

Cell Segmentation via Shape Modeling

Kelvin Gu^{1,2}, Michael Mayhew², and Alexander Hartemink²

¹ Department of Statistical Science, Duke University,

² Department of Computer Science, Duke University

kg44@duke.edu, michael.mayhew@duke.edu, amink@cs.duke.edu

Abstract. We develop a novel generative model for arbitrary 2D shapes, based on multiscale deformation. We demonstrate how our model can be used to segment objects within an image by fitting a shape-sensitive contour to each object’s boundary. We demonstrate our model by performing segmentation on differential interference contrast microscopy (DIC) images of yeast, where many cells possess aberrant shape and boundaries that contain major gaps or touch other cells. Our method can be seen as a ‘shape-sensitive’ and statistical active contour.

Keywords: active contours, generative modeling, shape modeling, morphometry, Bayesian statistics, Bezier curves, cell segmentation

1 Introduction

When analyzing an image of cells, it is often useful to identify the region belonging to each cell. This is known as cell segmentation, which enables us to observe single-cell features, such as a cell’s size, or the number of peroxisomes it contains, etc.

The difficulty of segmentation can range from trivial to arbitrarily hard, depending on the quality of the image’s region or boundary information, which we characterize below.

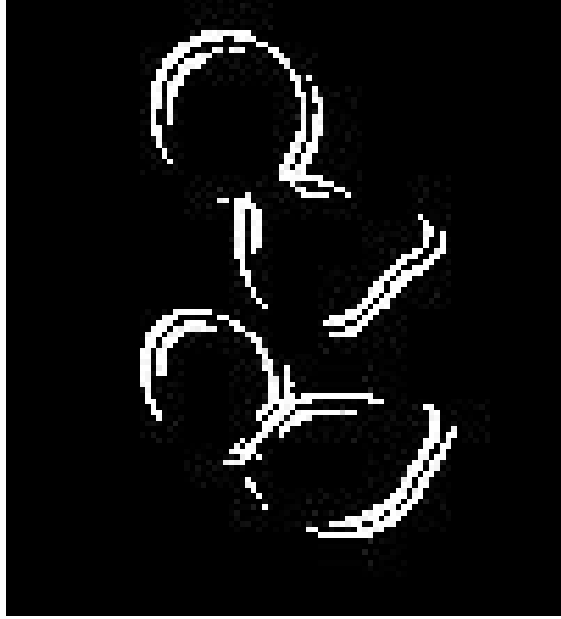
If a cell’s interior texture or brightness differs cleanly from the background, a simple threshold will isolate it. Such images possess good region information. However, numerous popular imaging modalities lack good region information, such as differential interference contrast microscopy (DIC).

If an image lacks good region information, one can often still detect cell boundaries using an edge detection algorithm, such as the Canny edge detector []. If the detected edge pixels completely enclose the cell region with no gaps, then algorithms such as watershed [], which flood-fill the enclosed space, can recover the enclosed interior. Such images contain good boundary information. Again, certain imaging modalities simply do not reveal complete, closed cell boundaries.

If an image possesses poor boundary information, a third source of information is often exploited to achieve segmentation: shape. For example, if the target cells share a known rigid shape (e.g. circular), template-matching or variants of the Hough transform [] can be used to detect them. However, such rigid-shape methods are not designed to handle a cell population with significant shape variability. Even with significant variability, shape is intuitively still an important source of information. After automated segmentation, a cytologist can often spot errors on the basis of just shape and size. Yet, it has proven difficult to leverage this intuition during the segmentation process. At one extreme, watershed-based methods rarely use shape information at all (or employ highly-involved machinery to include it []). At the other extreme, template and Hough-transform based methods stick rigidly to one exact shape.

Perhaps the best methods to date which flexibly address shape are active-contours, a family of contour-fitting algorithms which iteratively move towards boundaries in the image according to an energy minimization principle. They can be initialized to arbitrary shapes, and are also controlled by a balloon force, a thin-plate energy, and several other forces.

However, active-contours still seem far from exploiting the full information captured in shape. First, parameters such as balloon force control how a contour should evolve, not how it should look, and therefore do not map to any notion of shape. For example, if a contour is initialized inside a closed boundary, balloon force will encourage it to expand and fill the cavity as desired. However, if the boundary contains a gap, the same balloon force will cause the contour to spill through the gap. This implies that even if a population of cells all share similar shape, no single parameter setting may be satisfactory for all cells. Second, active



contours do not quantify uncertainty in their fit. This is harmful if a bad fit is used for downstream tasks. Finally, active contours fit each object independently, and therefore cannot share statistical strength across multiple cells if they come from the same shape distribution.

