

FINDING DOTS IN MICROSCOPIC IMAGES

Elena Bernardis

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial
Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2011

Stella X. Yu, Adjunct Assistant Professor
Computer and Information Science
Clare Boothe Luce Assistant Professor
Computer Science, Boston College
Supervisor of Dissertation

Jianbo Shi, Associate Professor
Computer and Information Science
Supervisor of Dissertation

Jianbo Shi, Associate Professor
Computer and Information Science
Graduate Group Chairperson

Dissertation Committee

Kostas Daniilidis, Professor
Computer and Information Science

Jean Gallier, Professor
Computer and Information Science

Camillo J. Taylor, Associate Professor
Computer and Information Science

James Gee, Associate Professor
Radiology

Alyosha A. Efros, Finmeccanica Associate Professor
The Robotics Institute and Computer Science, Carnegie Mellon University

FINDING DOTS IN MICROSCOPIC IMAGES

COPYRIGHT

2011

Elena Bernardis

Acknowledgments

No thesis is ever completed in isolation. While many people have played an important role in my thesis, there are some who deserve special recognition.

I would like to express my deep gratitude towards my advisors, Stella X. Yu and Jianbo Shi. Stella provided me with valuable guidance for shaping both my research and its presentation. The work on dots would have not been possible without her. Jianbo helped me lay the foundations of my computer vision knowledge and always encouraged me to ask interesting questions while never settling for easy solutions.

My committee members, Kostas Daniilidis, Alyosha Efros, Jean Gallier, James Gee and C.J. Taylor provided me with useful comments and feedback that improved the overall thesis presentation and therefore I am indebted to them as well. I would especially like to acknowledge Alyosha for the work on textures and deeply thank him for being my unofficial mentor over the past few years. A long overdue thanks goes to Jean for suggesting many years ago to ‘look into computer vision’, needless to say, my life would have taken a very different path without his insight.

The time spent in the GRASP lab would have not been the same without the presence of an amazing group of people. I would like to thank current and past members of the vision group: Roy Anati, Jeffrey Byrne, Katerina Fragkiadaki, Sandy Patterson, Gang Song, Praveen Srinivasan, Alexander Toshev, Liming Wang, Weiyu Zhang, Qihui Zhu, to name a few, for making the lab a collaborative and positive space. I would especially like to thank Roy for being a great cubicle mate and for his friendship over the years. A separate thank you goes to Phillipos Mordohai for the guidance he provided while being a

postdoc in the lab. I am also very grateful to have crossed paths while in the lab with three wonderful people that also became great friends: Marilina de Gennaro, Alireza Tahbaz-Salehi and Michael Zavlanos. From the copyright of the ‘weeeee’, to the hours spent around Phila taking photographs, their friendship over the years goes beyond what I could ever express with words.

Finally, and most importantly, I would like to thank my sister Sarah for being there from the beginning, with her unique humor and endless support. To her, I dedicate this thesis.

ABSTRACT
FINDING DOTS IN MICROSCOPIC IMAGES

Elena Bernardis

Stella Y. Xu and Jianbo Shi

Extracting and counting numerous ‘dots’, i.e. small round regions, in large microscopic images is encountered in a wide range of medical and scientific research, from studying human cells for cancer prognosis to counting silicon wafer defects for solar cells improvement. Extracting dots is a challenging segmentation problem: faint boundaries and low contrast between regions, large intensity variations within the regions and conjoined clusters of dots make some dots hard to tease apart even upon close inspection. Background clutter or a variety of different shapes in the background can increase the challenge even further.

This thesis presents a constrained spectral graph partitioning framework to deal with the fine granularity of these small structures together with the ongoing challenge of dealing with the complexity associated with increasing image sizes. The segmentation of the entire image is obtained from a set of patch segmentations which are independently derived but subject to stitching constraints between neighboring patches. The constraints come from mutual agreement analysis on patch segmentations from a previous round.

For each individual segmentation, we introduce our ‘Finding Dots’ model to popout dots simultaneously as many disconnected components of one common foreground. We note that many applications do not require precise segmentation and model this by viewing dot boundaries as flexible regions of their own. By distancing ourselves from all traditional image segmentation methods that emphasize precision of boundary locations and shapes, we obtain a solution that is paradoxically closer to the desired segmentation. The features we use are a pixel-centric relational representation that encode local geometry. We introduce two types of grouping cues: short-range attraction based on feature similarity and long-range repulsion based on feature dissimilarity. Repulsion is at the basis of the dots popout: it plays an active and complementary role to local attraction as it operates

at a different (larger) spatial range. Our work is in fact the first successful application of attraction and repulsion to real segmentation problems.

Finally, we exploit the complementary information given by region segmentation and contour grouping to present another way to incorporate local shape information to segment objects with faint boundaries along regions of low contrast. The information of the most salient region segments is combined together with the edge map obtained from the responses of an oriented filter bank. This enables us to define a new contour flow on the graph nodes, which captures region membership and enhances the flow in the low contrast or cluttered regions. The graph setup and our proposed region based normalization give rise to a random walk that allows bifurcations at junctions arising between region boundaries and favors long closed contours. Junctions become key routing points and the resulting contours enclose globally significant regions.

Contents

Acknowledgments	iii
Preface	xvi
1 Introduction	1
1.1 Motivation: Dots are Everywhere	1
1.2 Challenges: Segmenting Dots	5
1.3 Contributions and Outline	9
2 A Watershed Moment in the Watershed Transform: Review of Recent Reformulations	13
2.1 Morphological Segmentation	16
2.2 Energy-Driven Watershed Segmentation: Watersnakes	21
2.3 Watersheds on Graphs: Watershed Cuts	25
2.4 Generalization of Watersheds: Power Watersheds	29
3 Robust Segmentation by Cutting across	
A Stack of Gamma Transformed Images	32
3.1 Introduction	32
3.2 Constrained Cuts with Attraction and Repulsion	35
3.2.1 Graph Representation	35
3.2.2 Criterion with Attraction and Repulsion	36

3.2.3	Partial Grouping Constraints	36
3.2.4	Optimal Solution	37
3.3	Pairwise Grouping Cues from Image Intensities	37
3.3.1	Short-Range Attraction within Individual Peaks	38
3.3.2	Long-Range Repulsion between Peaks	39
3.3.3	Pixel Correspondence and Cue Projection	39
3.3.4	Partial Grouping Constraints	41
3.3.5	Algorithm	41
3.4	Experiments	41
3.5	Summary	44
4	Finding Dots: Segmentation as Popping out Regions from Boundaries	46
4.1	Introduction	46
4.2	Finding Dots with Spectral Graph Cuts	50
4.2.1	Pixel-Centric Convexity Feature Vector F	50
4.2.2	Grouping Cues: Attraction A and Repulsion R	51
4.2.3	Graph Cuts with Attraction and Repulsion	54
4.3	Experiments	55
4.4	Summary	60
5	Pop Out Many Small Structures from a Very Large Microscopic Image	62
5.1	Introduction	62
5.1.1	Challenge 1: Segmenting many small structures	64
5.1.2	Challenge 2: Dealing with a large image	66
5.1.3	Our Solution: Popping out many small structures in a large image	67
5.2	Spectral Graph Partitioning Subject to Stitching Constraints	69
5.2.1	Segmentation with Stitching: Constrained Graph Partitioning . .	70
5.2.2	Local Features and Grouping Cues	71
5.2.3	Stitching Constraints U	75

5.3	Experiments	76
5.3.1	Segmentation Parameters	76
5.3.2	Constraint Propagation and Elongated Structures	78
5.3.3	Benchmark with General Segmentation Methods	82
5.3.4	Benchmark with Domain Specific Methods	85
5.4	Summary	87
6	Shape Extraction through Region-Contour Stitching	93
6.1	Introduction	93
6.2	Graph Setup	94
6.2.1	From Filter Responses to Edge Mask	94
6.2.2	Region Segmentation using Normalized Cuts	95
6.2.3	From Region Segmentation to Lorentz Edge Flow on the Edge Mask	96
6.2.4	Contour Graph Weight Setup	98
6.3	Salient Contour as Persistent Cyclic Random Walk	101
6.4	Circular Embedding for Contour Grouping	103
6.5	Contour Cuts as a Hermitian eigenvalue problem	104
6.6	Computational Solution	106
6.7	Experiments	106
6.8	Summary	107
7	From Dots to 3D Tubes: Structural Correspondence as A Contour Grouping Problem	110
7.1	Introduction	110
7.2	Contour Grouping for 3D Correspondence	112
7.2.1	Untangling Cycles for Contour Grouping	113
7.2.2	Graph Setup for Structural Correspondence	114
7.2.3	Circular Embedding for Random Walk Cycles	117
7.3	Experiments	118

7.4	Summary	119
8	From Dots to Real-Scene Textures	122
8.1	Automatic Parameter Estimation	123
8.2	Experiments on the CMU NRT database	125
8.3	3D Scene Rendering From Textured Images	128
8.4	View-dependent Texture Transfer	130
8.5	Experiments and Discussion	133
9	Conclusions	135

List of Tables

5.1	Benchmark on the ‘Counting Lymphocytes on Histopathology Images’ contest dataset.	86
-----	--	----

List of Figures

1.1	Many small structures in a large image.	2
1.2	Many small and thin structures in a large image.	3
1.3	Segmenting many small and clustered structures.	4
1.4	An ideal isolated dot scenario.	5
1.5	A common dot scenario.	6
1.6	Advantages of more dots.	6
1.7	Typical challenges in segmenting dots	7
1.8	Robustness and efficiency for segmenting many small structures in large images.	8
1.9	Thesis roadmap and contributions	10
2.1	Shortcomings of watershed segmentation	14
2.2	Motivation behind markers in watershed approaches.	15
2.3	Levelings.	18
2.4	Tailored synchronous flooding.	20
2.5	Watershed lines, catchment basins and markers, illustrated in 1D and 2D.	22
3.1	Stereocilia segmentation.	33
3.2	Local intensity fluctuation challenges.	34
3.3	Method Overview.	35
3.4	Pairwise attraction and repulsion.	38
3.5	Segmentation comparison between gamma stacks	40
3.6	Coarse-to-fine stereocilia segmentations.	42

3.7	Segmentation in the presence of noisy and low-contrast images.	43
3.8	Segmentation scores with respect to ground-truth stereocilia centers. . . .	45
4.1	Finding dots challenges	47
4.2	Precise versus flexible boundaries	48
4.3	Finding dots overview.	49
4.4	Pixel-centric convexity feature vector F	52
4.5	Short-range attraction A and long-range repulsion R	53
4.6	Comparison of results on the 4 representative images	56
4.7	Precision-recall statistics for meanshift, watershed, isoperimetric, and our method.	57
4.8	Our sample results on haircells and A549 cells.	58
4.9	Our sample results on HEK293T kidney cells and silicon wafer dislocation Images.	59
4.10	Finding dots or ‘textons’ in natural scene images	61
5.1	Many small structures in a large image.	63
5.2	Segmenting cells challenges.	64
5.3	Segmenting cells challenges.	65
5.4	Efficiency versus robustness dilemma for segmenting small regions in a large image.	66
5.5	Segmentation subject to stitching constraints algorithm overview.	68
5.6	Computing pairwise features and grouping cues.	72
5.7	Local pairwise cues based on feature similarity for three typical scenarios. .	74
5.8	Finding larger dots by increasing the convexity feature radius r_c	77
5.9	Automatic core radius estimation for parameter selection.	78
5.10	Hierarchical dot structures and radii parameters.	79
5.11	Beyond simple stitching.	80
5.12	Thin structures and constraint propagation.	81
5.13	Precision-recall statistics for epithelial cells and histopathology images. .	83

5.14	Sample results on the ‘Counting Lymphocytes on Histopathology Images’ contest dataset.	85
5.15	Results on human epithelial A549 and embryonic kidney HEK293T cells.	88
5.16	Results on HEK cells with a variety of ‘convex shapes’.	89
5.17	Results on histopathology images depicting tumor-like lesions.	90
5.18	More results on histopathology images depicting tumor-like lesions.	91
5.19	Results on EM brain sections of drosophila fly.	92
6.1	Contour-region stitching algorithm intuition.	95
6.2	Region Segmentation and Lorentz force on edges.	96
6.3	Computation of the Lorentz force from the soft region segmentation eigen- vector.	97
6.4	Schematic representation of our contour-region stitching method.	99
6.5	Graph normalization for directed contour random walk setup.	101
6.6	Examples of paths found by sampling.	104
6.7	Extracted contours for several object images taken from the Berkeley seg- mentation dataset.	109
7.1	3D Stereocilia segmentation.	111
7.2	Extracting 3D tubes by finding correspondences across image stacks.	112
7.3	Structural correspondence as a contour grouping problem.	115
7.4	Embedding Space.	116
7.5	Extracting 3D tubes shrinking across image stacks.	120
7.6	Extracting 3D tubes shifting across image stacks.	121
8.1	Textons in real-scene images.	122
8.2	Adaptive radii estimation.	124
8.3	Automatic texton segmentation.	125
8.4	Results on the CMU NRT dataset.	126
8.5	Automatic texton segmentation on complex surface textures from the CMU NRT dataset.	127

8.6	3D texture synthesis.	129
8.7	3D texture synthesis results.	131
8.8	3D texture synthesis results.	132
8.9	Shortcomings of texture synthesis results.	133
8.10	Automatic texton extraction as potential input for 3D texture synthesis. . .	134

Preface

Fisches Nachtgesang (Night Song of a Fish)

-
U U
- - -
U U U U
- - -
U U U U
- - -
U U U U
- - -
U U
-

Christian Morgenstern, 1905

Chapter 1

Introduction

1.1 Motivation: Dots are Everywhere

Counting, measuring, and comparing numerous small structures in a large image is an often encountered task in many research fields based on the analysis of microscopic images, from cancer to hearing research, from materials science to neuroscience, just to name a few. Images can be very different in nature and visual appearance.

