IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 13-18, 2010

Finding Dots: Segmentation as Popping out Regions from Boundaries

Elena Bernardis
University of Pennsylvania
Philadelphia, PA 19104

Stella X. Yu
Boston College
Chestnut Hill, MA 02467

Abstract

Many applications need to segment out all small round regions in an image. This task of finding dots can be viewed as a region segmentation problem where the dots form one region and the areas between dots form the other. We formulate it as a graph cuts problem with two types of grouping cues: short-range attraction based on feature similarity and long-range repulsion based on feature dissimilarity. The feature we use is a pixel-centric relational representation that encodes local convexity: Pixels inside the dots and outside the dots become sinks and sources of the feature vector. Normalized cuts on both attraction and repulsion pop out all the dots in a single binary segmentation. Our experiments show that our method is more accurate and robust than state-of-art segmentation algorithms on four categories of microscopic images. It can also detect textons in natural scene images with the same set of parameters.

1. Introduction

Finding dots, i.e. *small round regions*, in an image is a frequently encountered task in medical and scientific research. The dots could be microscopic views of cochlea haircells, epithelial A549 cells, HEK293T embryonic kidney cells or silicon wafer defects (Fig. 1). Counting these dots, locating them, and measuring their intensity are important for understanding hearing mechanism, cancer development, or material properties. Given the large number of dots in each image and the large number of images in these applications, it is essential to have a computer vision algorithm which extracts these dots automatically.

Finding dots is a challenging segmentation problem. These microscopic images often have poor imaging quality (Fig. 1a), large intensity variation (Fig. 1b-c), and extensive occlusion and conjunction between dots (Fig. 1c-d). Even to the human eye, while fuzzy haircells do pop out, low-contrast A549 and kidney cells need scrutinizing, and conjoined silicon pits require thinking to separate them.

Our goal is to develop an algorithm that is capable of finding dots in all these types of images (Fig. 2).

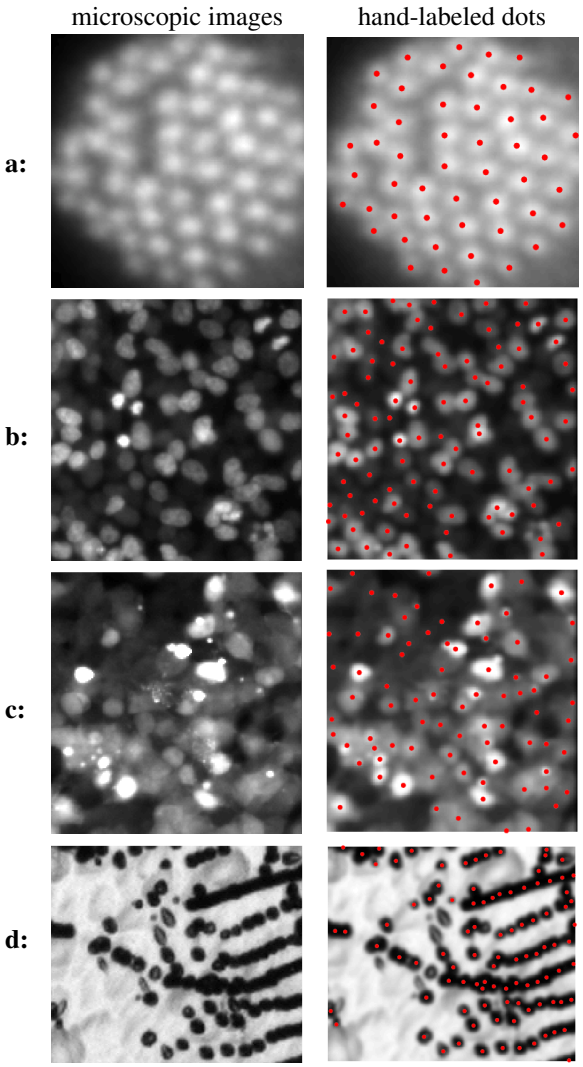

Figure 1. Finding dots in an image is a challenging segmentation problem when the image has poor imaging quality, large intensity variation, occlusion and conjunction between dots. These dots could be: **a)** Haircells in hearing research, courtesy of Pathak and Corey at Harvard University, **b,c)** Hoechst stained nuclei of A549 cells and HEK293T embryonic kidney cells in cancer research, courtesy of Sosale at UPenn, **d)** Etch pit dislocations of silicon wafer in material research, courtesy of Bertoni at MIT.