In this paper, we propose to explicitly model shape variability and use our model to perform segmentation. Our novel statistical method easily fits arbitrary 2D, simply-connected shapes while also giving an intuitive multiscale representation for the shape of each segmented cell.

We demonstrate our method on a challenging set of aberrant yeast images. Like active contours, our method does not require good region or boundary information, as necessary for thresholding or watershed-based methods. Furthermore, we show how our method can simultaneously fit multiple cells even when they are tightly packed, a scenario which traditionally confuses active-contours. Finally, we note that our general model for shape can also be used for shape registration and classification, which are explored further in [].

2 Methods

2.1 Overview

Our shape model is a Bayesian hierarchical model. First, we introduce the representation for a closed curve, which we use to model a cell’s boundary. Second, we describe a Bayesian formulation for curve-fitting. Third, we introduce the likelihood function and prior distribution used in our formulation. The prior is a probability distribution over shapes and is used to guide curve fitting, such that fitting favors curves of a certain shape, size or even smoothness. Fourth, we make our shape prior adaptive, so that rather than favoring a specific shape, it learns population-level shape similarity, and uses this information to guide curve-fitting. Finally we introduce a set of latent variables that become necessary when modeling cells that are closely packed, with poorly separated boundaries. After presenting the full model, we then describe inference via Gibbs sampling.

We emphasize that our method performs curve-fitting and shape analysis jointly, which provides three crucial advantages: 1) If a cell boundary contains large gaps, shape information is critical for bridging the gap. 2) If cells are clumped together, their boundaries meet and can be easily confused. Shape information helps the curve decide which boundary it should be fitting. 3) If we have uncertainty fitting a cell boundary, that uncertainty needs to be acknowledged if we plan to do shape analysis on the extracted curves.

2.2 Curve Representation

We model a cell’s boundary as a special kind of parametric curve, $R(t)$ []. The curve is defined by a set of $2n + 1$ control points $\{c_j\}_{j=0}^{2n}$, where n is the degree of the curve and we may choose it to be any positive

integer, depending on how many control points we want. (For convenience, we will define the total number of control points as r , where $r = 2n + 1$.) The curve is a function of t , and can be viewed as the trajectory of a particle over time. At every time t , the particle's location is defined as some convex combination of all control points. The weight accorded to each control point in this convex combination varies with time according to a set of basis functions, $\{B_j(t)\}_{j=0}^{2n}$, with one basis function setting the weight for each control point. This representation is very similar to Bezier curves, used in graphic design [].

$$R(t) = \sum_{j=0}^{2n} c_j B_j(t), \quad t \in [-\pi, \pi] , \quad (1)$$

$$B_j(t) = \frac{K_n}{2^n} \left\{ 1 + \cos \left(t + \frac{2\pi j}{2n+1} \right) \right\}^n, \quad K_n = \frac{(2^n n!)^2}{(2n+1)!} , \quad (2)$$

where $c_j = [c_{j,x} \ c_{j,y}]^T$ specifies the location of the j th control point and $B_j : [-\pi, \pi] \rightarrow [0, 1]$ is the j th basis function. Our curve also has the following important properties:

1. The curve is always closed. Therefore, if it does not self-intersect, it will represent the boundary of a simply-connected shape.
2. When $n = 1$, the curve is defined by $r = 3$ control points, which always specify an ellipse. n can be increased to capture an arbitrary amount of curve complexity.
3. The entire curve is smooth, because it is infinitely differentiable.
4. We emphasize that our curve has a finite representation, fully expressed by the control points.

For notational convenience, let us store $\{c_j\}_{j=0}^{2n}$ in a single vector (for the rest of the paper, we express sets of control points in this 'stack format'):

$$c = [c_{0,x} \ c_{0,y} \ c_{1,x} \ c_{1,y} \ \dots \ c_{2n,x} \ c_{2n,y}]^T \quad (3)$$

Then, (1) can be written in vector notation:

$$R(t) = B(t)c \quad (4)$$

$$B(t) = \begin{bmatrix} B_0(t) & 0 & \dots & B_{2n}(t) & 0 \\ 0 & B_0(t) & \dots & 0 & B_{2n}(t) \end{bmatrix} \quad (5)$$

We have now stacked all the control points in the vector c , and the values of the basis functions at a particular time t , in $B(t)$.