In Fig. 5.1, we illustrate a fluorescent image of a frog's inner ear. Hair cells of the inner ear transduce mechanical signals into electrical signals [VKC07] and their localization is vital for medical research on hearing. Each hair bundle is composed of tens of stereocilia organized in an organ-pipe-like formation. When extracting the larger bundles or the 15 pixels cilia from the frog's hairbundles, one has to address complexity issues associated with a 1600×1600 pixels image. Different structures can be seen in Fig. 1.2, a 1800×1800 pixels image of a drosophila's fly brain region from electron microscopy (EM) data. While some researchers are interested in the thin membranes to automatically reconstruct the synaptic circuits in different brain regions of drosophila [TLM08], others, for example, study the blood vessels traversing the slices perpendicularly. Salient regions assume a wider range of shapes in this image, and small round regions (e.g. vesicles) appear densely packed together with thin and elongated structures (e.g. neurons and membranes).

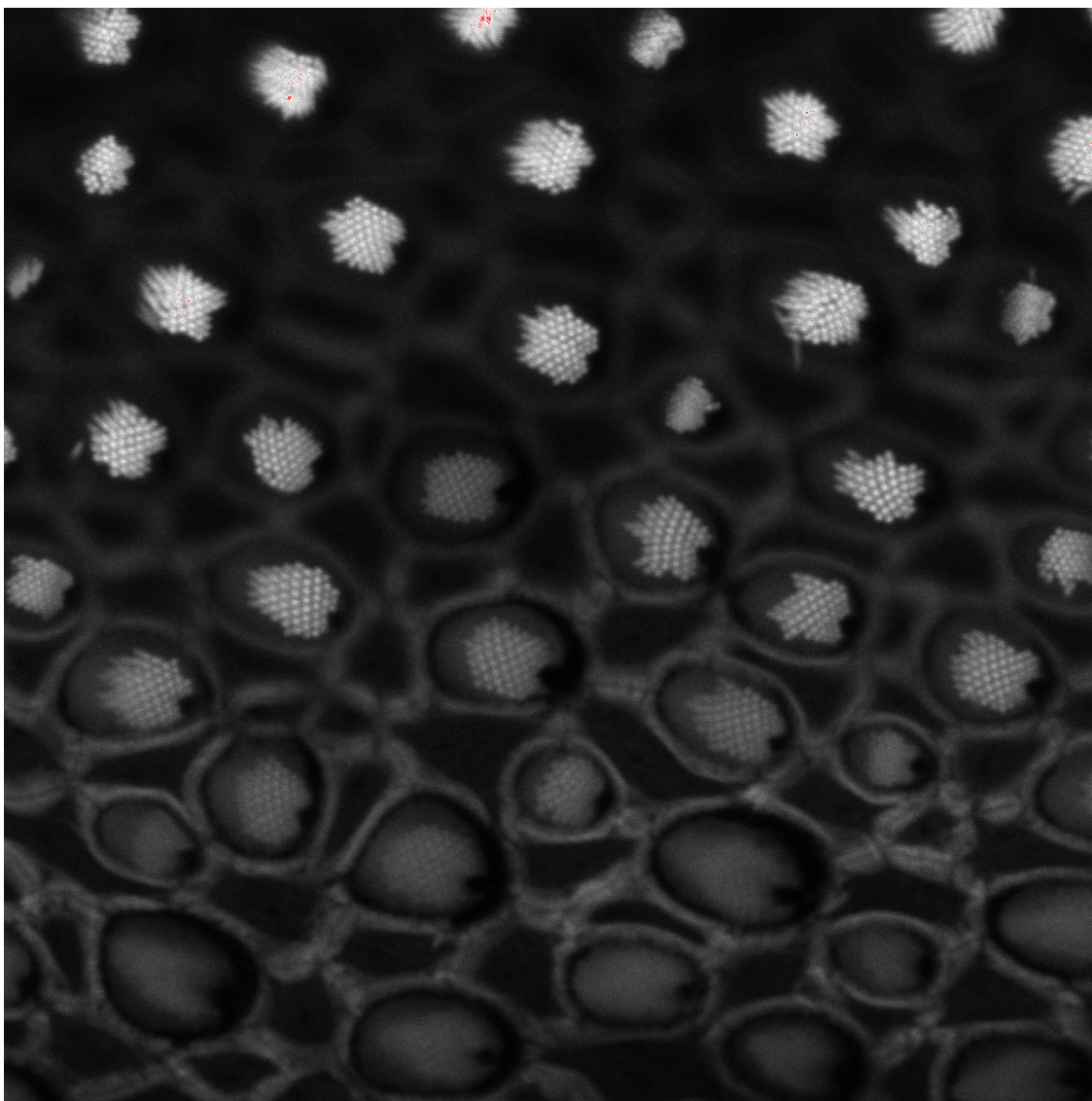


Figure 1.1: Many small structures in a large image. Stereocilia bundles of the frog's inner ear, displayed in a 1600×1600 pixels fluorescent image. Cells, or cilia, are approximately 15 pixels in diameter (Image Courtesy: Medha Pathak and David Corey at Harvard).

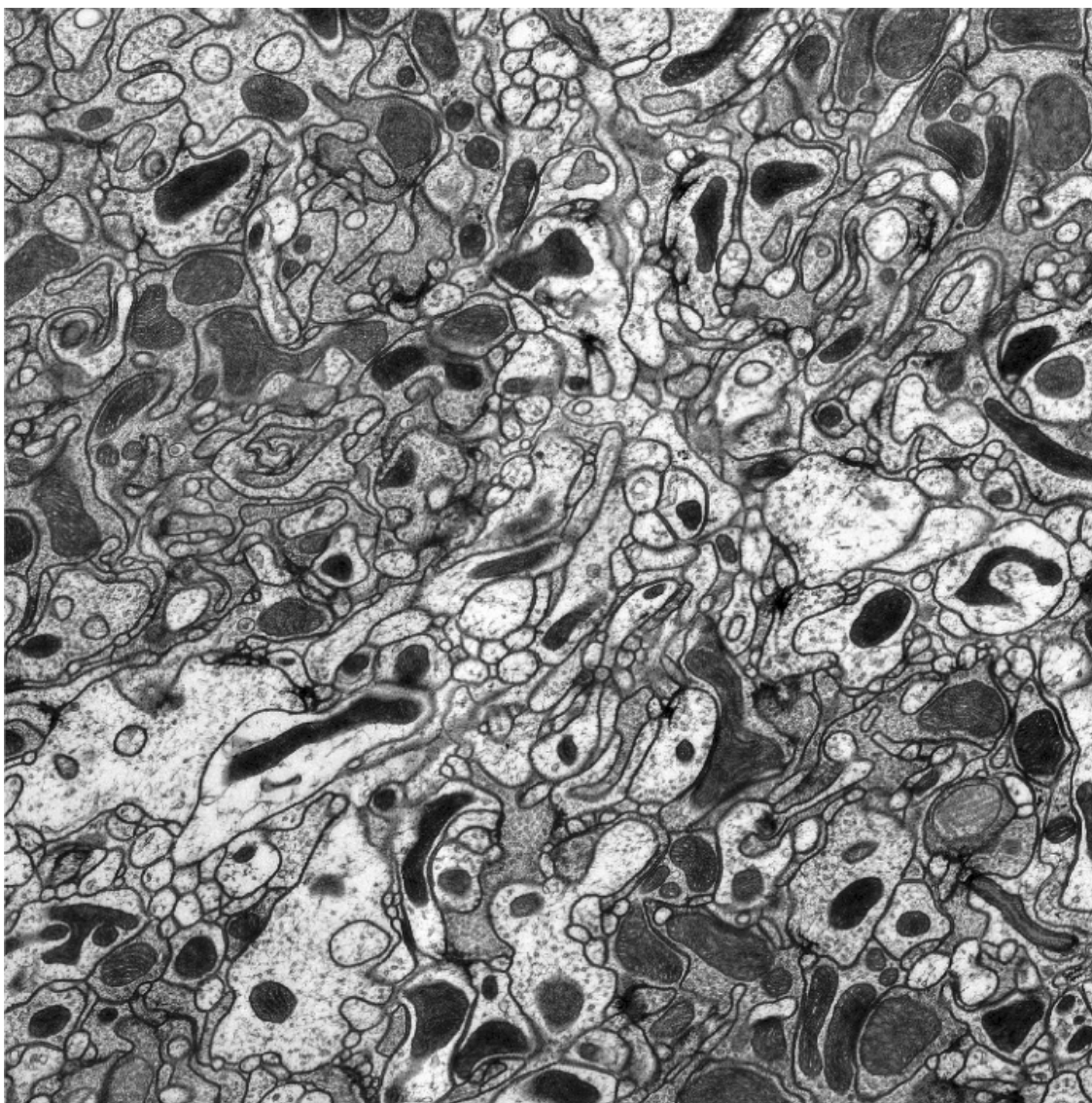


Figure 1.2: Electron microscopy (EM) data from the medulla brain region of the drosophila fly, 1800×1800 pixels, depicting small and thin structures of different shapes densely packed together (Image courtesy: Mitya Chklovskii, C Zhiyuan Lu, Rick Fetter, Shinya Takemura and Ian Meinertzhagen at Janelia Farm Research Institute).

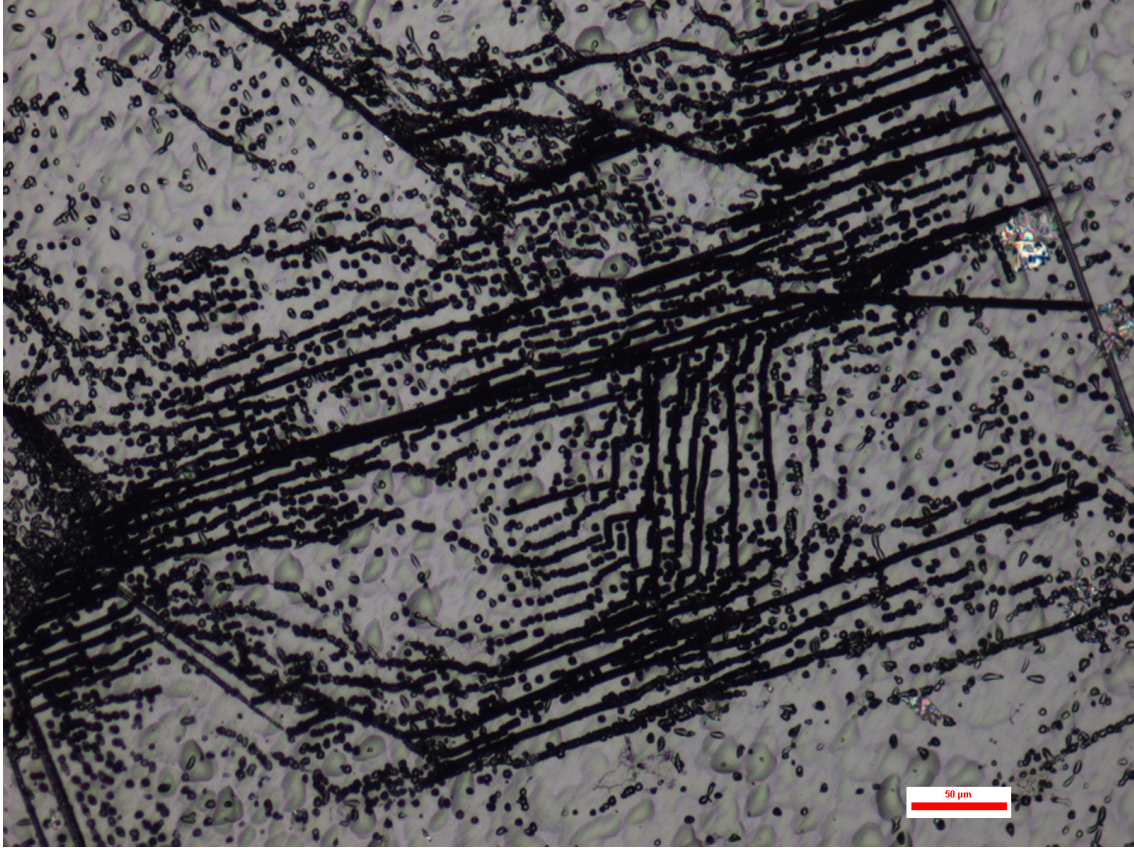


Figure 1.3: Etch pit dislocations in silicon wafer in materials research. Defects are clustered together and hard to tease apart even for humans (Image courtesy: Bertoni at MIT).

