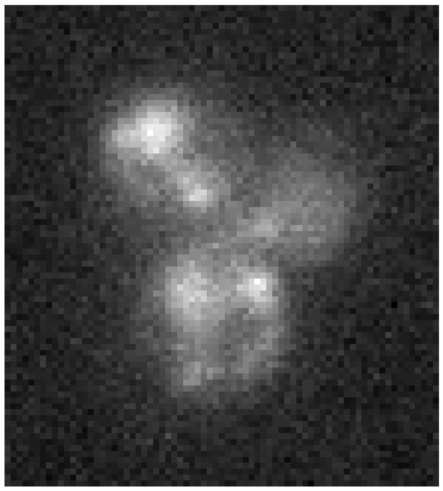

Fig. 8. An example of a region of a DNA image. The pixel belonging to the foreground usually has higher DNA intensity.

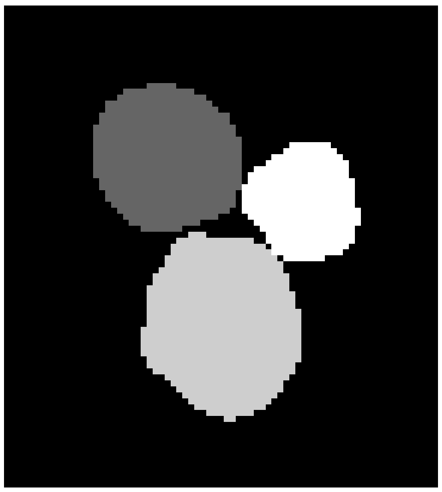

Fig. 6. An example of the hand segmentation mask.

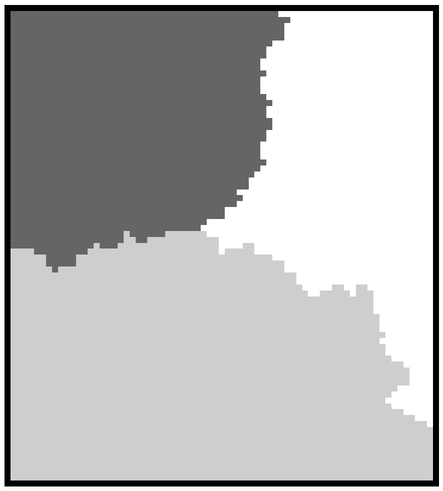

Fig. 9. An example of the masks resulting from seeded watershed segmentation.

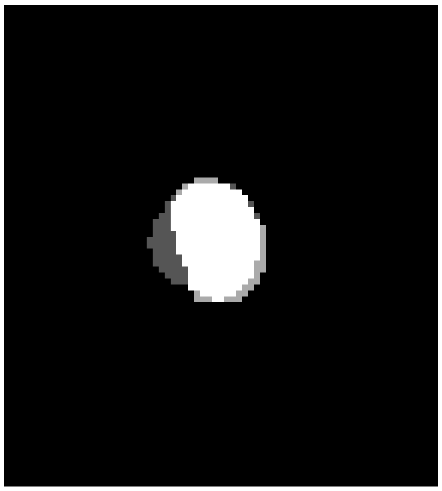

Fig. 7. An example of 80% cell overlap between the hand label data (deep gray) and the machine generated mask (light gray). The overlap pixels are in white color.

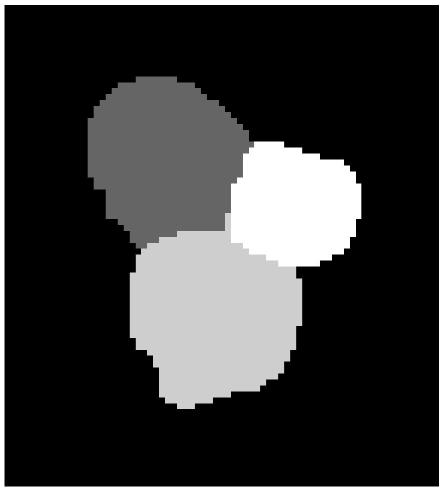

Fig. 10. An example of the masks resulting from automated segmentation using the prior updating algorithm with voting potential.

added the hand-labeled masks from the training set as positive examples to have a bigger training set. A weight vector is learned from the training set, and this vector is used to classify the masks in the testing set. A score threshold for testing set is learned from the value which gives the largest cell-level F measure in the training set. This score threshold is then used in the testing set. The masks with scores greater than the threshold are classified as positive images. Results for all methods with leave-one-out cross validation are shown in Table III. In these results, we also tuned the ridge parameter in the logistic ridge regression so that the cell-level Precision can be maximized. The ridge term corresponds to the prior of the coefficient in the logistic regression. The post processing results in a much larger cell-level precision but a lower cell-level recall. Since there are on average more than 25 cells of the same location pattern in the GFP channel, high quality masks are preferable for automated pattern analysis even if some cells are lost. After post processing, the cell-level precision is increased from 75.9% to 89.5%. We performed the one-sided paired t-test on the cell-level precision between the masks with and without post processing, and the p-value (0.0015) suggests that post processing can statistically significantly improve the cell-level precision. The p-value between masks of post processing and watershed is even smaller (0.0002).