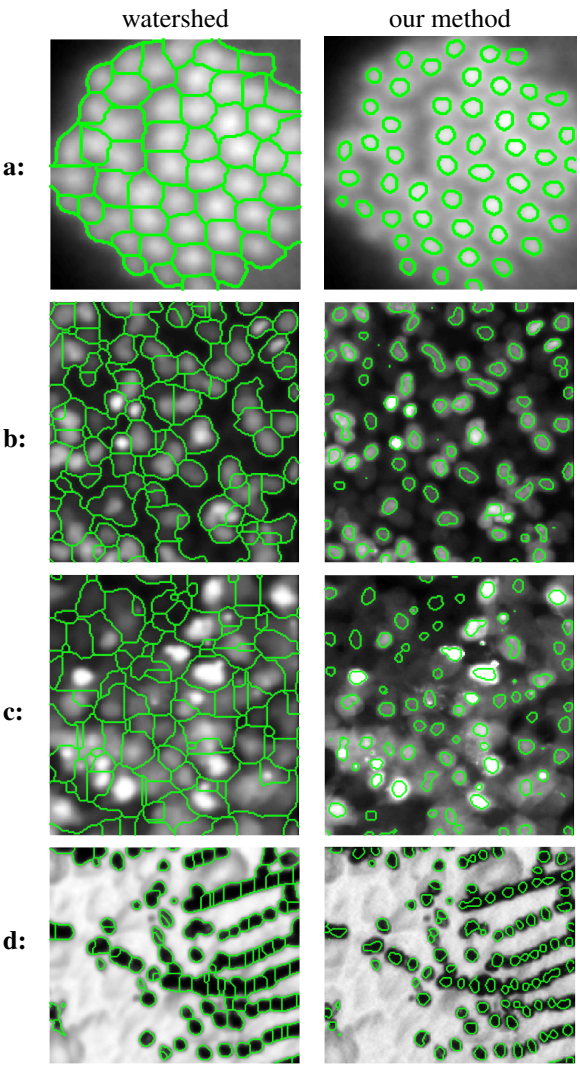

Figure 2. Our algorithm is more accurate and robust at finding dots in 4 types of images, all with the same set of parameters.

Image segmentation is conventionally formulated as separating regions of homogeneous features such as intensity and texture [6, 5, 12, 4, 9]. However, the difference of features in adjacent regions is not always large enough to separate them completely, creating gaps along region boundaries. A common remedy for completing the gaps is to include in the formulation a prior term which favours a segmentation with smooth boundaries [11, 7, 14, 18, 13].

All these traditional formulations of segmentation view regions as solid entities occupied by pixels, and boundaries as abstract lines taking up zero space in-between.

While precise region delineation is useful for image manipulation (Fig. 3a) or object recognition and grasping, it is not necessary for our dot finding (Fig. 3b) and many other applications. Where boundaries should be located is flexible, so long as the core of each dot is retained in the region.

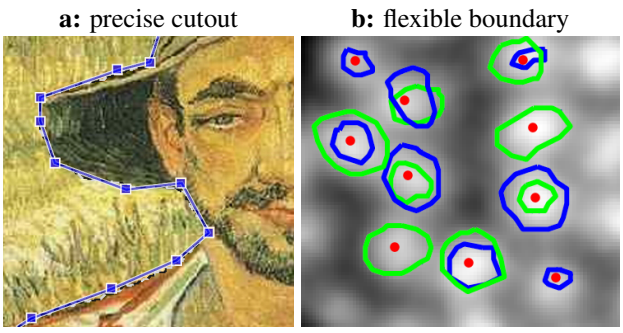

Figure 3. Precise boundary delineation is useful for image manipulation but not necessary for our dot finding applications. **a)** Image cutout (Li et al, Lazy Snapping, SIGGRAPH 2004) is an editing tool which produces a segmentation that follows the facial contour precisely. **b)** Finding haircells only requires locating the core of dots where the boundaries between dots could be flexible.

If we view boundaries not as regions' dependent existing only in the 1D space, but rather as regions of their own in the 2D space, we can effectively pop out all the dots simultaneously from a two-way region segmentation (Fig. 4).

That boundaries form regions of their own has long been observed in [8], but only as a hazard to real image segmentation which needs to be actively suppressed [10]. The idea is that while edges themselves are good features for *intensity* segmentation, only their statistics over small windows are meaningful features for *texture* segmentation. These window statistics prevent edges from breaking up an area of the same texture, but they also tend to break up an area of the same intensity: Boundaries between black and white areas certainly have different statistics from either the black area or the white area, and thus become regions of their own.

The features that make our dot boundaries regions of their own are not statistics which characterize local textural appearance, but patterns which characterize local geometry. Most evident in Fig. 1d, what allows us to break a long tube into a string of small dots is local convexity created by a few dents. However, instead of measuring local convexity with curvature numbers, we describe it using a distributed relational representation, i.e., each pixel has a pixel-centric flow field, which is a sink for pixels inside the dots (intensity peaks) and a source for pixels outside the dots (Fig. 4*F*).

We formulate our algorithm in the spectral graph cuts framework [12], where pixels are nodes of a weighted graph, and finding dots becomes dissecting the graph based on weighted connections between nodes. When segmentation is viewed as finding regions of homogeneous features, the weights are affinitive. They characterize how much two pixels attract each other to the same group, a larger weight for larger feature similarity. When segmentation is viewed as popping out a collection of core regions from their boundary regions, the weights also need to be divisive. They characterize how much core pixels and boundary pix-