Very different in nature and visual appearance is Fig. 1.3: etch pit dislocations in a 1200×1600 pixels silicon wafer. Studying the density of these structural defects is crucial for example in determining the efficiency of solar cells that are manufactured on silicon wafer substrates [HBSB08, BPV⁺11]. Dislocations can be encountered isolated but are usually conjoined and clustered together in what could resemble very compact strings of beads. Separating them out, even for a human eye, requires very close inspection.

In all these images, the challenge of segmenting the numerous small structures is therefore two-fold: fine segmentation granularity when dealing with the size of the small segment and segmentation complexity when dealing with the size of the large image.

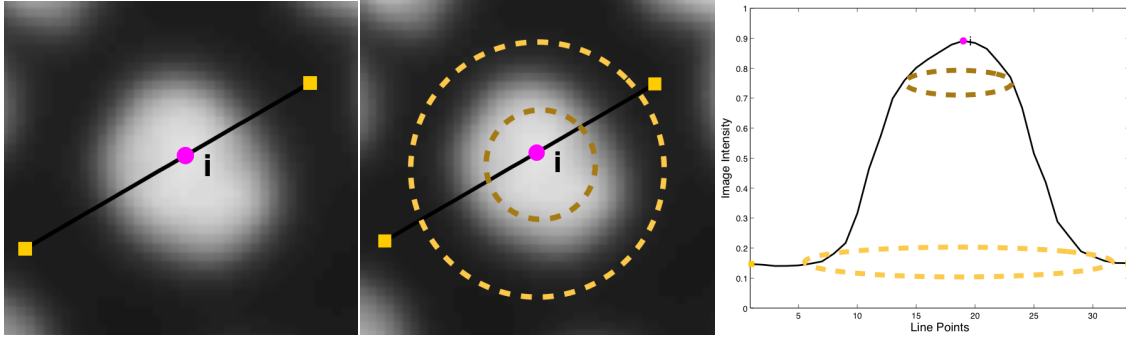


Figure 1.4: An ideal isolated dot scenario. Even in the ideal case, an isolated dot presents a blurry boundary. Delineating its boundary involves deciding where the intensity profile should be broken: a higher threshold yields a tight boundary enclosing only the highest intensity dot center and a lower threshold yields a region enclosing the entire dot.

1.2 Challenges: Segmenting Dots

In this thesis, we focus on extracting the small convex regions, which, given their visual appearance, will be henceforth referred to as ‘dots’. While intuitively very simple, dots can be complex from a segmentation point of view. In microscopic images, they usually appear with blurry boundaries as shown in Fig. 1.4. Even in the simplest isolated case, precise boundary location is subject to human interpretation. While simple intensity thresholding could be sufficient for counting isolated dots in this ideal scenario, in practice, and as illustrated in the figures of the previous section, this rarely occurs.

More common is the case of seeking a blurry dot embedded within a noisy background, as depicted in Fig. 1.5. In order to delineate a contour around the dot, one needs to have some notion of the dot’s shape. Otherwise, with the lack of external guidance, the segmentation easily results in leakages. The same effect results when dots are within proximity of other dots, where the leakage could result in a segmentation clustering multiple dots together. While similar intensity adjacent faint dots are indeed a challenge from the segmentation point of view, it is interesting to note that the presence of other dots can actually be exploited as cues to define the boundaries between the dots. The intuition is depicted in Fig. 1.6: the existence of the second dot prevents the region of the first one to

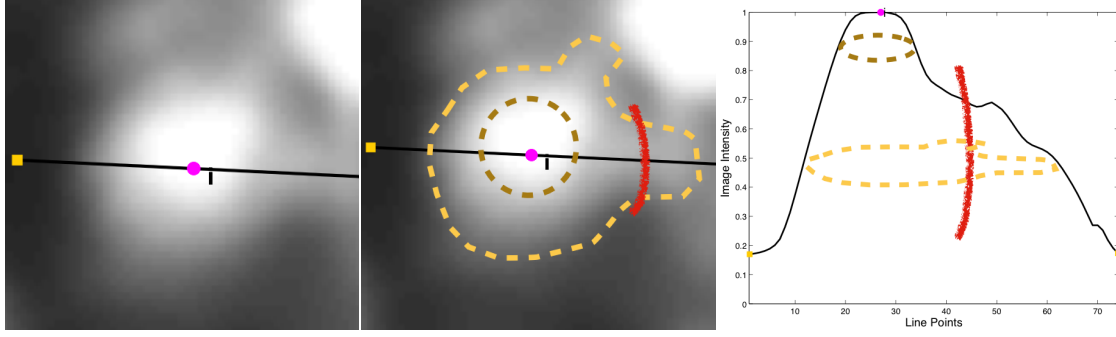


Figure 1.5: A common dot scenario. In practice, intensity fluctuations are usually present, a dot is often encountered embedded within a noisy background. In the absence of dot shape information, the segmentation easily results in leakages.

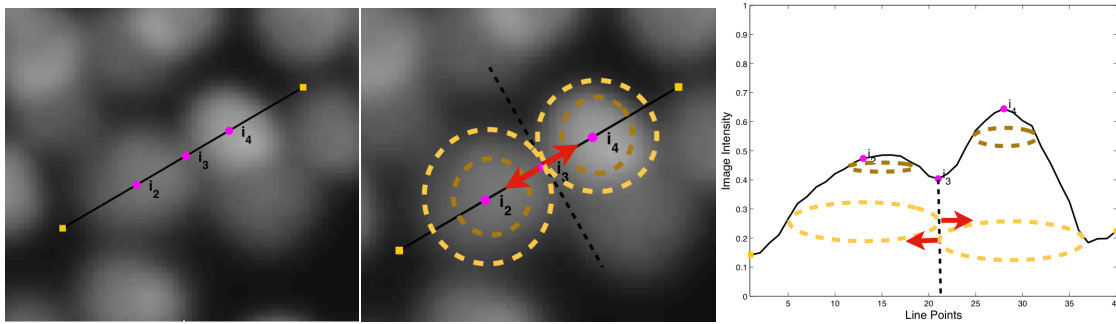


Figure 1.6: Advantage of more dots. The presence of a second dot prevents the region segment of the first dot from leaking into the possible ‘dot’ space of the second one, showing that the occurrence of more dots can be actually beneficial.

leak into the possible ‘dot’ space of its neighbor.

Both the presence of local intensity fluctuations and the clustering of dots have been the underlying reasons of failure of many segmentation approaches. The underlying challenges of typical adjacent dot pair scenarios are illustrated in Fig. 1.7: (1) isolated dots; (2) salient adjacent dots; (3) faint adjacent dots; and (4) one faint dot adjacent to a salient one. Although the watershed transform [Mey94] was initially developed as a non-parametric method for contour extraction of bubbles in a radiologic plate [BL79], similar to the dots in our images, its dependency on local intensity fluctuations oversegments the image in the absence of seeds or markers (Fig. 1.7f). On the other hand, a global approach such as k -means clustering (Fig. 1.7g) fails in the presence of adjacent dots of similar intensities,

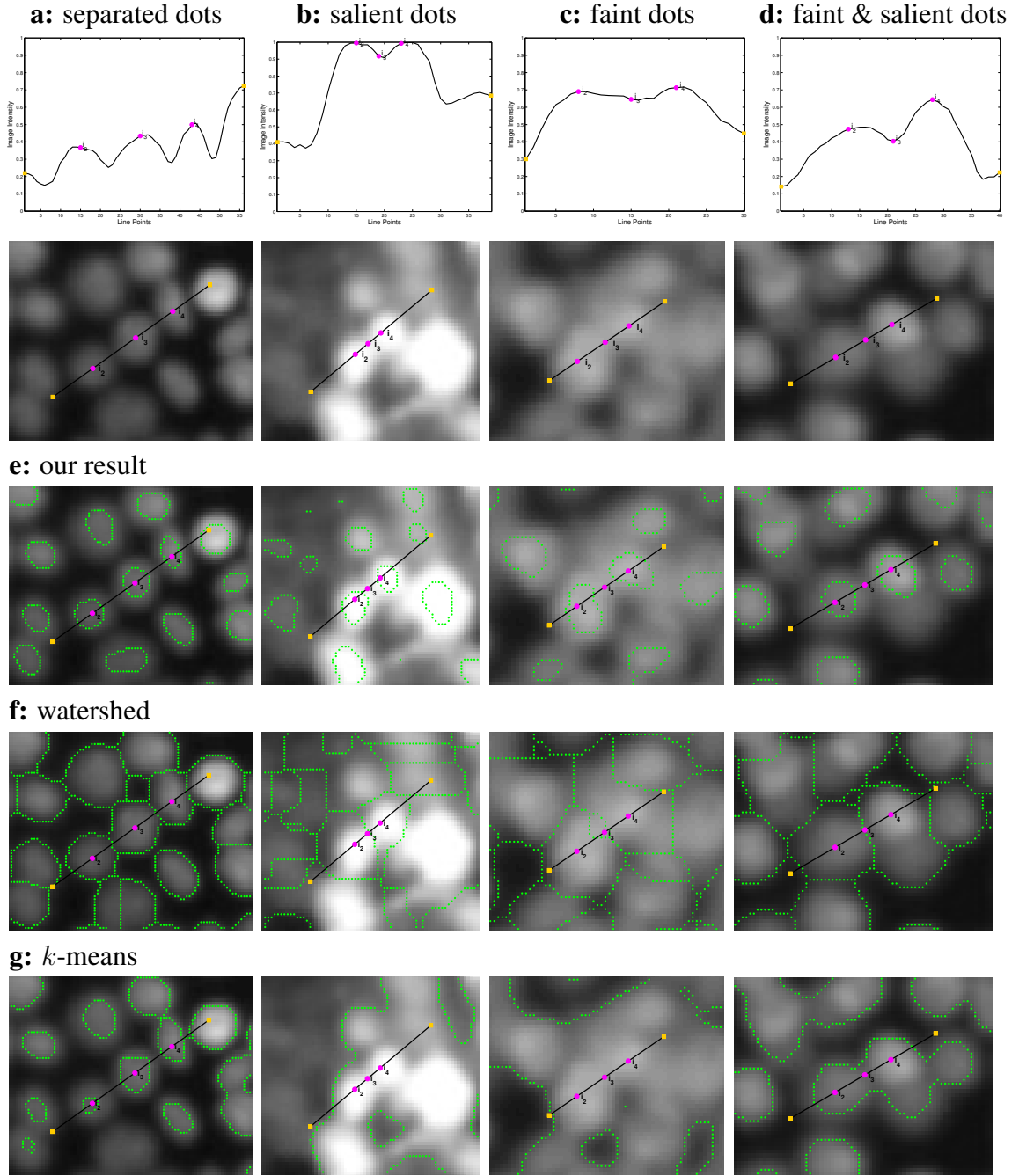


Figure 1.7: Dot scenarios. **a)** separated dots with faint boundaries ; **b)** a pair of salient dots; **c)** a pair of faint dots; and **d)** a faint adjacent to a salient dot. **e)** Our method is able to pick the contours of the dots independently of their size, intensity and geometry, while **f)** watershed tends to oversegment and **g)** k -means clustering is unable to deal with the intensity variation between adjacent dots.

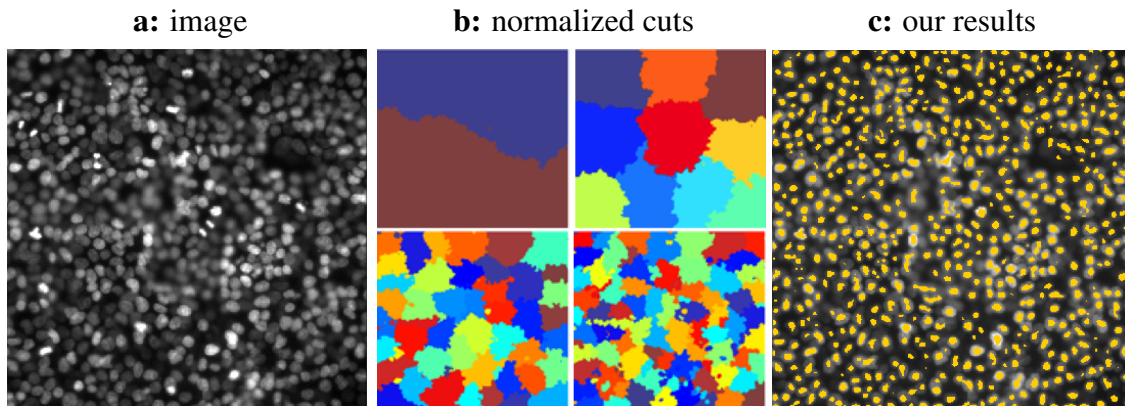


Figure 1.8: **a)** Robustness and efficiency for segmenting many small structures in large images. **a)** Epithelial cells image. **b)** Normalized cuts (N-cuts) for 2, 10, 32, 64 regions is meant to segment large regions in natural images. While robust in general, it fails for these structures. **c)** Our method combines the robustness of normalized cuts while improving its efficiency and is able to find all dots simultaneously in a two-way segmentation (gold).

even when, in case Fig. 1.7d, the two dots peak at different intensities.