2.3 Bayesian Formulation

The data for our model is a set of N 2-dimensional points, $\{p_i\}_{i=1}^N$, that are scattered close to the true cell boundary. In practice, these are the pixels identified as 'edges' by an edge-detection algorithm. Since our parametric curve can be thought of as a function expressing the trajectory of a particle over time, we view each data point, p_i , as a noisy estimate of the particle's location at a given time t_i ,

$$p_i = R(t_i) + \epsilon_i = B(t_i)c + \epsilon_i , \quad (6)$$

where we assume that ϵ_i is Gaussian noise,

$$\epsilon_i \sim N_2(0, \sigma_p \mathbb{I}) . \quad (7)$$

$N_2(\cdot)$ is the bivariate normal distribution. \mathbb{I} is the identity matrix, which, together with σ_p , specifies a radially symmetric normal distribution.

For notational convenience, let us store the sets $\{p_i\}_1^N$, $\{t_i\}_1^N$ and $\{\epsilon_i\}_1^N$ in the same stack format as c :

$$p = [p_{1,x} \ p_{1,y} \ p_{2,x} \ p_{2,y} \ \dots \ p_{N,x} \ p_{N,y}]^T , \quad (8)$$

$$\epsilon = [\epsilon_{1,x} \ \epsilon_{1,y} \ \epsilon_{2,x} \ \epsilon_{2,y} \ \dots \ \epsilon_{N,x} \ \epsilon_{N,y}]^T , \quad (9)$$

$$t = [t_{1,x} \ t_{1,y} \ t_{2,x} \ t_{2,y} \ \dots \ t_{N,x} \ t_{N,y}]^T . \quad (10)$$

Let us also define:

$$A(t) = \begin{bmatrix} B(t_1) \\ B(t_2) \\ \dots \\ B(t_N) \end{bmatrix} \quad (11)$$

Then, we can rewrite (6) for all i , as:

$$p = A(t)c + \epsilon \quad (12)$$

$$\epsilon \sim N_{2N}(0, \sigma_p \mathbb{I}) \quad (13)$$

To perform curve-fitting, we want to infer the set of control points which define a trajectory that reflects the data. Following a Bayesian framework, this translates to asking for $\mathbb{P}(c|p)$, the posterior distribution over control points, given the data. To compute this distribution, we invoke Bayes Rule,

$$\mathbb{P}(c|p) = \frac{\mathbb{P}(p|c)\mathbb{P}(c)}{\mathbb{P}(p)} \quad (14)$$

To compute the right hand side, we must determine $\mathbb{P}(p|c)$ (the likelihood function) and $\mathbb{P}(c)$, which is the prior distribution on c . The likelihood function captures the ‘error’ between a curve specified by c and the data, p . This can be viewed as the objective function to be maximized for curve-fitting. The prior distribution can be viewed as an additional ‘regularization term’ added to the objective. In Section , we explain how to sample from the posterior distribution via Gibbs sampling, in which case it is unnecessary to know $\mathbb{P}(p)$, a normalizing constant.

2.4 Likelihood and Prior

Likelihood function. From (10) and (11), we can now specify the likelihood function as,

$$\mathbb{P}(p|c) = N_{2N}(p; A(t)c, \sigma_p \mathbb{I}) \quad (15)$$

Shape prior. We now specify and motivate the prior, $\mathbb{P}(c)$. When we do not have good data regarding a cell’s boundary, we would like to guide curve-fitting with soft biases about the smoothness, size or even shape of the curve. We accomplish this by specifying a prior distribution on curve shape. A curve’s shape is fully determined by its control points, so this amounts to specifying a distribution over the set of control points. We propose a completely novel distribution, $\mathbb{P}(c)$, which expresses shape. This distribution quantifies the probability of control points generated by a special random process that we now describe (and illustrate in Figure ??).

Random process. First, we motivate the random process in terms of cell morphology:

Note that many cells tend towards an elliptical form. Thus, sometimes a cell’s shape can be constructed as a coarse ellipse with finer details layered on top. If we wish to include surface texture, we can even add another layer of very fine details on top of the previous layer. Our random process generates shapes in exactly this multiscale, additive manner.