V. Conclusion

Graphical model segmentation is a fast and accurate segmentation method, and it is especially useful for segmenting a field containing cells that are touching each other, as is often the case with yeast images. It also has the advantage of not requiring the explicit seeds which the seeded watershed algorithm and level set based methods do. When the seeding is proper, seeded watershed algorithm can usually achieve robust segmentation results. However, the segmentation results are very sensitive to the thresholds and different regions may need different thresholds to achieve good results. A high threshold will result in over-segmentation and a low threshold will result in under-segmentation. If there is a good source of seeding available, such as for 3D stack of images, we can find the initial seeding by overlapping the DNA images over stacks. In other cases, when the initial seeding is tricky, graphical model segmentation is an good alternative.

There is still room for improvement on the quality of the masks, both in the parameter learning in the loopy belief propagation and the features and procedures in the post processing. Another important issue is the detection of out-of-focus cells. As we examined our incorrect masks carefully, a lot of false positives came from out-of-focus cells. Our results could be improved further by removing these cells.

It is worth noting that the computational requirements for these methods are quite different. While the time needed for seeded watershed is within a second, a MATLAB implementation of our graphical model segmentation and post processing took around 6 minutes per image. While this is much slower, it is small compared to the time required to tag and image each yeast protein and faster than some other methods (such as some implementations of active contours). Segmentation is such a hard problem that there is no universal method for segmenting all kinds of images. However, we expect our algorithm to perform well on a large range of fluorescent microscopy images. We hope that the graphical model segmentation can be useful as the first step in an automated analysis of protein subcellular location patterns in multi-cell images, especially for high throughput fluorescent microscopy images.

Acknowledgment

We thank Adam Carroll and Erin O'Shea for providing the yeast images. This work was supported in part by NSF grant EF-0331657.

References

[1] X. Chen, M. Velliste, and R. F. Murphy, "Automated interpretation of subcellular patterns in fluorescence microscope images for location proteomics," *Cytometry A.*, vol. 69A, no. 7, pp. 631–640, 2006.

[2] E. Bengtsson, C. Wählby, and J. Lindblad, "Robust cell image segmentation methods," *Pattern Recognition and Image Analysis*, vol. 14, no. 2, pp. 157–167, 2004.

[3] K. Rodenacker and P. Bischoff, "Quantification of tissue sections: Graph theory and topology as modelling tools," *Pattern Recogn Lett*, vol. 11, pp. 275–284, 1990.

[4] R. Lotufo and A. Falcao, *The ordered queue and the optimality of the watershed approaches*. In: Mathematical Morphology and its Application to Image and Signal Processing, Kluwer Academic Publishers, 2000, pp. 341–350.

[5] M. Velliste and R. F. Murphy, "Automated determination of protein subcellular locations from 3D fluorescence microscope images," in *Proc. Intl. Symp. on Biomedical Imaging (ISBI)*, 2002, pp. 867–870.

[6] L. Coulot, H. Kirschner, A. Chebira, J. M. F. Moura, J. Kovacevic, E. G. Osuna, and R. F. Murphy, "Topology preserving STACS segmentation of protein subcellular location images," in *Proc. Intl. Symp. on Biomedical Imaging (ISBI)*, 2006, pp. 566–569.

[7] C. O. de Solorzano, R. Malladi, S. A. Lelievre, and S. J. Lockett, "Segmentation of nuclei and cells using membrane related protein markers," *Journal of Microscopy*, vol. 201, no. 3, pp. 404–415, 2001.

[8] B. Taskar, A. Pieter, and D. Koller, "Discriminative probabilistic models for relational data," in *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*. Morgan Kaufmann, 2002, pp. 485–49.

[9] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 261–268.

[10] R. Potts, "Some generalized order-disorder transformations," in *Proc. Cambridge Philosophical Soc.*, vol. 48, 1952, pp. 106–109.

[11] S.-C. Chen, G. J. Gordon, and R. F. Murphy, "A novel approximate inference approach to automated classification of protein subcellular location patterns in multi-cell images," in *Proc. Intl. Symp. on Biomedical Imaging (ISBI)*, 2006, pp. 558–561.

[12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[13] S.-C. Chen and R. F. Murphy, "A graphical model approach to automated classification of protein subcellular location patterns in multi-cell images," *BMC Bioinformatics*, vol. 7, pp. 90–102, 2006.

[14] W. K. Huh, J. V. Falvo, L. C. Gerke, A. S. Carroll, R. W. Howson, J. S. Weissman, and E. K. O'Shea, "Global analysis of protein localization in budding yeast," *Nature*, vol. 425, no. 6959, pp. 686–691, 2003.

[15] B. A. Cipra, "An introduction to the ising model," vol. 94, pp. 937–959, 1987.

[16] N. Cressie and J. L. Davidson, "Image analysis with partially ordered markov models," *Computational Statistics and Data Analysis*, vol. 29, no. 1, pp. 1–26, 1998.

[17] T. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *IEEE transactions on Systems, Man and Cybernetics*, vol. 8, pp. 630–632, 1978.