Many methods have been proposed to overcome local noise fluctuation dependency. Energy-driven methods involve the minimization of an energy function, and can be formulated either on regions [MS89, GG90, ZY96] or contours, such as level set methods [MS97] and active contours or snakes [XP98]. Despite a number of advances [ZSXZ10, LXGF05], their main limitation remains a high sensitivity to initial seed selection and the inability to tease clustered regions apart. Graph cuts methods [BVZ01] produce a labeling via a maximum flow computation between foreground and background seeds. Their main drawback is a bias towards a ‘small cut’, hence resulting in boundaries that would cut through thin protruded regions. Despite advances to address this issue (e.g. random walker algorithm [Gra06b]), all these methods remain highly dependent of initial seeds selection.

Independent of initial seeds, spectral-graph methods [SM00, Yu05] are prized for their ability to grasp the large structural organization of an image from the global integration of local cues. While this property is desired for understanding a real-scene image, it unnecessarily handles a huge number of pixels in a large image, since segmenting cells

in one region really should not be influenced by cells far from them. It also prevents small structures from being segmented all at once, since a larger image size leads to larger regions instead of numerous small ones given a fixed number of segments.

In this framework, pixels in the image become nodes of a weighted graph and partitioning the image becomes dissecting the graph based on weighted connections between nodes. Attraction cues are commonly used to encode affinity between pixels, but they are not sufficient to extract the many small dots (as shown in Fig. 1.8b); they are based on absolute intensity differences and do not encode geometrical information. Nevertheless, it has the ability in general of cleaning up spurious edges and bridging gaps among missing or faint contours. However, in order to find all the salient segments, oversegmentation is needed. While in extracting larger shapes this leads to fragmentation along the salient shape boundary, in the case of many small dots, it leads to an exceedingly large number of segments.

Our goal is to segment all the dots within a spectral-graph partitioning framework. With a new representation of the pairwise local cues, we encode local shape information in the cues, and exploit the presence of other dots via the crucial role of repulsion, thus allowing all dots to simultaneously popout from a common background (Fig. 1.8c).

1.3 Contributions and Outline

The main focus of this thesis is to extract dots in images by studying pairwise local cues within a spectral-graph partitioning framework (Fig. 1.9). We maintain the robustness of the spectral-graph approach while we successfully extract the numerous small structures by encoding local image geometry in local pairwise cues and using repulsion cues to first repel dots from each other and later to popout dots from their common background.

- Chapter 2 gives a historical overview of morphological segmentation approaches, which deserve special mention given their computational efficiency and widespread use for medical image analysis. We start with the original watershed formulation,

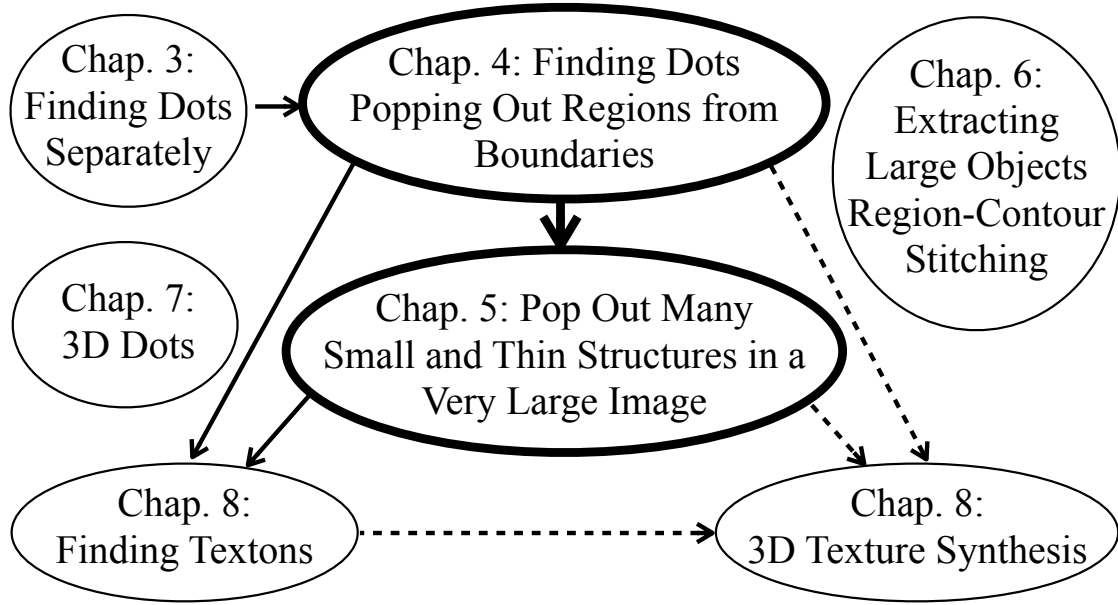


Figure 1.9: Thesis contributions. This thesis focuses on extracting many small regions from an image. Chap. 3 is preamble for dot segmentation, where dots are segmented one by one in a traditional approach. Our main result are presented in the ‘Finding Dots’ model (Chap. 4) and its extension to very large images (Chap. 5), where dots are extracted simultaneously as many disconnected components of one foreground segment. Applications to textons and 3D texture synthesis are given in Chap. 8. While our work concentrates on region segmentation, we also present region extraction from a contour grouping perspective: Chap. 6 focuses on extracting larger objects and Chap. 7 presents a contour grouping approach for solving structural correspondence between dots in 3D space.

and conclude with a recent generalization of watersheds on graphs.

- Chapter 3 presents a spectral-graph partitioning approach to segment dots that is robust to local intensity fluctuation. We perform the segmentation on a stack of gamma transformed versions of the original image and define local pairwise grouping cues that encourage dots to stay together and background regions to divide apart. In our first approach for segmenting dots, we follow the standard segmentation mindset: the goal of the segmentation is to extract one segment for each dot present.
- Chapter 4 presents our core result for finding dots. By taking a new viewpoint for segmenting these dot-like regions with faint boundaries, we simultaneously popout

all dots in a two-way segmentation, as many disconnected components against one common background. Our approach consists of a constrained spectral-graph formulation featuring two types of grouping cues: short-range attraction based on feature similarity and long-range repulsion based on feature dissimilarity. Traditionally overlooked, repulsion cues play a significant role for segmenting this type of image. To construct the local cues, we go beyond local intensity differences and instead use a pixel-centric relational representation to encode local geometry. Our finding dots approach is robust to local intensity fluctuations and automatically extracts all regions of interest simultaneously.

- Chapter 5 addresses the challenges of finding many dots, as well as other small and thin structures in a large image. Extracting these regions poses a dilemma in terms of segmentation granularity due to fine structures and segmentation complexity due to large image sizes. Our key observation is that for images of small repeated structures, only a certain amount of global information is needed; contrary to real-scene images, these small structures in different parts of the image are independent of each other. We propose an extension of our previous constrained spectral graph partitioning framework to obtain a final segmentation from a set of patch segmentations, each independently derived but subject to stitching constraints between neighboring patches. The final segmentation not only stitches solutions seamlessly along overlapping patch borders, but also refines the segmentation in the interiors.
- Chapter 6 goes beyond extracting small dots. The goal is to extract region boundaries of larger objects with faint boundary problems from a contour grouping point of view. Again within a spectral-graph framework, we exploit the complementary information given by region segmentation and contour grouping to present another way to incorporate local shape information to segment objects with faint boundaries along regions of low contrast. Local cues for contour grouping now carry region

information which allows bridging of gaps due to faint boundaries between contours. Contour flows stitch oversegmented regions into large and salient ones which become, therefore, less fragmented.

- Chapter 7 discusses a novel viewpoint to approach structural correspondence across an image stack of dots in 3D space as solving a contour grouping problem. Finding 3D cellular tubes becomes finding closed contours. We derive grouping cues between cells in adjacent slices based on their ability to relate in the 3D space. Those that form long 3D tubes become the most salient contours, while those of shorter lengths become less salient. In a spectral graph-theoretical framework for contour grouping, such a separation by the contour length is reflected in complex eigenvectors of different magnitudes, from which these 3D tubes of varying lengths can thus be extracted, obviating the need for identifying missing correspondences.
- Chapter 8 applies the dot model to extract real-scene textons, i.e. repeated textural elements. By using the segmentation subject to stitching approach, we adaptively estimate parameters in different parts of the image to account for viewing perspective and scale changes. We then present a very simple but effective approach for synthesizing novel views of a given textured surface from a single photograph that could benefit from prior textons extraction.
- Chapter 9 concludes with a summary of the work in this thesis as well as a discussion on directions for future work.

Chapter 2

A Watershed Moment in the Watershed Transform: Review of Recent Reformulations

The watershed transform has been extensively used in image segmentation and lies within the theory of mathematical morphology. The image itself is considered a topographic surface, i.e. as a relief function, and dividing lines, known as *watershed lines*, between regions of interest are computed by following the ‘drop of water’ principle. The watershed lines are a set of separating points from which a drop of water can flow down towards at least two regional minima. These attraction zones, also called *catchment basins* can provide an alternate interpretation of the watershed transform. The watershed lines become the set of points that are equidistant to the minima of the relief function, where the distanced traveled is usually in terms of a topographical distance, i.e. a spatial distance between two points weighted by the gradient norm.

There are many possible relief functions, but it is a common choice to use the morphological gradient (i.e. the image gradient followed by a small dilation and erosion) instead of the original image. Since image gradient computation is very sensitive to intensity fluctuations, applying the watershed transform in an automatic fashion directly on the

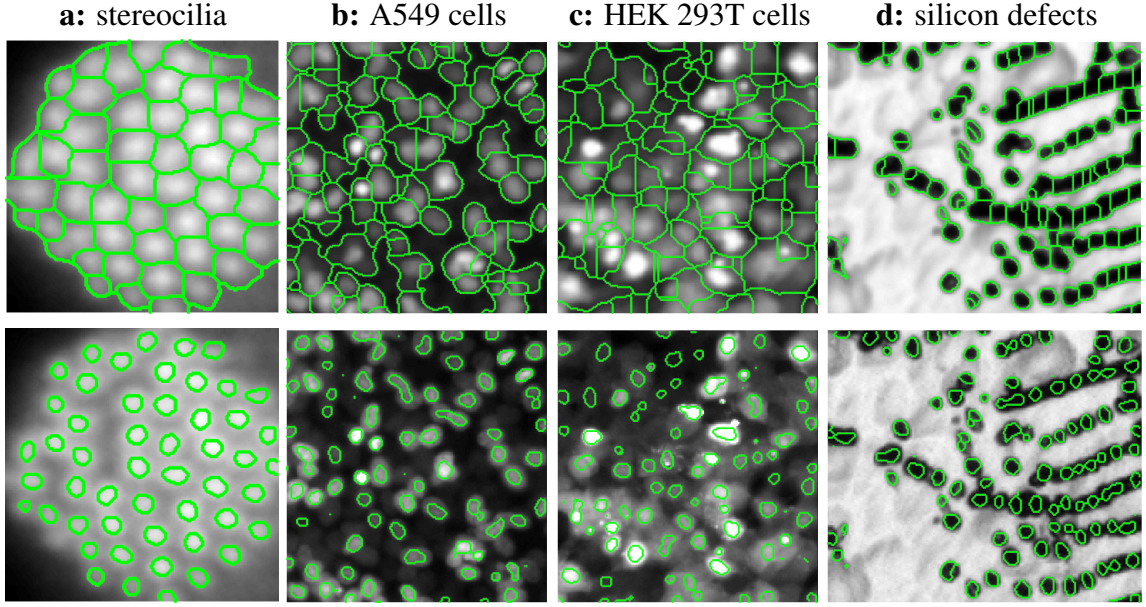


Figure 2.1: Our algorithm (bottom row) is more accurate and robust than watershed examples (top row) at finding dots in 4 types of images, all with the same set of parameters.

morphological gradient image results in severe oversegmentation (Fig. 2.2). A solution is to introduce initial seeds, or markers, either automatically or manually, in order to choose the regions that would be sources for the flooding and drive the subsequent segmentation.

The segmentation can therefore be thought of as a combination of two steps: a first one that relies on the watershed transform to extract an initial set of contours, and a second step that relies on defining markers either manually or automatically in order to refine the contours of interest. Finding the markers represents the most interesting and challenging part, while the watershed transform step itself is done entirely automatically. When the watershed transform step is done automatically, finding the markers turns out to be a very challenging exercise. The traditional method of morphological segmentation (based on watershed transform with automatic marker selection) fails in the instance where the segmentation requires an unknown number of regions.