In the first step, we generate a degree $n_1 = 1$ curve (an ellipse) by randomly drawing $r_1 = 3$ control points from a bivariate Gaussian distribution centered at $(0, 0)$. We call these level-1 control points, indicated by a superscript 1.

$$c_j^{(1)} \sim N_2(0, \sigma_1 \mathbb{I}) \quad , \text{ for } j \in \{0, 1, 2\} \quad (16)$$

Again, we write $\{c_j^{(1)}\}_0^2$ in stack format:

$$c^{(1)} \sim N_6(0, \sigma_1 \mathbb{I}) \quad (17)$$

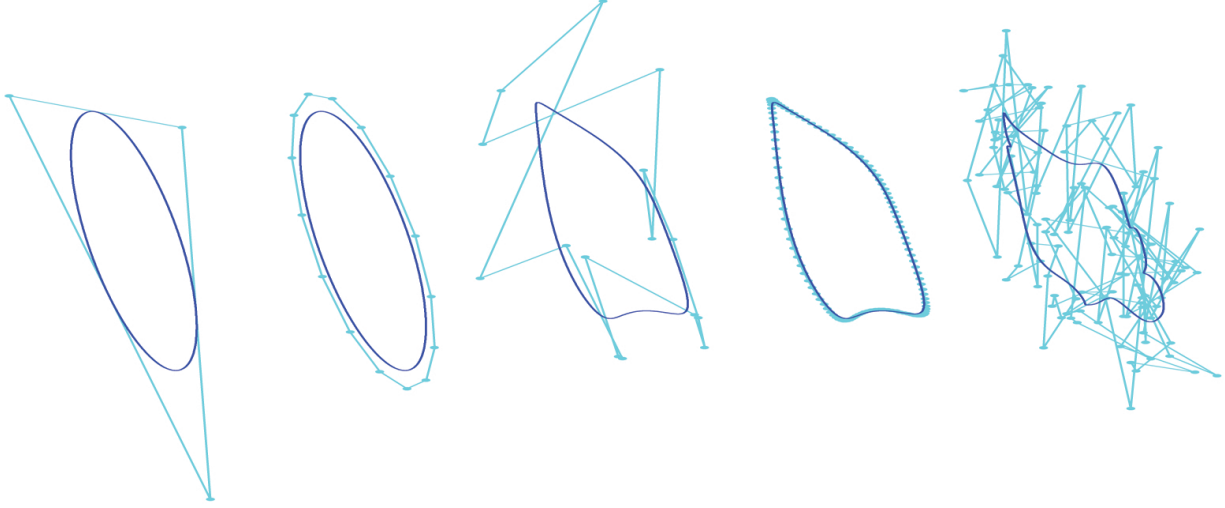


Fig. 1. An illustration of the shape generation process. **From left to right:** initial ellipse specified by three control points, ellipse after degree elevation, addition of fine details, degree elevation again, and addition of even finer details. Dark lines indicate the curve, pale dots indicate the control points, and pale lines connect the control points in order. Note that after each degree elevation, the control points appear to be much more ‘orderly.’ This is because each point now controls a smaller region of the curve.

Note that σ_1 controls the spread of the initial three control points, which roughly determines the size of the cell. We then wish to add finer details to this coarse elliptical curve. However, three level-1 control points cannot express anything other than an ellipse. Therefore after the ellipse is generated, we re-express the same ellipse using some larger number of control points, r_2 (with corresponding degree n_2). This step is called degree elevation and is a completely deterministic operation.

Definition 1. *If we start with a curve of degree n , defined by control points $\{c_i\}_0^{2n}$, and elevate its degree by the positive integer v , then the new degree elevated curve is defined by a new set of control points $\{\hat{c}_j\}_0^{2(n+v)}$, where each degree-elevated control point \hat{c}_j is defined as:*

$$\hat{c}_j := \frac{1}{2n+1} \sum_{i=0}^{2n} c_i + \frac{\binom{2(n+v)}{n+v} K_n}{2^{2n-1}} \sum_{k=0}^{n-1} \frac{\binom{2n}{k}}{\binom{2(n+v)}{v+k}} \sum_{i=0}^{2n} \cos \left((n-k) \left(\frac{-2j\pi}{2(n+v)+1} \right) + \frac{2(n-k)i\pi}{2n+1} \right) c_i . \quad (18)$$