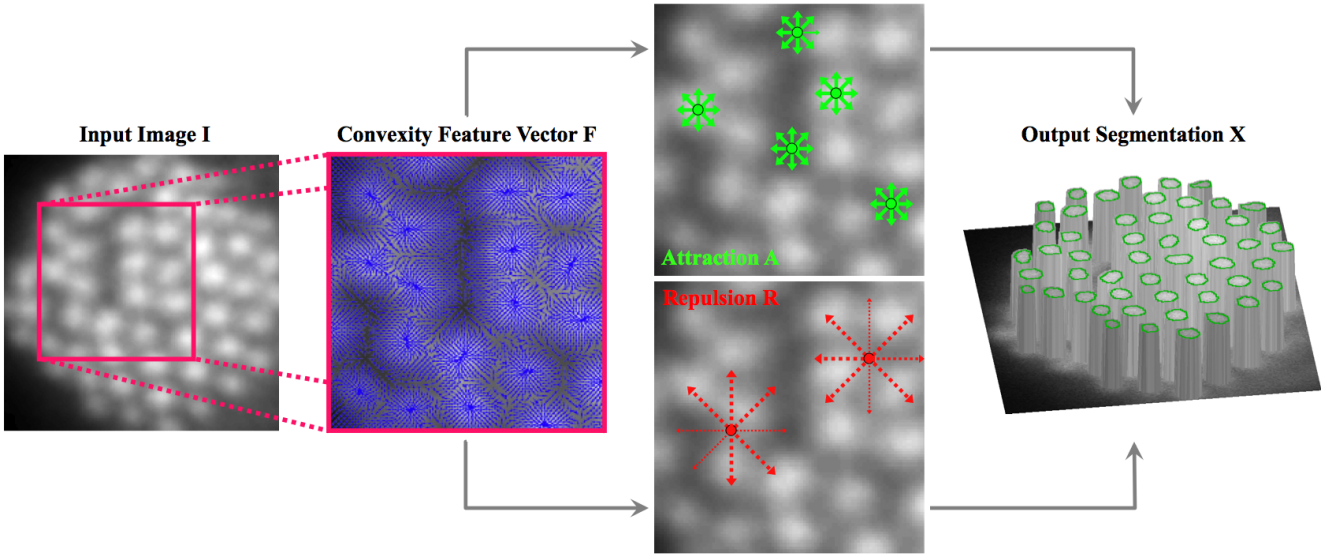

Figure 4. Algorithm overview. Given image I, we first compute a flow field feature vector F which characterizes the geometry of local intensity distributions: A pixel inside a dot (intensity peak) becomes a sink, and a pixel outside the dots (intensity valley) becomes a source. We then compute (green) attraction A between nearby pixels based on similarity in F, and (red) repulsion R between distant pixels based on dissimilarity in F. A two-way node partitioning based on both attraction and repulsion pops out all the dots from their backgrounds.

els repel each other, a larger weight for larger feature dissimilarity. If attraction binds pixels locally both inside and outside the dots, repulsion actively binds all dots to form one group against their common boundaries (Fig. 4A, R). The best segmentation cuts off connections between pixels of minimal attraction and maximal repulsion, popping out all the dots in a two-way region segmentation (Fig. 4X).

We detail our model in Section 2, present experimental results in Section 3, and conclude in Section 4. Our method works better than a few segmentation algorithms on finding dots in a variety of microscopic images, and it works well on real images, all with the same set of parameters. It is potentially a practical tool for a wide range of applications.

2. Finding Dots with Spectral Graph Cuts

We formulate the dot finding problem as a weighted graph partitioning problem, where nodes denote pixels, weights attached to edges connecting two nodes encode grouping cues between the pixels, and finding dots becomes a node bipartitioning problem: pixels inside the dots form one group, and pixels between dots form the other group.

Given an image I, we first compute the feature vector F at each pixel, which is a flow field with sinks and sources characterizing intensity peaks and valleys respectively, then establish short-range attraction A with feature similarity, and long-range repulsion with feature dissimilarity, and finally use normalized cuts with attraction and repulsion to obtain a two-way segmentation [15, 16].

2.1. Pixel-Centric Convexity Feature Vector F

Since dots are small round regions of bright pixels, we first attach to each pixel a peak direction vector $p(i)$ that indicates where pixels of higher intensity are located in its local convex vicinity. Let $L(i)$ denote the 2D location of pixel i in the image, and $|\cdot|$ the L_2 norm of a vector. Consider pixel i and pixel a in its neighbourhood $N(i)$. If a can be reached in a straight line from i with nondecreasing intensity, a is a higher intensity pixel in the same convex region. $p(i)$ computes the average direction from neighbour a's, weighted by the total nondecreasing intensity $T(i,a)$ along the straight line from i to a:

$$p(i) \propto \sum_{a \in N(i)} T(i,a)(L(a) - L(i)),\ |p(i)| = 1 \quad (1)$$

$$T(i,a) = \sum_{\substack{I(m_1) \le I(m_t) \le \ldots \le I(m_k) \\ m_1 m_2 \ldots m_k = \text{line}(i,a)}} I(m_t) \quad (2)$$

$p(i)$ can be regarded as pixel i's local estimation of the direction towards the dot it belongs to. Pixels inside a dot have $p(i)$'s pointing towards the center, whereas those between dots have $p(i)$'s pointing away from it (Fig. 5a).

While the vector field $\{p(a) : a \in N(i)\}$ characterizes where pixel i is in the convex shape of a dot, all the directions need to be normalized with respect to $p(i)$ so that pixels (i's) at an equal distance to the center of a dot have similar vector fields no matter how they are oriented towards the dot. We define the pixel-centric feature vector F as:

$$F(i,a) = < p(i), p(a) > \quad (3)$$

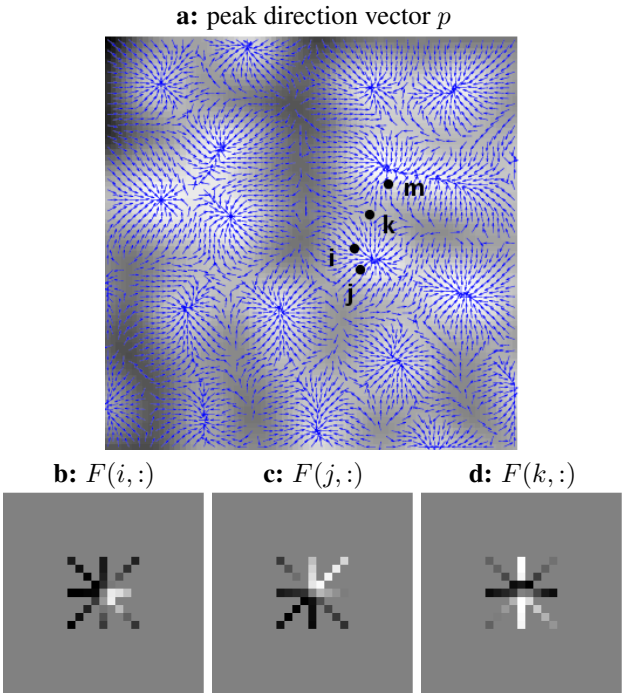

Figure 5. Pixel-centric convexity feature vector F. **a)** Peak vector p at each pixel points towards the center of dot the pixel belongs to. The dot centers and boundaries are sinks and sources in the vector field. **b,c,d)** Feature vector F at pixels i, j, k marked in a). $F(i)$ indicates how much each neighbour agrees with pixel i on $p(i)$, i.e., where it thinks the peak direction is. i and j have different local p fields, but both are inside a dot and have similar F fields which are far different from k, a pixel outside any dot.

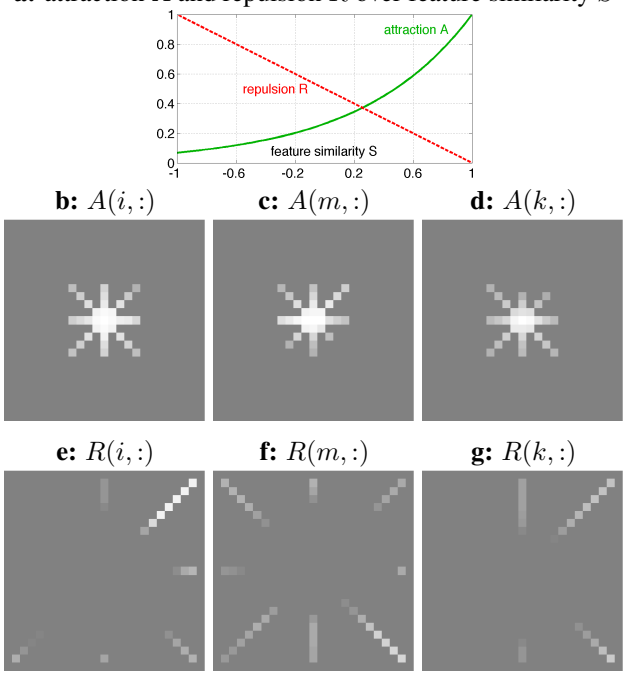

Figure 6. Short-range attraction A based on feature similarity S and long-range repulsion R based on feature dissimilarity $1 - S$. **a)** Attraction A is proportional to S and defined for nearby pixels, whereas R is inversely proportional to S and defined only for distant pixels. **b,c,d)** A and **e,f,g)** R at pixels i, m, k marked in Fig. 5a. Both i and m repel k, causing them to group together.

where $<,>$ denotes vector inner product. $F(i,:)$ shows how much i's neighbours agree with i on the direction the dot lies in, with $p(i)$ itself factored out. Pixels inside a dot have mostly positive values (Fig. 5b,c), whereas pixels between dots have both positive and negative values (Fig. 5d).

2.2. Grouping Cues: Attraction A and Repulsion R

Since the feature vector F is a direction vector, we use the inner product between two feature vectors to measure feature similarity S. The larger the similarity, the larger the attraction, and the smaller the repulsion.

$$S(i,j) = \frac{< F(i,:), F(j,:) >}{|F(i,:)| \cdot |F(j,:)|}, \quad j \in N(i) \tag{4}$$

$$A(i,j) = e^{-\frac{1-S(i,j)}{\sigma}}, \quad |L(j) - L(i)| \leq r_A \tag{5}$$

$$R(i,j) = \frac{1 - S(i,j)}{2}, \quad |L(j) - L(i)| \leq r_R \tag{6}$$

Note that $r_A \ll r_R$, i.e., attraction only operates at a short range to pull pixels in the same dot together, whereas repulsion only operates at a long range to push pixels completely inside dots and pixels completely outside dots apart.

2.3. Graph Cuts with Attraction and Repulsion

Given attraction A and repulsion R between pixels, we segment the image using the normalized cuts criterion [15]:

$$\max \varepsilon = \frac{\text{within-group } A}{\text{total degree of } A} + \frac{\text{between-group } R}{\text{total degree of } R}$$

This criterion can then be written in a matrix form using $n \times 2$ partition matrix X, where $X(i,g) = 1$ if pixel i belongs to group g, $g = 1, 2$. n is the total number of pixels. Let 1_n denote $n \times 1$ vectors of 1's, and $D_W = \text{Diag}(W 1_n)$ the diagonal *degree* matrix for any $n \times n$ weight matrix W.

$$\text{maximize} \qquad \varepsilon(X) = \sum_{g=1}^{2} \frac{X_g^T W X_g}{X_g^T D X_g} \tag{7}$$

$$\text{subject to} \qquad X \in \{0,1\}^{n \times 2}, X 1_2 = 1_n \tag{8}$$

$$U^T X = 0 \tag{9}$$

$$\text{where} \qquad W = A - R + D_R, \tag{10}$$

$$D = D_A + D_R \tag{11}$$

U is an $n \times c$ constraint matrix: If pixels a and b are known to belong in the same region (e.g. from a background mask), we have one constraint $X(a,:) = X(b,:)$, i.e. $U(a,k) = 1$, $U(b,k) = -1$ as the k-th constraint in matrix U.