The watershed transform has various advantages: parameter independence, complete

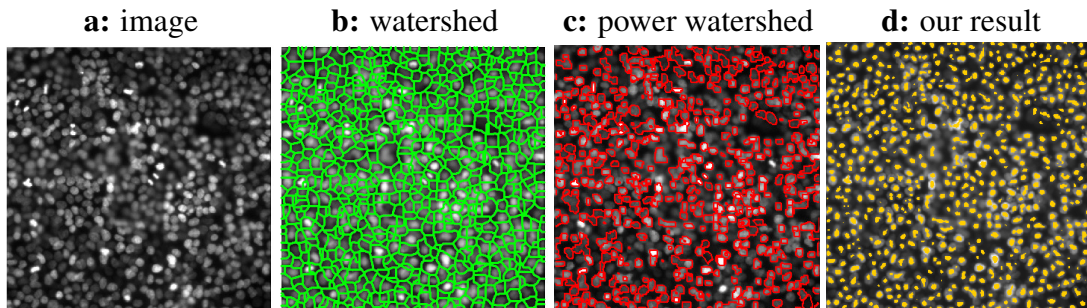


Figure 2.2: Motivation behind Markers. The image gradient is highly sensitive to local intensity fluctuations and gives rise to an elevated number of minima (b), i.e. results in severe oversegmentation. Latest watershed methods using markers to drive the segmentation, such as power watersheds (c), highly improve the results but fall short when dealing with these type of structures as low contrast dots are overseen. We are able to find all desired dots in a two way segmentation with the same set of parameters.

edge detection with good localization, detection invariance to lighting, robustness to presence of contrasting regions and accompanied high edge sensitivity, ability to handle gaps (e.g. in the absence of any edges between two markers, it finds the points equidistant between the two) and finally, computation can be efficiently implemented through flooding of the topographical surface. Nevertheless, the original flooding algorithm, in which pixels are individually labeled as the basins are enlarged, is inherently of local nature and hence can be easily disrupted by local intensity fluctuations. As previously mentioned, adding markers can overcome the oversegmentation problem, but it is generally used with interactive or semi-supervised segmentation. In order to deal with oversegmentation also in an automatic way, different levels of flooding are associated with a hierarchical segmentation, thus allowing the elimination of contours or combining regions of less importance.

The following sections show how the original local watershed segmentation method can be shifted from a pixel level to a more global region level through the construction of hierarchical structures [Mey05]. Both automatic and interactive segmentations benefit from the hierarchical structures, as a priori information can be integrated in a variety of ways by (i) controlling the rate of the flow in the automatic version or by (ii) using markers to construct the hierarchy in the semi-supervised case. The watersnakes method [NWB03]

also relies on a hierarchical watershed segmentation. The initial segmentation is used to improve the contours and is able to add priors on the contour smoothness that cannot be integrated with the other approaches. Finally, the watershed cuts graph framework [CBNC09] extends the definition of watersheds in terms of optimal minimum spanning forests, and provides a faster linear-time algorithm to find the optimal cut. The method can be used in conjunction with the hierarchies on region-adjacency graphs, instead of at the pixel level, to accelerate the computation even further. The construction of optimal spanning forest has since been used to unify other graph-based seeded image segmentation methods, namely graph cuts [BVZ01], random walker [Gra06b] and geodesics [XS07, CSB08], into a more generalized class of seeded watershed segmentations called *power watersheds* [CGNT09], illustrated in Fig. 2.2.

2.1 Morphological Segmentation

The original watershed based on flooding [Mey94] can be viewed as a local process that takes place at the pixel level. Almost thirty years after his original formulation, [Mey05] explores the inherent hierarchical structure of the flooding idea to shift the viewpoint to a more global region level. The author presents two main approaches in terms of floodings and in terms of levelings to construct better hierarchies and extract the desired contours.

If $f(x)$ gives the altitude of the relief at point x , the maximum erosion at $\epsilon(f)(x)$ is given by the maximum altitude difference encountered between x and its neighbors.

Definition (Topographic distance). *The topographic distance $L(x, y)$ between two points x and y is given by*

$$L(x, y) = \min_{\pi \in [x \rightsquigarrow y]} \sum_{i=2}^n \mu p_i \quad (2.1)$$

where $\pi = \langle x = p_1, p_2, \dots, p_n = y \rangle$ are all possible paths between x and y and $\mu(x) = f(x) - \epsilon(f)(x)$ is the slope of steepest descent.

The lines of steepest descent are given by the geodesics of this distance. Let the image be represented by a graph, in which image pixels correspond to graph nodes. The *regional*

minima $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K$ of the relief function f are K finite distinct connected sets of points of a constant value of f , denoted altitude $f(m_i)$ for each regional minimum \mathcal{M}_i , from which there is no ascending path to other points of strictly lower altitude. Constructing the catchment basins for the minima will intuitively be a shortest path problem to find the path of minimum weight between the markers \mathcal{M}_i and image points, alternatively.

Definition (Catchment Basin, Watershed Lines I). A catchment basin (CB_i) of a regional minimum \mathcal{M}_i of the set of pixels which are closer to \mathcal{M}_i than any other regional minimum for the topographical distance. The watershed line (WS) of the function f is then defined as the set of points not belonging to any catchment basins.

The most efficient implementation of watershed is based on the idea of flooding of a topographical surface, where the relief is flooded from sources placed at each regional minimum. In each region, lakes increase their level uniformly over the relief surface, until two lakes meet. The pixels where this happens are the ones at the same topographical distance from the two minima and hence are part of the watershed line.

Partitions and Hierarchies. Intuitively, finding the catchment basins will correspond to finding a segmentation of the image, i.e. a partition of image space \mathcal{X} into a family of connected regions $\Omega_1, \Omega_2, \dots, \Omega_K$ with the following properties:

- $\bigcup_{i=1}^K \Omega_i = \mathcal{X}$
- $\forall i \neq j : \Omega_i \cap \Omega_j = \emptyset$

The regions $\Omega_1, \Omega_2, \dots, \Omega_K$ are elements of $\mathcal{P}(X)$, the power set of \mathcal{X} containing all subsets of \mathcal{X} . The properties state that if two regions do not have an empty intersection, then they must be the same region and that the union of all regions equals the image space itself. Partitions can be finer or coarser, and this is measured by an order relation between them, with increasing level flooding. As regions merge together, multiple finer regions get included within larger coarser ones and further causing the partitions to get nested. The collection of the partitions at the different levels of refinement forms a hierarchy of

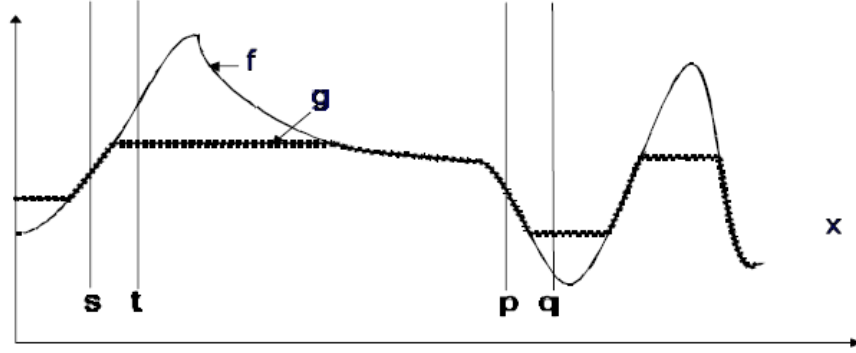


Figure 2.3: Levelings. The function g is a leveling of the function f , with $\lambda = 0$. Image taken from [Mey05] .

partitions, which can be formally described as building a classification tree with special properties:

Definition (Hierarchy). A hierarchy \mathcal{H} is a collection of subsets of $\mathcal{P}(X)$ together with an inclusion order relation, such that if $A, U, V \in \mathcal{H}$ then $A \subset U$ $A \subset V \implies U \subset V$ or $V \subset U$, and in addition the following axioms hold:

- If $A, B \in \mathcal{H}$ then $A \cap B \in \{A, B, \emptyset\}$ and
- $\forall A \in \mathcal{H}, \cup\{B \in \mathcal{H} | B \subset A; B \neq A\} = \{A, \emptyset\}$.

The different levels of flooding are characterized by a stratification index: an index function $\sigma : \mathcal{A} \rightarrow \mathbb{R}$ which is strictly increasing with the inclusion order and therefore can be used to extract partitions of a desired level of coarseness by thresholding the hierarchy.

To construct hierarchical structure, we need to add a notion of order and distance between the partitions within the hierarchy. The ultrametric distance is interesting because the triangle inequality is replaced by a stronger *ultrametric inequality* so that the sets of ultrametric balls of increasing radii λ produce a family of nested partitions of the image domain. The radii of the balls give the stratification index for the family of partitions which is strictly increasing with the inclusion order and the resulting structure is called a

stratified hierarchy of partitions. The hierarchy of the catchment basins associated to the family of floodings is in fact equivalent to the definition of an ultrametric distance.

Hierarchies based on Levelings. Leveling can be seen as preprocessing filters for the segmentation to enlarge homogeneous regions by symmetrically erasing peaks and flooding valleys. The definition can be understood in terms of quasi-flat zones, i.e. connected pixels with the same intensity value that satisfy the following symmetrical relation: $|f(x) - f(y)| \leq \lambda$ for all neighboring pixels x, y and maximal slope λ . A leveling is then given by:

Definition (Leveling). *A function g is a leveling of a function f if and only if for any neighboring pixels x and y we have $g(x) > g(y) + \lambda \implies f(x) \geq g(x)$ and $g(y) \geq f(y)$.*

Levelings can be generated starting from any marker function g . The result (Fig. 2.3) retains the initial characteristics of the marker function and the remaining contours will coincide with the function f simultaneously restoring boundary sharpness. The parameter λ equals the maximal slope of the quasi-flat zone, so that for $\lambda = 0$ the smooth zones become flat. Since levelings enlarge quasi-flat zones and enlarge or suppress minima and maxima without introducing new ones, either approach can be used to construct a hierarchy. If extrema are chosen as markers to construct the hierarchy, the merging of the regions can be controlled by the vanishing of the extrema between one level and the next. Alternatively, the quasi-flat zones of a family of increasing levelings itself forms a hierarchy. Quasi-flat zones can result for either large regions associated to homogenous zones, or very small ones where the gradient is high. The hierarchy is obtained by first constructing a fine partition without the transition zones and then merging the finer level regions to obtain coarser ones.

Hierarchies based on Floodings. Floodings are a special type of leveling in which the function g also has to satisfy $g \geq f$:

Definition (Flooding). *A function g is a flooding of a function f if and only if $g \geq f$ and for neighboring pixels x and y we have $g(x) > g(y) \implies g(x) = f(x)$.*

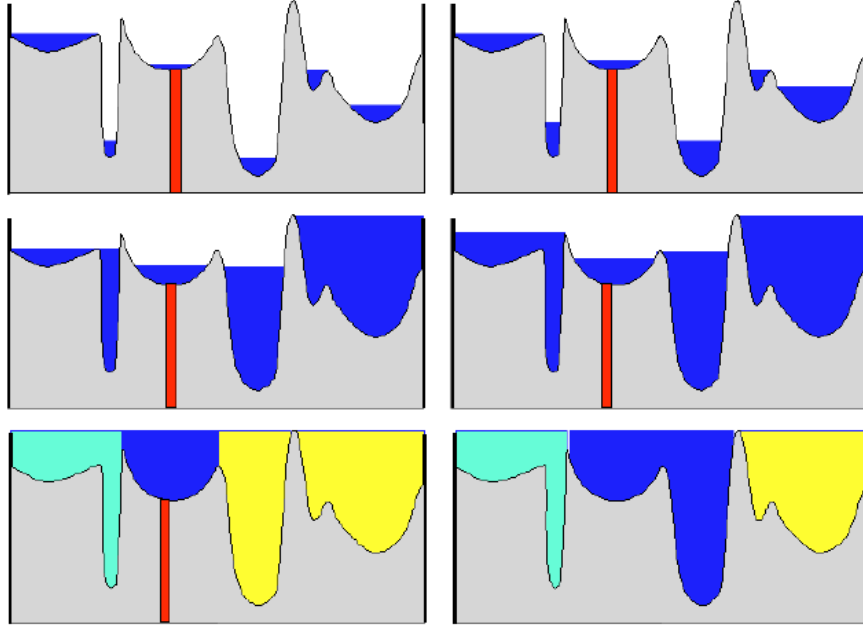


Figure 2.4: Tailored synchronous flooding. Four different levels are presented where red indicated that the minimum marker has been slowed down by a factor of 5. The resulting segmentation into 3 regions is shown together with the analogous segmentation in the case of no source being slowed down. Image taken from [Mey05].

Following the analogy of a flood, for each type of flooding increase, there is a corresponding hierarchy of catchment basins. The rate of the flooding can be controlled in a variety of ways: flooding can depend on height, area or volume of the regions; Fig. 2.4 illustrates an example of tailored flooding to favor some regions over others by reducing the rate of flow associated to specific minima, which for example can be defined as initial markers. These three approaches can be grouped by the concept of *fuzzy markers*. To each marker an associated value $0 \leq \lambda \leq 1$ can be associated to control the rate of the flooding at that particular source. Using fuzzy markers allows to close the gap between marker driven segmentation and unsupervised multiscale segmentation.