The only important point about degree elevation is that there is a linear relationship between the original control points and the degree-elevated points. Therefore, we express degree elevation from degree a to degree b using the linear operator $D^{(a,b)}$:

$$\widehat{c}^{(1)} = D^{(n_1, n_2)} c^{(1)} . \quad (19)$$

Now that we have expressed the same ellipse with more control points, we add a random effect to each new control point to obtain a new curve with finer details. The new ‘level-2’ control points are stored in the ‘stacked’ vector $c^{(2)}$. Note that the level-2 curve has degree n_2 , not 2.

$$c^{(2)} = \widehat{c}^{(1)} + d^{(2)} \quad (20)$$

$$d^{(2)} \sim N_{2r_2}(0, \sigma_2 \mathbb{I}) \quad (21)$$

$d^{(2)} \in \mathbb{R}^{2r_2}$ is intuitively the ‘details’ that have been added to the ellipse. σ_2 influences the size of each $d_j^{(2)}$, and therefore determines ‘smoothness.’ Finally, we degree-elevate again to an even higher degree, n_3 ,

and add another layer of details, $d^{(3)}$, to produce level-3 control points, $c^{(3)}$, which can express fine textural details:

$$\widehat{c^{(2)}} = D^{(r_2, r_3)} c^{(2)} , \quad (22)$$

$$c^{(3)} = \widehat{c^{(2)}} + d^{(3)} , \quad (23)$$

$$d^{(3)} \sim N_{2r_3}(0, \sigma_3 \mathbb{I}) . \quad (24)$$

σ_3 influences the amount of deviation from the level-2 shape, and therefore determines the ‘texture bumpiness.’

If desired, we could recurse further and produce level-4 control points. However, for the dataset that we are tackling, this level of complexity is unnecessary. The end result of the process is a set of level-3 control points. These are the control points characterized by our prior distribution. In other words, in the expression $\mathbb{P}(c)$, $c := c^{(3)}$.

Summary of prior. The parameters of our shape distribution are $n_1, n_2, n_3, \sigma_1, \sigma_2$ and σ_3 . Each n controls the number of control points at each level, σ_1 influences object size, σ_2 influences smoothness and σ_3 controls texture.

We do not have a closed form expression for $\mathbb{P}(c)$. However, using the random process described above, we develop a sampling technique which still allows us to use $\mathbb{P}(c)$ and sample from $\mathbb{P}(c|p)$, the target posterior. We explain this in Section .

We also explore how to make the prior favor a very specific shape, invariant to rotation and minor rescaling/shear, in []. Lastly, we note that it is possible to introduce dependence between control points by specifying a more complex covariance structure, and we plan to consider this in future work.

Fig. 2. : illustrate curve-fitting

Caption: Wherever a cell possesses strong boundary information, the fine-scale curve will be pulled strongly towards the data. Wherever boundary information is missing, it will fall back towards the coarser ellipse curve.

2.5 Adaptive Shape Prior

Suppose we do not know exactly what sort of shape, size or smoothness the prior should favor, but we want it to express the knowledge that a population of cells should be similar.

So, rather than treating the parameters of the prior distribution as fixed values that we specify, let us now assume σ_1, σ_2 and σ_3 are unknown values that should reflect trends in the population. Since the parameters of the shape distribution are now random variables, we must put a prior on them to enable computation of their posterior distribution. We use weakly informative Gamma priors.

2.6 Summary of Model

We now summarize the model in terms of multiple cells.

$$\sigma_1 \sim Ga(\alpha_1, \beta_1) , \sigma_2 \sim Ga(\alpha_2, \beta_2) , \sigma_3 \sim Ga(\alpha_3, \beta_3) \quad (25)$$

$$\text{For each cell } k \in \{1, 2, \dots, K\}, \quad (26)$$

$$c^{(1),k} \sim N_{2r_1}(0, \sigma_1 \mathbb{I}) \quad (27)$$

$$c^{(2),k} \sim N_{2r_2}(D^{(n_1, n_2)} c^{(1),k}, \sigma_2 \mathbb{I}) \quad (28)$$

$$c^{(3),k} \sim N_{2r_3}(D^{(n_2, n_3)} c^{(2),k}, \sigma_3 \mathbb{I}) \quad (29)$$

$$p^k \sim N_{2N}(A(t^k) c^{(3),k}, \sigma_p \mathbb{I}) , \quad (30)$$

where $c^{(l),k}$ is the stacked vector of level- l control points for cell k , p^k is the stacked vector of data points for cell k , and t^k is the stacked vector of times corresponding to each point in p^k .