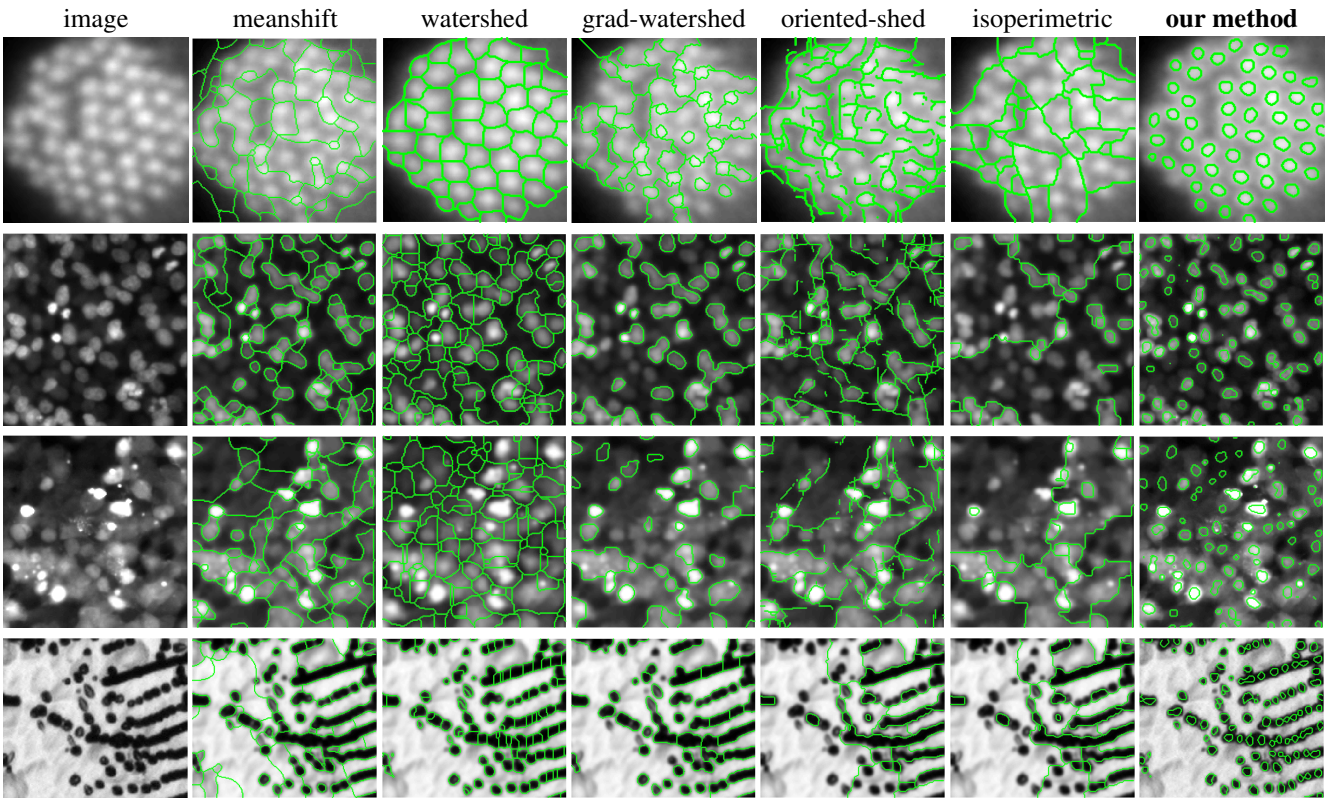

Figure 7. Comparison of results on the 4 representative images in Fig. 1. Meanshift cannot break up conjoined dots. Standard watershed tends to oversegment, and gradient-based watershed misses many weak dots. Oriented watershed does not produce closed regions. Isoperimetric segmentation tends to under-segment the image. While all these methods require some post-processing, our method produces isolated dots all at once in a two-way region segmentation.

We follow the solution procedure developed in [17, 16] and use their code online to find a near-global optimum to this constrained normalized cuts problem.

3. Experiments

We implement our algorithm in MATLAB and apply it to 4 sets of microscopic images as well as real scene images. Each microscopic dataset has 60 images, provided by our collaborators and considered representative of their images.

The dot diameter ranges from 8 to 20 pixels. The same set of parameters are used for all our results: $\sigma = 0.75$, $r_A = 4$, $r_B = 12$. We threshold R to remove repulsion cues from intensity peaks: $R(i,j) = 0$, if $\sum_j F(i,j) < -10$. No constraints (Eqn. 9) are used except for dislocation images, where a background mask from intensity thresholding is applied to avoid extracting unwanted lighter shapes.

While our space complexity is more than the traditional normalized cuts with attraction cues only, our time complexity is often less. In particular, since we are not treating one dot as one region [3], but treating all the dots as a single region, we get all the dots in a two-way segmentation.

We compare our method to 4 other algorithms (Fig. 7).

Meanshift: We use a local implementation [2]. It enhances intensity differences, but it is sensitive to scale choices and cannot break up dots based on convexity.

Watershed: We use MATLAB's built-in function in two different ways: Watershed is directly applied to either the intensity image or the gradient magnitudes (with radius 5) of the image, in the same procedure as MATLAB's demo on Marker-Controlled Watershed Segmentation. While the standard watershed results tend to be over-fragmented in the presence of local intensity fluctuation, the gradient-based watershed results tend to be under-segmented and miss many dots of weak contrast. We drop the latter from further evaluation.