Segmenting with hierarchies. A hierarchical approach to the segmentation, based on the watershed and markers, allows to inject various types of information in the segmentation process, through the markers themselves or through adaptations of the hierarchy depending on the specific application. The hierarchy construction based on floodings

can be done by one flooding of the gradient image and hence is very fast. The rate of the flooding can be adjusted to account for different properties (e.g., size, contrast or a balance between the two) also in different parts of the image, making the construction versatile for a variety of applications. This method fails in the presence of thin regions as the gradient image will wash out the information of the original structure instead of producing a minimal region. In these cases, a hierarchy based on levelings is better. Thin structures are detected as flat zones, so the hierarchies constructed on the quasi-flat zones of levelings will capture their fine details. Finally, the original simpler hierarchy construction based on merging regions together from an initial fine partition can be used in many different scenarios, since a variety of similarity measures can be used and merging criteria can even change with the coarsening of the partition.

2.2 Energy-Driven Watershed Segmentation: Watersnakes

Although the selection of the regional minima from markers allows for better partitioning and an approach to overcome oversegmentation, it does not deal with the smoothness of the watershed lines. Within the original watershed setup, no a priori information can be given to the system to improve on the boundary smoothness of the watershed lines. With the introduction of watersnakes, [NWB03] show how to integrate these priors by representing watershed segmentation as the result of the minimization of an energy function.

Let $L_i(x)$ is the corresponding topographical distance transform $L_i(x) = L(x, \mathcal{M}_i) = \inf_{y \in \mathcal{M}_i} L(x, y)$ between point x and the regional minimum \mathcal{M}_i introduced in Eqn. 2.1 (recall that the topographic distance is the spatial distance between two points weighted by the gradient norm of f , which in a continuous domain can be expressed as $L(x, y) = \inf_{\gamma \in [x \rightsquigarrow y]} |\nabla f(\gamma(s))|$). We can then redefine:

Definition (Catchment basin, Watershed Lines II). *A catchment basin (CB_i) of a regional minimum \mathcal{M}_i of the set of pixels which are closer to \mathcal{M}_i than any other regional minimum*

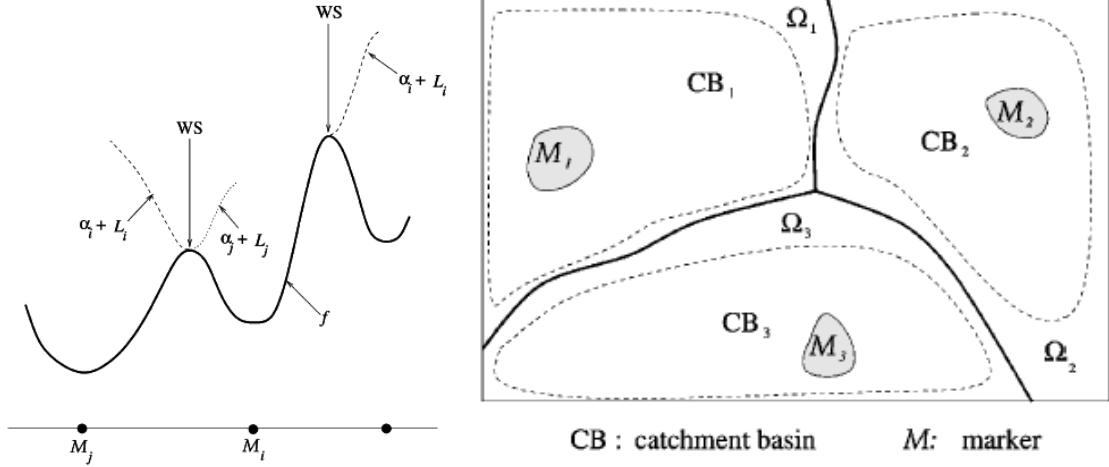


Figure 2.5: Watershed lines, catchment basins and markers, illustrated in 1D and 2D (image taken from [NWB03]). The 1D (left) illustrates the definition in terms of the altitude α_i of each regional minimum \mathcal{M}_i and of the corresponding topological distance transform L_i . It can be noted that within each catchment basin CB_i , $f(m_i) + L_i(x)$ equals $f(x)$. The 2D depiction (right) shows the partition of the image into regions $\Omega_1, \dots, \Omega_K$, each containing one and only one catchment basin CB_i derived from a regional minimum \mathcal{M}_i .

for the topographical distance $L(x, y)$:

$$CB_i = \{x \in \mathcal{X} | \forall j \neq i, 1 \leq j \leq K : f(m_i) + L_i(x) \leq f(m_j) + L_j(x)\}. \quad (2.2)$$

The watershed line (WS) of the function f is then defined as the set of points not belonging to any catchment basins:

$$WS(f) = \mathcal{X} \setminus \bigcup_i CB(\mathcal{M}_i). \quad (2.3)$$

As shown in the 1D illustration in 2.5, within the catchment basin CB_i , $f(m_i) + L_i(x)$ equals $f(x)$, and beyond the catchment basin, it becomes the reflection of $f(x)$ with respect to the horizontal line passing through the watershed point.

Criterion (Watershed segmentation). A partition of the space \mathcal{X} into regions $\Omega_1, \dots, \Omega_K$, that additionally satisfies $Int(\Omega_i) = \Omega_i$, where $Int(R)$ denotes the interior of the set R

and the bar it's closure. If the following property is satisfied:

$$\forall x \in \Omega_i = \alpha_i + L_i(x) = \min_{1 \leq j \leq K} \{\alpha_j + L_j(x)\}. \quad (2.4)$$

where α_i corresponds to $f(m_i)$, the altitude of the function f on the corresponding regional minimum, and $L_i(x)$ is the corresponding topographical distance transform. The partition is a watershed segmentation for the function f .

The additional partition property ensures that each boundary segment separates at least two different regions. Each watershed partition thus constructed contains one and only one catchment basin and the borders of the regions lie within possibly thick watershed lines as shown in Fig. 2.5. Interestingly, the function $f(x)$ can be fully reconstructed from the distances to the regional minima, i.e. $f(x) = \min_{1 \leq i \leq K} \{f(m_i) + L_i(x)\}$

The main idea of the paper is based on the following theorem, which connects the definition of watershed (Eqn. 2.4) with a possible energy function:

Theorem. : A partition $\Omega_1, \Omega_2, \dots, \Omega_K$ minimizes the following function:

$$E(\Omega_1, \Omega_2, \dots, \Omega_K) = \sum_{i=1}^K \iint_{\Omega_i} \{\alpha_i + L_i(x)\} dx \quad (2.5)$$

if and only if it is a watershed segmentation.

With this definition of watershed, it is possible to add a boundary length to the energy function, hence introducing a smoothness prior to the region boundary:

Criterion (Watersnakes segmentation). The watersnakes segmentation is obtained by minimizing the following function:

$$E(\Omega_1, \Omega_2, \dots, \Omega_K) = \sum_{i=1}^K \left(\iint_{\Omega_i} \{\alpha_i + L_i(x)\} dx + \beta \int_{\partial\Omega_i} dS \right) \quad (2.6)$$

where β is a weighting coefficient.

This energy function falls within the region-based minimization approaches. In order to better understand the connection to energy-minimization methods we compare Eqn. 2.6

with the energy term used in the region competition method [?]. In [?], the first term is replaced by $-\log P(I_x|a_i)dx$, the probability density of the measured intensity of pixel x given the assumption that $x \in \Omega_i$ and a_i is a parameter for this distribution. In both methods, this term measures homogeneity within the region Ω_i , but the lack of additional parameters in the Watersnakes formulation allows for a much simpler minimization procedure eliminating parameter upgrading steps. Each of the regions $\Omega_1, \Omega_2, \dots, \Omega_K$ in the partition corresponds to a unique region in the original watershed criterion. The added boundary term allows smoothing of the boundaries of these regions.

Proposed Solution. There are two methods to implement the watersnakes minimization energy problem (Eqn. 2.6) in the discrete domain. The first, based on a region growing algorithm and a second, based on energy discretization. Although very intuitive, the region growing method has a few shortcomings. If implemented with level sets [2.6], it is only efficient in the presence of two markers. On the other hand, if implemented by assigning marker labels to the boundary pixels as the region is growing at some speed given by a compression force determined by the local relief function geometry, the approach smoothes boundaries in many practical applications and does not guarantee a solution.

[NWB03] present a method based on energy discretization. A local contour length estimate is needed in order to calculate by how much the region perimeter increases when a pixel is added to the region's border using only local information. The perimeter of Ω_i is estimated using a Chamfer neighborhood:

$$\int_{\Omega_i} dS = \sum_{x \in \Omega_i} [w_4 n_4(x) + w_8 n_8(x) + w_k n_k(x)], \quad (2.7)$$

where w 's are the weighting coefficients (estimated empirically) and $n_4(x), n_8(x), n_k(x)$ are the number of pixels in the four, eight and 'knight-move' connected neighborhood of x but outside of Ω_i .

In order to compute the partition of the image on a discrete grid, the continuous watersnakes energy function given in Eqn. 2.6 now becomes:

$$E(\Omega_1, \Omega_2, \dots, \Omega_K) = \sum_{i=1}^K \sum_{x \in \Omega_i} \{\alpha_i + L_i(x) + \beta(w_4 n_4(x) + w_8 n_8(x) + w_k n_k(x))\} \quad (2.8)$$

where we have substituted the local contour estimate found in Eqn. 2.7 for the smoothing term, which is non-zero only near the region boundary. The algorithm is divided into a two stage process to minimize Eqn. 2.8: a first stage for the original watershed segmentation (shown in algorithm box 1) and a second stage (algorithm box 2) in order to integrate the watersnakes smoothing step.

Watersnakes shows an improvement over region-based energy minimization methods; even if edges between two markers are faint, the resulting contours will still lie between the markers, since the latter are given in the initial segmentation. Watersnakes are therefore capable of filling out the gaps between broken edges as they try to minimize the distance to both regional minima. In addition, as the watershed line always corresponds to the most significant edges between the markers, watersnakes improve the original watershed lines to smoothen out the contour, without converging to weaker edges and the process is entirely parameter independent.

Algorithm 1 Stage 1: Watershed Segmentation

1. Compute K distance transforms $L_i(x)$
 2. Initialize the regions Ω_i from the regional minima \mathcal{M}_i
 3. From unassigned pixels on the outer boundary of the regions, select one with minimal value $\alpha_i + L_i(x)$, where i is the label of the adjacent region, and assign the selected pixel to the region Ω_i .
 4. Iterate Step 3 until all pixels are assigned.
-

2.3 Watersheds on Graphs: Watershed Cuts

The intuition behind the drop of water principle of the original watershed algorithm is formalized in the work of [CBNC09]. Let the image space \mathcal{X} be represented by a graph $G(V, E)$, where vertices V correspond to pixels in an image and edges E to weights between the pixels. Instead of assigning altitudes of the relief map using values at the nodes, the values associated with the edges are used. The altitude f then becomes a map $f \in (F) : E \rightarrow Z$, so that for an edge u between pixels x and y , $f(u)$ is the altitude of

Algorithm 2 Stage 2: Watersnakes Iterated Smoothing

1. Start with a low-resolution level by sub-sampling the label image as well as the K distance transforms.

for each finer partition in the hierarchy, **do**

2. For every boundary pixel compute the stability according to the change of energy:

$$\Delta E(x, i \rightarrow j) = (\alpha_i - \alpha_j) + (L_j(x) - L_i(x)) \\ + 2\beta \left\{ w_4[n_4(x, i) - n_4(x, j)] + w_8[n_8(x, i) - n_8(x, j)] + w_k[n_k(x, i) - n_k(x, j)] \right\}.$$

3. Select the boundary pixel with lowest ΔE and perform the reassignment when this value is negative.

4. Recompute the stability of the pixel $\Delta E(x, i \rightarrow j)$.

5. Iterate steps 3 and 4 until no further reassignment is possible.

end for

the function F . Each regional minima \mathcal{M}_i forms then a subgraph X of G and is called the minimum with respect to the function f as it satisfies:

- X is connected
- k is the altitude of any edge of X and
- the altitude of any edge adjacent to X is strictly greater than k .

The notion of flooding from initial sources in order to enlarge the catchment basins until the watershed lines are reached, can be translated to finding the *extension* of the subgraphs given by the minima. To appreciate the equivalence of the catchment basins and the drop of water principle on the lines separating them, the their basic definitions are reformulated in terms of graph.