It is also possible to use our shape model to learn a more specific shape from the population, and again we refer to [].

2.7 Modeling Closely Packed Cells

There are several complications that arise when analyzing multiple cells that are ‘clumped together’. In the yeast image set that we tackle, multiple cell boundaries are often touching and it is unclear where one boundary ends and another begins. Thus, for each data point p_i , we do not know which cell/curve k owns it. We call this the ‘ownership problem.’ To solve it, we treat the ownership of each point p_i as a latent variable o_i , with an additional category for ‘no owner’. Furthermore, for each data point, we do not actually know its corresponding t_i . Until now, we have assumed that this was known. This problem is typically called the ‘parametrization problem’. Again, we treat each t_i as a latent variable. Finally, cells in the image are not all centered at $(0, 0)$, as we assumed earlier. Therefore, for each cell, we include a latent variable which centers it: $\mu^k \in \mathbb{R}^2$.

Fig. 3. : clumped cells

With these additional latent variables, we add the following details to our model:

$$o_i \in \{1, 2, \dots, K + 1\} \text{ for each } i \text{ (where } K+1 \text{ is the ‘no owner’ category)} \quad (31)$$

$$\{p_i\}_{i=1}^N = \text{the set of all boundary points appearing in the image} \quad (32)$$

$$\{t_i\}_{i=1}^N = \text{the set of all times corresponding to each } p_i \quad (33)$$

$$p^k = \text{the stacked vector of the set } \{p_i | o_i = k, i \in \{1 \dots N\}\} \quad (34)$$

$$t^k = \text{the stacked vector of the set } \{t_i | o_i = k, i \in \{1 \dots N\}\} \quad (35)$$

$$\mu^k \sim \text{Unif}[\text{domain of the image}] \quad (36)$$

$$o_i \sim \sum_{\kappa=1}^K \left(\frac{1}{K} \right) \mathbf{1}_{\kappa}(k) \quad (37)$$

$$t_i \sim \Phi_{c^{(3),o_i}}^{-1}(\text{Unif}[0, L_{o_i}]) \quad (38)$$

where $L_{o_i} = \Phi_{c^{(3),o_i}}(\pi)$ is the length of curve o_i . ($\Phi_{c^{(3),o_i}} : [-\pi, \pi] \rightarrow \mathbb{R}$ is the function mapping time to distance-traveled, along the curve specified by $c^{(3),o_i}$, which is the set of level-3 control points for the cell specified by o_i .) We also make one revision to (27):

$$c^{(1),k} \sim N_{2r_1}(m^k, \sigma_1 \mathbb{I}) \quad , m^k = \begin{bmatrix} \mu^k \\ \mu^k \\ \mu^k \end{bmatrix} \quad (39)$$

The prior distribution on each o_i , in (), is simply uniform over all cells. The prior distribution on each t_i , in (), is simply uniform over the length of the curve selected by o_i . The prior distribution on each μ^k is uniform over the image. All three priors are uninformative. We elaborate on this choice in [].

2.8 Model Fitting (Posterior Computation)

To perform curve-fitting, we seek the value of $c^{(3),k}$ for each cell k . We sample from the posterior distribution for for each $c^{(3),k}$ Gibbs sampling. Sampling is efficient because most of the conditional posteriors in our Gibbs sampler have a closed form (i.e. the conditional priors are conjugate).

If cells are not ‘clumped together’, then we do not need to make o_i latent, and sampling is extremely efficient. The Gibbs sampler reaches steady state in 1 or 2 steps, and each step is nearly instantaneous even

when fitting 50-100 cells. Also, the posterior distribution is very tight and 2 or 3 more samples are sufficient to obtain a good point estimate.

If we must introduce latent o_i^N , our Gibbs sampler appears to reach its steady state within roughly 60 steps. Nearly all steps are spent assigning pixels to the correct owner. Currently, the slowest stage in each step is checking the line-of-sight condition for each pixel. We elaborate on posterior computation in [] but state the necessary conditional updates here.