Oriented watershed: We use the implementation provided by [1]. It relies on Pb [10] to clean up watershed lines and has been shown to work well on most natural images. The inadequacy of Pb on our images causes it to have worse results than the standard watershed.

Isoperimetric: We use the implementation provided by [6]. It recursively partitions an image into two regions depending on initial seeds automatically chosen from pixels of minimal intensity. Its results are under-segmented and lack sensitivity to local convexity.

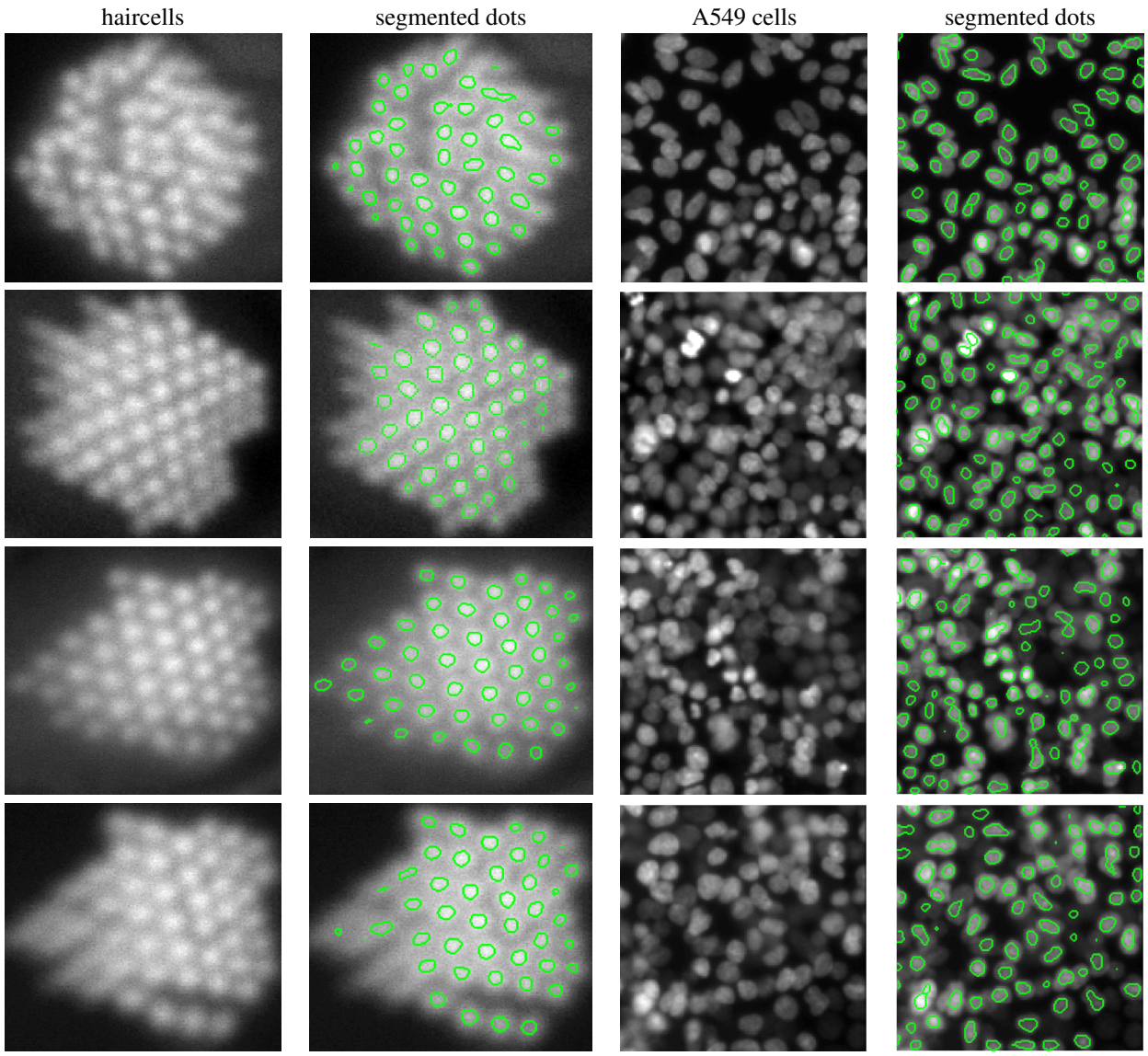

Figure 8. Our sample results on haircells and A549 cells. We can segment out clear or fuzzy dots in a near-regular or irregular layout.

While these algorithms are not designed to find dots, their results demonstrate the difficulty and issues involved in segmenting dots. Unlike any of these algorithms, our method does not require post-processing and produces all the isolated dots at once in all these images (Fig. 8, Fig. 9).

We benchmark these results against human labeled dot centers. Given m ground-truth dot centers and n segment centers for an image, let D_{ij} be the Euclidean distance between dot i and segment j. If it is less than a certain radius threshold ρ, we consider (i, j) a matched detection.

$$\text{precision} = \frac{\#\{j : \min_{i=1}^{m} D_{ij} \leq \rho\}}{n} \tag{12}$$

$$\text{recall} = \frac{\#\{i : \min_{j=1}^{n} D_{ij} \leq \rho\}}{m} \tag{13}$$

The precision measures how many true dots are picked out in the segmentation, and the recall measures how many segmented dots are in fact true dots. Fig. 10 shows that our method performs much better than these other methods. The data points with perfect precision and lower recall rates correspond to haircell images, where our method fails to detect some extremely blurry haircells in the periphery which usually only experts can pick out (e.g. Fig. 9 Row 4).