Definition (Watershed Cut, Watershed Lines III). *Let S be a subgraph of E and $\mathcal{M}(f)$ denote the union of the vertex sets and edge sets of all the minima of f . S satisfies the ‘drop of water’ principle, and is therefore defined as a watershed cut of f , if given an extension S of $\mathcal{M}(f)$, for any $u = \{x_0, y_0\} \in S$, there exist two descending paths $\pi_1 = \langle x_0, \dots, x_n \rangle$ and $\pi_2 = \langle y_0, \dots, y_m \rangle$ in S such that x_n and y_m are vertices of two distinct minima of f and $f(u) \geq f(\{x_0, x_1\})$ whenever π_1 is not trivial, and similarly for π_2 .*

From this definition it follows that if S is a watershed of f , then S is also a cut for $\mathcal{M}(f)$. The catchment basins intuitively ‘extend’ these minima. In graph terminology, the seeds or minima are a subgraph X whose extensions become the catchment basins. A subgraph Y is an *extension* of X if both X and Y are non-empty subgraphs of G , $X \subset Y$ and if any connected component of Y contains exactly one connected component of X . When the extension has reached its maximum size, a graph cut is obtained:

Definition (Basin Cut, Catchment Basins III). *Let S be a subgraph of E , S is a basin cut of the function f if, from each point of the vertex set V to the minima $\mathcal{M}(f)$ there exist a path of steepest descent for f in the graph induced by S . Any component of S is a catchment basin of f .*

Finally, one last definition is needed before presenting the proof of optimality and consistency of watersheds on graphs. Intuitively, a forest relative to a subgraph X is an extension Y of X such that any cycle in Y is also a cycle in X . Y is called a *spanning forest relative to X* if its vertex set $V(Y)$ equals the vertex set V of the graph itself. A relative minimum spanning forest is then defined as:

Definition (Relative minimum spanning forest). *Let X, Y be subgraphs of G . Y is the minimum spanning forest (MSF) relative to X for the function f on G if Y is a spanning forest relative to X and if the sum of its edges weights is less than or equal to the weight of any other spanning forest relative to X . In this case, Y is also a relative MSF.*

Consistency and optimality of watersheds. First, [CBNC09] prove that watersheds can be equally defined in terms of the catchment basins, where the construction of the basins can be viewed as (i) a shortest-path problem, to find the optimal path between the regional minimum \mathcal{M}_i , i.e. the *marker*, and an image point or (ii) in terms of the dividing lines between these basins (through the drop of water principle):

Theorem (Consistency). *Let S be a subset of the edges E . S is a basin cut if and only if S is a watershed cut of F .*

Second, [CBNC09] establish the optimality of the watershed cut, i.e. by showing the equivalence of a cut that satisfies the ‘drop of water’ principle with the cuts induced by the minimum spanning forests (MSF) relative to the minima of a map:

Theorem (Optimality). *Let S be a subset of the edges E . The set S is an MSF-cut for the minima of the relief function $\mathcal{M}(F)$ if and only if S is a watershed cut of F .*

In other words, the graph X is a MSF relative to the minima of the relief function $\mathcal{M}(F)$ if and only if, for any $x \in V$ there exist a path in X from x to $\mathcal{M}(F)$ which is also a path of steepest descent for F . This result is interesting because, by proving that a relative MSF can be constructed from any minimum spanning tree, it reduces watershed computation to finding a minimum spanning tree in the graph. Classical algorithms such as Kruskal’s or Prim’s can be used to find the segmentation in quasi-linear complexity.

Proposed solution. In order to compute the watershed segmentation, [CBNC09] go beyond computation of a minimum spanning tree by introducing an algorithm that runs in linear time with respect to the number of edges E of the original graph. This is achieved by computing the flow cut of F in a series of depth-first search and breadth-first search along the the paths of steepest descent, to iteratively extract streams (i.e. a set of vertices L such that for any two points x, y on L there exist a path of steepest descent between these two points with respect to the map F). Given a graph $G(V, E)$ with an altitude function f , the algorithm computes a flow map ψ of f by extracting and labeling streams. It is worth noting that this algorithm does not take into account control of the watershed lines on plateau regions, which is left to be taken care of in a preprocessing step. But it does always guarantee convergence, even on plateaus in the graph (namely, connected subgraphs with constant altitude), since all nodes are eventually explored.

Depending on the application, the altitude f can be defined in various ways. Common choices would be the image gradient $f(x, y) = |I(x) - I(y)|$ where $u \in E$ is the edge between pixels x and y . The classical watershed problem, defines the edge weights between pixels x and y by the map $f(x, y) = \min\{I(x), I(y)\}$ where $I(x)$ is the grey-scale intensity value of the image at point x . Segmentation into k regions can be achieved by

starting with a function f that contains exactly k minima, which is achieved through a preprocessing filtering step that progressively fills in the gaps that are considered unimportant. As mentioned in the paper by [Mey05], the choice can depend on various factors; here the importance is given based on the area of each candidate region, i.e. the number of vertices of the subgraph that represents this region.

2.4 Generalization of Watersheds: Power Watersheds

Thanks to the consistency and optimality proven in the previous section by [CBNC09], watershed lines can be computed in quasi-linear time by finding the optimal spanning forests relative to the minima of the relief map. The work presented in [CGNT09] extends the class of watersheds on graphs to a new family of seeded watersheds, thus providing a general energy-minimization approach for computing watershed lines while linking watershed algorithms to commonly used seeded segmentation methods such as graph cuts [BVZ01], random walker [Gra06b] and shortest path optimization [XS07, CSB08].

Originally for interactive image segmentation, graph cuts produce a labeling by finding a minimum cut via a maximum flow computation between foreground and background seeds that are treated as sink/sources for a max/min flow computation. It is biased towards a ‘small cut’, yielding boundaries that cut through thin protruded regions. To avoid leaking and the shrinking bias present in graph cuts, the random walker algorithm, formulated on a weighted graph, determines the labels for the unseeded nodes by assigning pixels to seed for which it is most likely to send a random walker, viewable also as the seeds for which there is a minimum diffusion distance, as a semi-supervised transduction learning algorithm [DAK⁺08] or as an interactive version of Normalized Cuts [SM00]. In this framework, edge weights are treated as probabilities that the particles travel first to foreground or background seeds. These probabilities can be computed analytically without any random walk simulation, but the quality of the segmentation boundary depends more strongly than in the graph cuts scenario on original seed location. Shortest-paths (or

geodesics) assign pixels a foreground label if there is a shortest path between the pixels to another foreground seed rather than to any background seed; paths are weighted similarly to graph cuts and to random walker approaches; speed and shrinking bias advantage; stronger dependence on seed locations when compared to random walker; leakages more likely through weak boundaries (since a good single path is sufficient for connectivity).

The first segmentation algorithm linking graph cuts and random walker was presented in [SG07]. The image is viewed as a weighted graph $G = (V, E)$, with vertices $v \in V$ and edges $e \in E$, with non-negative, real weights w_{ij} on edges e_{ij} linking vertices v_i and v_j . The weights set are set as standard gaussian weighting function $w_{ij} = \exp(-\beta(I_i - I_j)^2)$. Given foreground F and background B seeds, a generalized segmentation is given by:

$$\min_x \sum_{e_{ij} \in E} (w_{ij}|x_i - x_j|)^q + \sum_{v_i} (w_{F_i}x_i)^q + \sum_{v_i} (w_{B_i}|x_i - 1|)^q \quad (2.9)$$

such that $x(F) = 1$, $x(B) = 0$ and segmentation output $s_i = 1$ if $x_i \geq 1/2$, zero otherwise. w_{B_i} and w_{F_i} are the unary weights penalizing foreground and background affinity at node v_i . It has been showed in [SG07] that when $q = 1$ gives graph cuts, when $q = 2$ gives random walker and as $q \rightarrow \infty$ it gives shortest paths.

The new segmentation algorithm which includes power watersheds [CGNT09] arises as a generalization of Eqn. 2.9. Recall from the previous section that regions of the watershed can be seen as connected components of an extension relative to the regional minima, separated by a set of edges by the ‘drop of water principle’. Watershed lines become a cut relative to the regional minima of the weight function, which satisfy the drop of water principle and can be computed by viewing them on a graph induced by a shortest-path forest rooted in the minima.

Generalizing the theory of [SG07] to include also the optimal spanning forest algorithm, therefore provides the unification of watersheds with the energy-minimization methods of graph cuts, random walker and shortest-paths. This new family of segmentation algorithms is obtained by separating the exponent between the weights and adding an additional parameter on the weights and variables in the generalized energy function.

Criterion. (*Power Watersheds*). A new family of segmentation algorithms is obtained by minimizing the following energy criterion:

$$\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q + \sum_{v_i} w_{F_i}^p x_i^q + \sum_{v_i} w_{B_i}^p |x_i - 1|^q \quad (2.10)$$

Letting $p \rightarrow \infty$ and varying q yields ‘power watersheds’, a family of previously unexplored segmentation models.

When p is a small finite value can interpret $q = 1, 2$ as before; when p and q converge together towards infinite solution can be computed with shortest path. When $q = 1$ and $p \rightarrow \infty$ the minimum is given by a maximum spanning forest algorithm, i.e. as the power of the weights increases to infinity, the graph cuts algorithm produces a segmentation corresponding to the maximum spanning forest. Power watersheds arise by letting $p \rightarrow \infty$ and varying q . By associating $p = \beta$, above some value of the parameter beta, the expensive max-flow computation can be replaced with an efficient maximal spanning forest computation, regardless of q .

Proposed solution. The algorithm proposed to minimize the energy term for any value of q when $p \rightarrow \infty$ is solved by a maximum spanning forest starting from the initial seeds and treating the plateau regions separately (in order to retain binary labeling). The algorithm is based on Kruskal’s or Prim’s algorithm to compute a maximum spanning tree but instead of a tree a forest is computed and a q -cut is performed on the plateaus. Additionally, unary terms are incorporated into the usual watershed segmentation algorithm as binary terms connected to phantom seeds. To retain the speed of the MSF algorithm, $q = 2$ in the experiments. Optimality of the power watershed is achieved if seeds are the only maxima in the image. Hence, it is achieved by applying a geodesic reconstruction on the gradients before applying the algorithm. Plateau regions (where weights are equal) are dealt with separately. Given the usual fixed size of plateaus, the algorithm runs in quasi-linear time making it comparable to standard watershed algorithms. Thanks to this approach, power watersheds can be easy to implement and resolve the issues of leakages and degeneracy of the solution on the plateaus of the original watershed framework.

Chapter 3

Robust Segmentation by Cutting across A Stack of Gamma Transformed Images

3.1 Introduction

Hair cells of the inner ear transduce mechanical signals into electrical signals [VKC07]. Each hair bundle is composed of tens of stereocilia organized in an organ-pipe-like formation of increasing height (Fig. 3.1). Automatic segmentation of these stereocilia in their fluorescent images is vital for medical research on hearing. Segmentation of such medical images appears to be governed by global intensity levels, yet imaging noise and local intensity fluctuation present considerable challenges (illustrated in Fig. 3.2).

We present a spectral-graph partitioning approach that is robust to local intensity fluctuation and can extract several regions of interest without any user initialization. We encode the impact of high and low intensities, which we will refer to as peaks and valleys, in local pairwise grouping cues that encourage peak regions to stay together and valley regions to divide apart. It is the job of global integration to decide where region boundaries should be.

Our idea is that regions of an image appear stable with respect to the gamma transformation of the image, while cues in each gamma transformed version reflect an ever

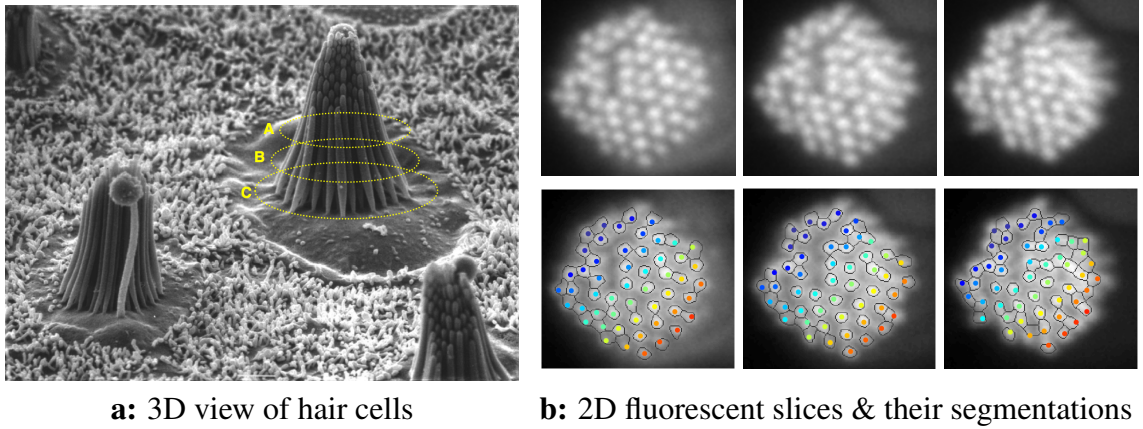
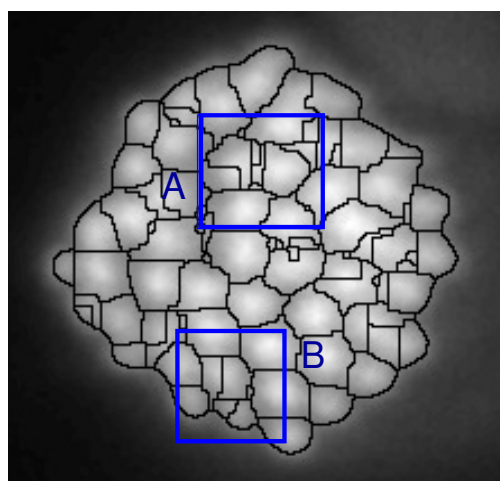


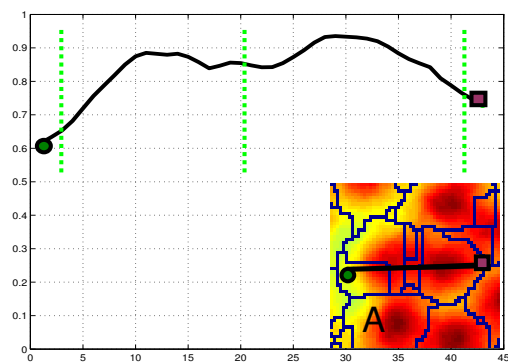
Figure 3.1: Stereocilia segmentation. **a)** Hair cells are composed of tens of stereocilia organized in an organ-pipe-like formation of increasing height. **b)** Fluorescent images (Row 1) and their segmentations (our results, Row 2) at multiple heights show the cross sections (e.g. A,B,C in **a**) of individual stereocilia (marked by colored dots).

changing balance between peaks and valleys, as peaks keep shrinking and valleys enlarging with an increasing gamma. The desired segmentation must be the global consensus of local cues from a stack of these gamma images.