First, the conditional updates for $c^{(1)}, c^{(2)}$ and $c^{(3)}$ all have the same form. For a given level l :

$$\mathbb{P}(c^{(l)}|-) = N_{2r_l}(\hat{\mu}, \hat{\Sigma}) \quad (40)$$

$$\hat{\Sigma}^{-1} = \Sigma_b^{-1} + T_a^T \Sigma_a^{-1} T_a \quad (41)$$

$$\hat{\mu} = \hat{\Sigma}(T_a^T \Sigma_a^{-1} c_a + \Sigma_b^{-1} T_b c_b) \quad (42)$$

$$(43)$$

We now specify $T_a, T_b, \Sigma_a, \Sigma_b, c_a$ and c_b for each level l .

Table 1. Values

level	T_b	T_a	Σ_b	Σ_a	c_b	c_a
1	\mathbb{I}_{2r_1}	$D^{(n_1, n_2)}$	$\sigma_1 \mathbb{I}$	$\sigma_2 \mathbb{I}$	m	$c^{(2)}$
2	$D^{(n_1, n_2)}$	$D^{(n_2, n_3)}$	$\sigma_2 \mathbb{I}$	$\sigma_3 \mathbb{I}$	$c^{(2)}$	$c^{(3)}$
3	$D^{(n_2, n_3)}$	$A(t)$	$\sigma_3 \mathbb{I}$	$\sigma_p \mathbb{I}$	$c^{(3)}$	p

We discretize the possible values of t_i to obtain a discrete approximation of its conditional posterior:

$$\mathbb{P}(t_i|-) = \frac{N(p_i; A(t_i)c^{(3), o_i}, \sigma_p \mathbb{I})}{\sum_{\tau} N(p_i; A(\tau)c^{(3), o_i}, \sigma_p \mathbb{I})} \quad (44)$$

We can make this arbitrarily accurate, by making a finer summation over τ .

We employ a pseudo-conditional update for each o_i . We explain this in [].

$$\mathbb{P}(o_i|-) = \frac{N(p_i; A(t_i)c^{(3), o_i}, \sigma_p \mathbb{I}) \mathbf{1}_{\text{LOS}}(p_i, c^{(3), o_i})}{\sum_{\gamma=1}^K N(p_i; A(\tau)c^{(3), \gamma}, \sigma_p \mathbb{I})} \quad (45)$$

$\mathbf{1}_{\text{LOS}}(\cdot, \cdot)$ is an indicator function which equals 1 if p_i has line-of-sight (LOS) to the curve o_i .

3 Results

3.1 Comparison Against Active Contours and Watershed

3.2 Performance Tests

If the nucleus of each cell is known, our segmentation technique is fully automated. We initialize a curve at each nucleus and allow it to ‘grow’ into the surrounding space. If the nuclei are not known, then the user may select a single point within each cell to initialize a curve.

We present our method on examples that are increasingly difficult. We use the same parameters on each example, with $n_1 = 1, n_2 = 15$ and $n_3 = 20$. We initialize one curve at the centroid of each nucleus. The curve is initialized to a circle of radius 4. The truncation distance for ownership was chosen to be 15 pixels. In the first sample, to ensure that each curve claims some pixels, we temporarily boost the truncation distance to 35.