While our method is designed to pop out all the dots in a microscopic image based entirely on the convexity flow field feature, Fig. 11 shows that it works equally well on natural scene images. The dots there are small repetitive patterns, or *textons*, in frontal, or slanted, tilted, and perspective views. These interesting results open up new possibilities for shape from texture algorithms.

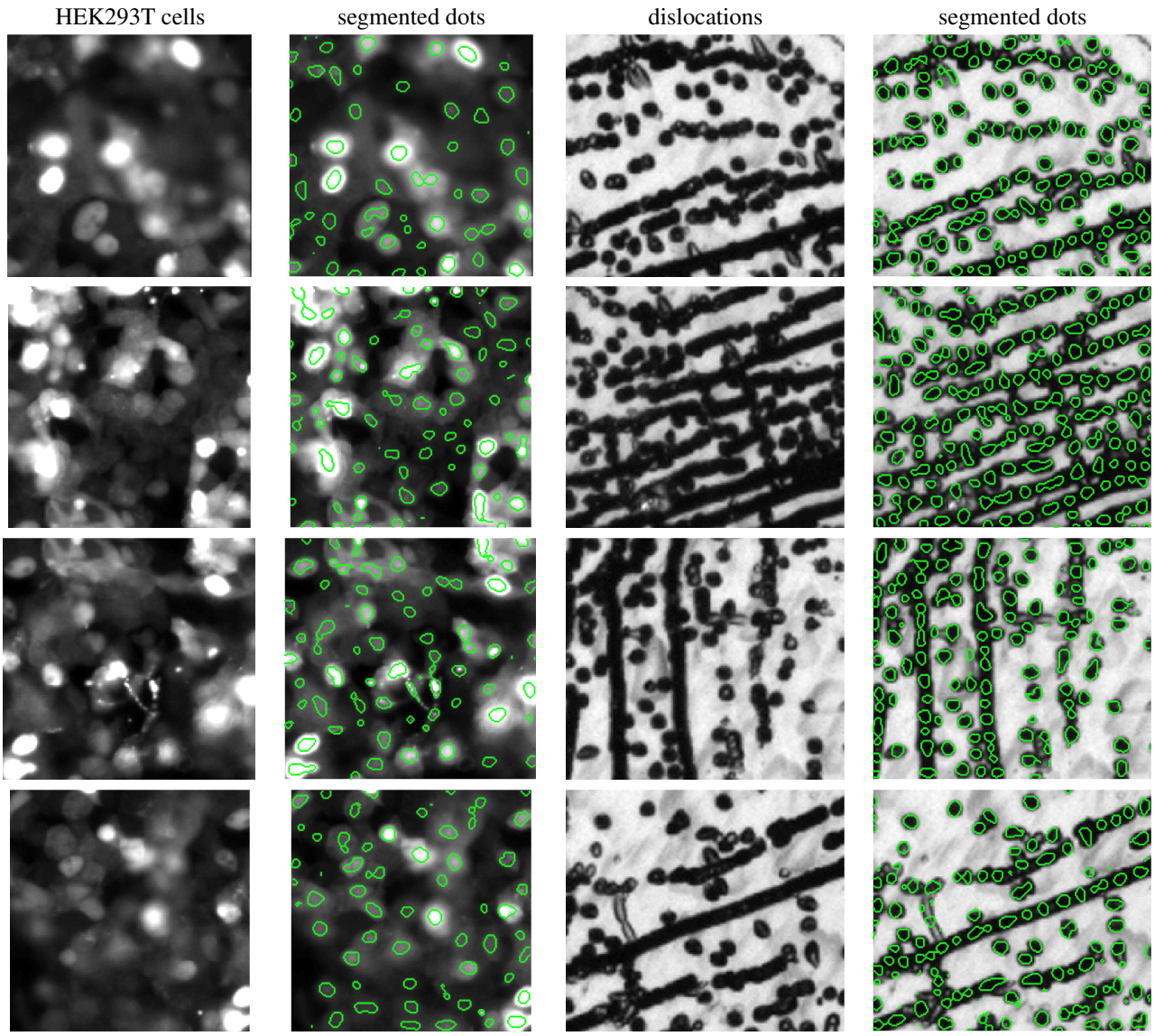

Figure 9. Our sample results on HEK293T kidney cells and silicon wafer dislocation Images. We can segment out dots independent of the size of their intensity contrast and even in the presence of partial occlusion and dot conjunction.

4. Conclusions

Finding dots is a challenging image segmentation task which naturally arises in a diversity of applications. We develop a spectral graph partitioning algorithm that pops out all the dots in a single two-way region segmentation. There are three key components to our algorithm.

The first is viewing dot boundaries as flexible regions of their own. Many applications do not require precise segmentation, neither does human vision. There is, however, a deeper computational reason. When too much emphasis is put on the precision of boundary locations and shapes (as in all traditional image segmentation methods), it is hard for the desired segmentation to become the dominant optimum of any criterion. When that requirement is relaxed, paradoxically the solution is closer to the desired segmentation.

The second is to encode geometry (convexity in particular) in a pixel-centric relational representation. While such a representation is coarse for each pixel, its distributive nature is capable of encoding subtle differences in local convexity with the ensemble of pixels. An extension of our work is to handle more complicated geometry other than convexity.