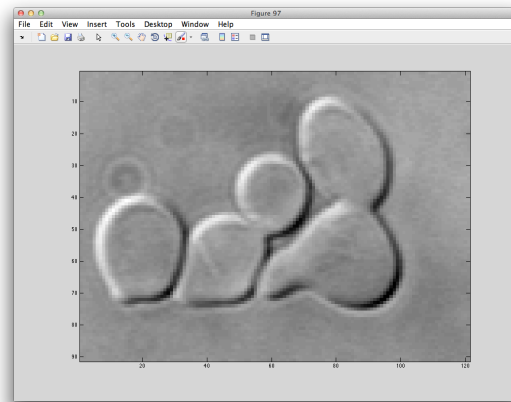


Fig. 4. clumped cells with boundary gaps that open into other cells

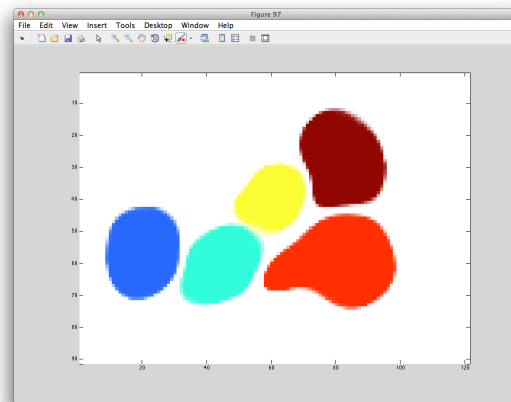


Fig. 5. Result

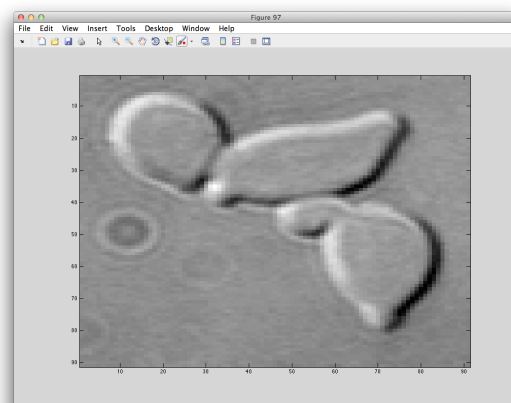


Fig. 6. Deformed buds with boundaries that open into 2 neighboring boundaries

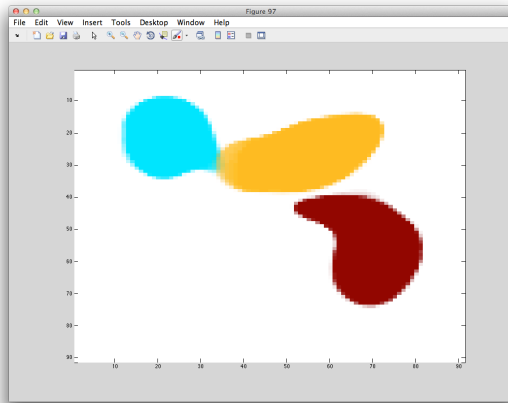


Fig. 7. Result

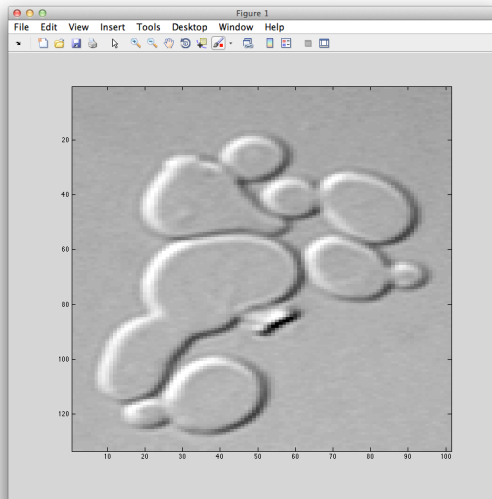


Fig. 8. Highly aberrant shapes with gaps

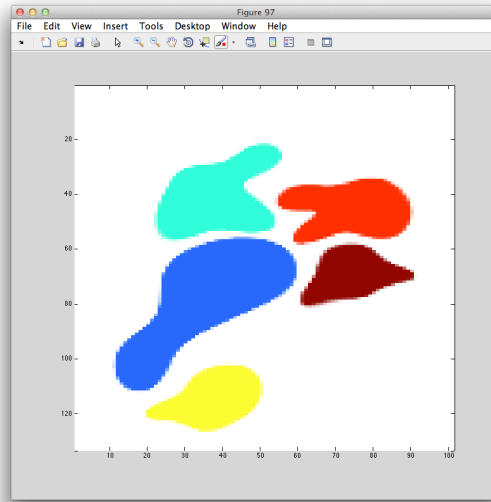


Fig. 9. Result

4 Discussion

References

1. Leaf Boundary Extraction and Geometric Modeling of Vegetable Seedlings.
2. Desideri, J., Janka, A.: Multilevel Shape Parameterization for Aerodynamic Optimization Application to Drag and Noise Reduction of Transonic/Supersonic Business Jet European Congress on Computational Methods in Applied Science and Engineering (2004)
3. Xu, C., Prince, J.: Snakes, Shapes, and Gradient Vector Flow IEEE Transactions on Image Processing, Vol 7, No 3 (1998)