The third component is to introduce grouping cues of both attraction and repulsion natures. While repulsion from feature dissimilarity seems to encode the same attraction cue of feature similarity, as it operates at a different (larger) spatial range, it plays an active and complementary role to local attraction. Popout would not be possible without re-

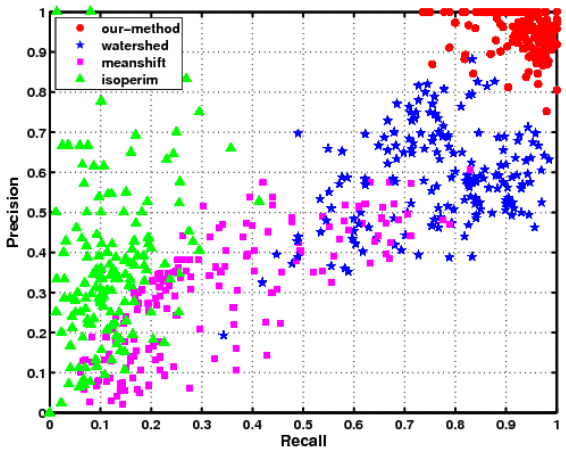

Figure 10. Precision-recall statistics for meanshift, watershed, isoperimetric, and our method on 4 categories of microscopic images. The oriented watershed shown in Fig. 7 is not included as it often does not produce closed dot regions. Our method (red round dots, upper right corner) has better precision and recall overall.

pulsion. While the mechanism of attraction and repulsion in spectral graph theory has been elucidated in [15], its utility has never been demonstrated on any visual tasks. Our work is in fact the first successful application of attraction and repulsion to real segmentation problems.

Our method can also detect textons in natural scene images without feature templates. These results provide exciting new possibilities for other computer vision algorithms.

Acknowledgements

This research is funded by NSF CAREER IIS-0644204 and a Clare Boothe Luce Professorship to Stella X. Yu.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. 2009. 5

[2] D. Barash and D. Comaniciu. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image Vision Comput.*, 22(1):73–81, 2004. 5

[3] E. Bernardis and S. X. Yu. Robust segmentation by cutting across a stack of gamma transformed images. In *EMMCVPR*, 2009. 5

[4] L. Grady. Fast, quality, segmentation of large volumes - isoperimetric distance trees. In *ECCV*, 2006. 2

[5] L. Grady. Random walks for image segmentation. *PAMI*, 28(11):1768–1783, 2006. 2

[6] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *PAMI*, 28:469–75, 2006. 2, 5

[7] T. Leung and J. Malik. Contour continuity in region based image segmentation. pages 544–59, 1998. 2

[8] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 2001. 2

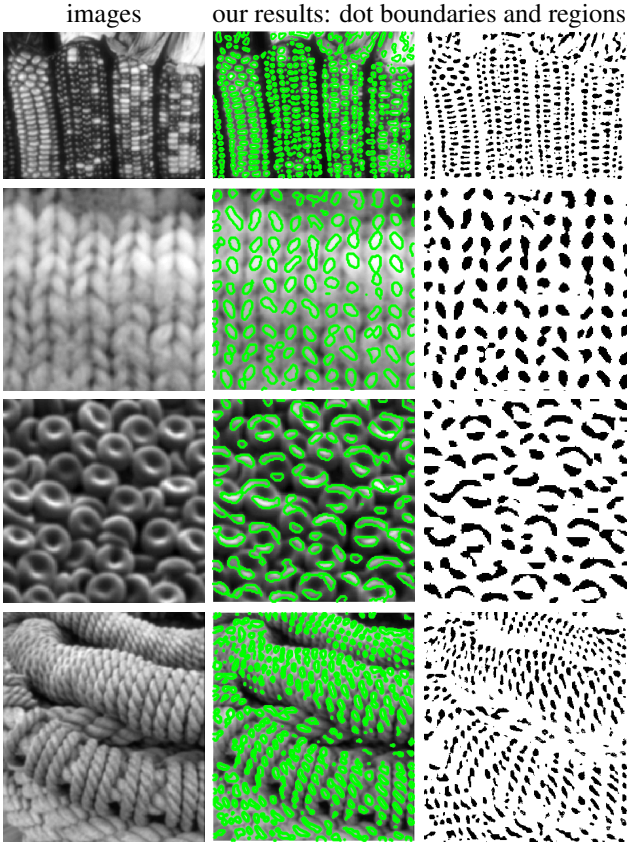

Figure 11. Finding dots in natural scene images with the same set of parameters used for microscopic images. These dots, i.e. small repetitive patterns, are useful for recovering shape from texture.

[9] R. Malladi and J. A. Sethian. Level set methods for curvature flow, image enchancement, and shape recovery in medical images. In *Proc. of Conf. on Visualization and Mathematics*, pages 329–45. Springer-Verlag, 1997. 2

[10] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004. 2, 5

[11] F. Meyer. Topographic distance and watershed lines. *Signal Process.*, 38(1):113–125, 1994. 2

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000. 2

[13] T. Windheuser, T. Schoenemann, and D. Cremers. Beyond connecting the dots: A polynomial-time algorithm for segmentation and boundary estimation with imprecise user input. In *ICCV*, Kyoto, Japan, 2009. 2

[14] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. on Image Processing*, 1998. 2

[15] S. X. Yu and J. Shi. Understanding popout through repulsion. In *CVPR*, 2001. 3, 4, 8

[16] S. X. Yu and J. Shi. Multiclass spectral clustering. In *ICCV*, 2003. 3, 5

[17] S. X. Yu and J. Shi. Segmentation given partial grouping constraints. *PAMI*, 26(2):173–83, 2004. 5

[18] Q. Zhu, G. Song, and J. Shi. Untangling cycles for contour grouping. In *ICCV*, 2007. 2