Illustrated in Fig. 3.3, given an image I , we first create several gamma transformed versions: $I_n = I^{\gamma_n}$. For each I_n , we define two complementary local pairwise grouping cues: a short-range attraction between nearby pixels with similar intensities and a long-range repulsion between distant pixels with similar intensities but separated by valleys. The former occurs most likely for pixels belonging to the same stereocilium and the latter for pixels belonging to adjacent stereocilia. Large repulsion demands single boundaries to occur somewhere between two distant pixels, whereas large attraction discourages the formation of boundaries between two nearby pixels, preventing the oversegmentation problems in Fig. 3.2. We establish rough local pixel alignment between gamma images and project cues derived from each I_n to the original image I through pixel correspondences. We seek a graph cut across the cue stack which respects both attraction and repulsion, producing segmentation X_k for the original image I , with a granularity determined by the number of eigenvectors k .



watershed segmentation



A: intensity fluctuation at boundaries

B: intensity fluctuation inside regions

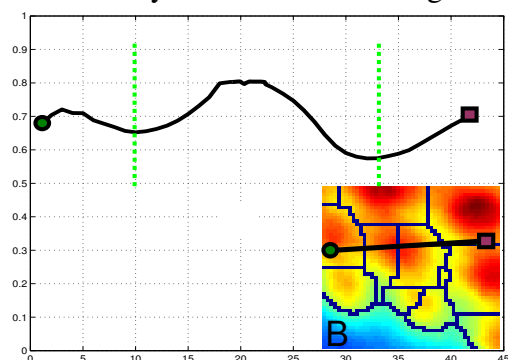


Figure 3.2: Local intensity fluctuation presents considerable challenges in medical image segmentation. **A)** Fluctuation at boundaries weakens the separation between two intensity peaks. **B)** Fluctuation inside regions tends to break up an otherwise well defined intensity peak. Both cases cause oversegmentations in watershed approaches. The solid black line plots the 1D intensity profile along the line connecting the two pixels in the inset, which shows the image in a marked window on the left. The dotted green lines mark the desired boundaries between intensity peaks.

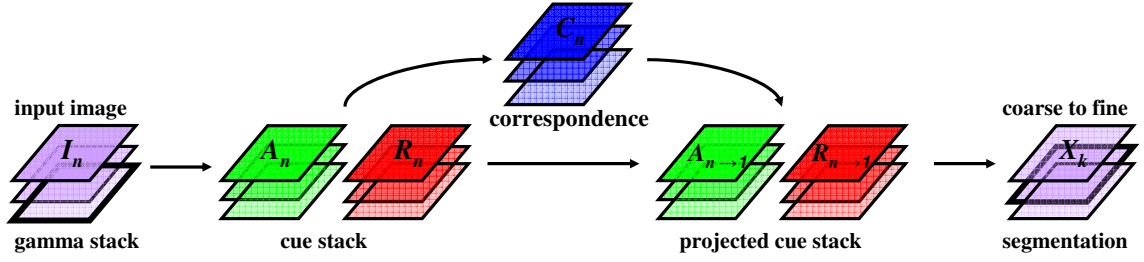


Figure 3.3: Method Overview. Given an image, we build a stack of its gamma transformed versions, i.e., $I_n = I^{\gamma_n}$. For each gamma image I_n , we derive pairwise attraction A_n and repulsion R_n between pixels. We compute pixel correspondences C_n between adjacent gamma layers, and project cues at each layer to the reference layer I_1 : $A_{n \rightarrow 1}$ and $R_{n \rightarrow 1}$. Cutting across the aligned cue stack produces segmentation X_k that is invariant to gamma transformations, k indicating the granularity of segmentation.

3.2 Constrained Cuts with Attraction and Repulsion

We formulate the segmentation in a spectral graph framework. We collect pairwise cues and seek the solution that optimizes a global criterion. We consider pairwise cues of three kinds: attraction A , repulsion R , and partial grouping constraints U . These cues have been studied separately in [YS04, YS01]. We combine them for the first time in a single framework.

3.2.1 Graph Representation

In spectral graph methods, an image I is represented by a weighted graph $G(V, E, W)$, where V denotes the set of nodes, E the set of edges connecting the nodes, and W the weights attached to edges. A pixel then becomes a node in the graph, each pairwise grouping cue becomes a weight between two nodes, and image segmentation becomes a graph node partitioning problem: We seek k partitions of node set V such that $V = \cup_{l=1}^k V_l$ and $V_i \cap V_j = \emptyset, \forall i \neq j$.

3.2.2 Criterion with Attraction and Repulsion

A good segmentation should have strong within-group attraction and between-group repulsion, and weak between-group attraction and within-group repulsion.

Characterizing this intuition with *linkratio* allows us to achieve both objectives simultaneously [YS03]. *linkratio* L of two node sets (P, Q) measures the fraction of connections from P to Q among all the connections P has:

$$\text{linkratio} \quad L(P, Q; W) = \frac{C(P, Q; W)}{C(P, V; W)} \quad (3.1)$$

$$\text{connections} \quad C(P, Q; W) = \sum_{i \in P, j \in Q} W(i, j) \quad (3.2)$$

In particular, we have $L(P, P; W) + L(P, V \setminus P; W) = 1$, i.e. maximizing a within-group *linkratio* is equivalent to minimizing its between-group *linkratio*.

We seek to maximize linkratios from within-group attraction and between-group repulsion, combined linearly according to their total degree of connections:

$$\max \quad \varepsilon = \sum_{l=1}^k \alpha L(V_l, V_l; A) + (1 - \alpha) L(V_l, V \setminus V_l; R) \quad (3.3)$$

$$\text{where} \quad \alpha = \frac{C(V_l, V; A)}{C(V_l, V; A) + C(V_l, V; R)} \quad (3.4)$$

α measures the total attraction, and $1 - \alpha$ the total repulsion.

3.2.3 Partial Grouping Constraints

We represent the partitioning by partition indicator $X = [X_1, \dots, X_k]$ where X_l is an $N \times 1$ binary indicator for partition V_l , $X_l(i) = 1$ if pixel $i \in V_l$, and 0 otherwise, $l = 1, \dots, k$. N is the number of pixels in the image.

We consider partial grouping constraints which require pixels a and b to belong in the same region, i.e. $X(a) = X(b)$. The collection of c such constraints result in $U^T X = 0$, where U is an $N \times c$ matrix, and each column of U has only two non-zero numbers, $+1$ and -1 .

3.2.4 Optimal Solution

Our criterion ε with pairwise attraction A and pairwise repulsion R , subject to grouping constraints U can be written in a compact matrix form:

$$\text{maximize} \quad \varepsilon(X) = \sum_{l=1}^k \frac{X_l^T W X_l}{X_l^T D X_l} \quad (3.5)$$

$$\text{subject to} \quad X \in \{0, 1\}^{N \times k}, X 1_k = 1_N \quad (3.6)$$

$$U^T X = 0 \quad (3.7)$$

$$\text{where} \quad W = A - R + D_R \quad (3.8)$$

$$D = D_A + D_R \quad (3.9)$$

where 1_n denotes the $n \times 1$ vector of all 1's, and $D_W = \text{Diag}(W 1_N)$ denotes the $N \times N$ degree matrix of weights W , with the diagonal containing the total degree of connections for each node.

Relaxing the binary constraints, we can solve this optimization problem [YS04] with the eigenvectors of $HD^{-1}WH$, where $H = I - D^{-1}U(U^T D^{-1}U)^{-1}U^T$, and then discretize the eigenvectors to obtain the final segmentation [YS03].

3.3 Pairwise Grouping Cues from Image Intensities

The success of global integration depends on the local cues that feed into it. We define a short-range attraction that pulls pixels towards region centers, a long-range repulsion that pushes pixels away from region boundaries, and a partial grouping constraints that force peripheral background pixels to belong together. With pixel correspondence between gamma images, we obtain a cue stack for the original image.

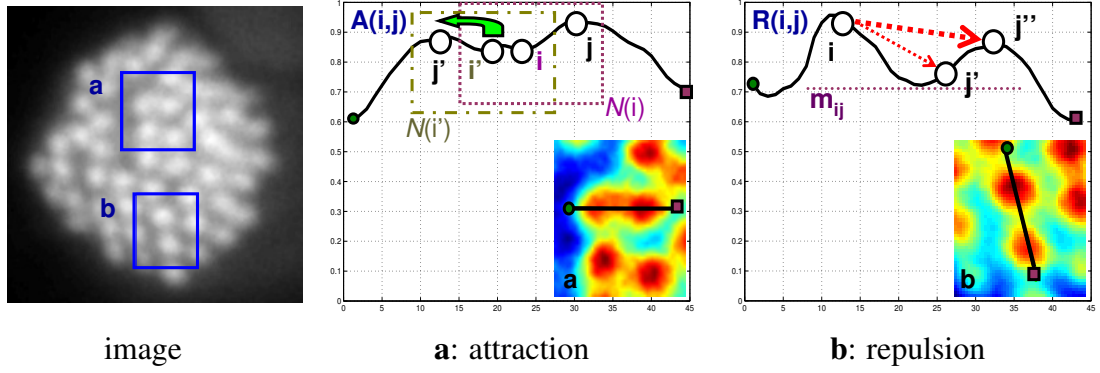


Figure 3.4: Pairwise attraction and repulsion. **a)** Our attraction is adaptive to the local intensity range within each neighborhood $\mathcal{N}(i)$, so that $A(i', j') \approx A(i, j)$, enhancing the discrimination of two nearby similar peaks. **b)** Our repulsion is strongest for nearby peaks and gets reduced as two pixels approach the inbetween valley: $R(i, j'') > R(i, j')$. m_{ij} is the minimal intensity level between pixels i and j .

3.3.1 Short-Range Attraction within Individual Peaks

Attraction $A(i, j)$ between pixels i and j encodes local intensity similarity. The straightforward definition

$$A(i, j) = e^{-\frac{|I_i - I_j|^2}{2\sigma_a^2}} \quad (3.10)$$

requires fine parameter tuning and tends to merge nearby peaks of similar intensities. We introduce a new definition that is asymmetrical between two pixels and acts to pull pixels towards intensity peaks.

For pixels i and j , $A(i, j)$ is inversely proportional to the the maximal intensity difference M_{ij} between i and any pixel on the line ij , with sensitivity regulated by local intensity range $\delta(i)$ in i 's neighborhood $\mathcal{N}(i)$:

$$A(i, j) = e^{-\frac{\max_{t \in \text{line}(i, j)} |I_i - I_t|^2}{2\delta_i^2 \cdot \sigma_a^2}} \quad (3.11)$$

$$\delta_i = \max_{t \in \mathcal{N}(i)} I_t - \min_{t \in \mathcal{N}(i)} I_t \quad (3.12)$$

We choose $\mathcal{N}(i)$ to be slightly larger than a stereocilium so that δ_i is estimated between the peak and surrounding valleys (Fig. 3.4a). With adaptive scaling by local intensity range

$\delta_i, A(i, j)$ effectively enhances attraction within weak peaks and allows a single parameter setting for σ_a to work on a variety of images.

3.3.2 Long-Range Repulsion between Peaks

Adjacent peaks provide a strong cue as to where the boundaries should lie. This cue is encoded by long-range repulsion. Intuitively, two pixels of similar intensity should belong to different peaks if they are separated by a valley. We define repulsion $R(i, j)$ between pixels i and j to be proportional to the difference with the minimal intensity m_{ij} encountered on the line ij :

$$R(i, j) = 1 - e^{-\frac{\min(|I_i - m_{ij}|, |I_j - m_{ij}|)}{\sigma_r}} \quad (3.13)$$

$$m_{ij} = \min_{t \in \text{line}(i, j)} I_t. \quad (3.14)$$

The farther away the pixels are from the valley, the larger the intensity difference with the minimum, and the larger the repulsion (Fig. 3.4b).

3.3.3 Pixel Correspondence and Cue Projection

With each gamma transformation, while peaks remain peaks and valleys remain valleys, their regions of influence change: Peaks shrink and valleys expand; pixels belonging to one peak region could become part of the background. Local grouping cues derived from gamma images consequently do not agree with each other. We establish rough pixel correspondence and project cues on individual gamma image back to the original image.

Let $A_n(i, j)$ be the affinity (i.e. attraction) between pixels i and j at gamma image I_n . We follow the approach in [Yu05] by computing the corresponding pixel location $C_n(i)$ as the center of mass of i 's affinity field and composing them recursively to obtained aligned

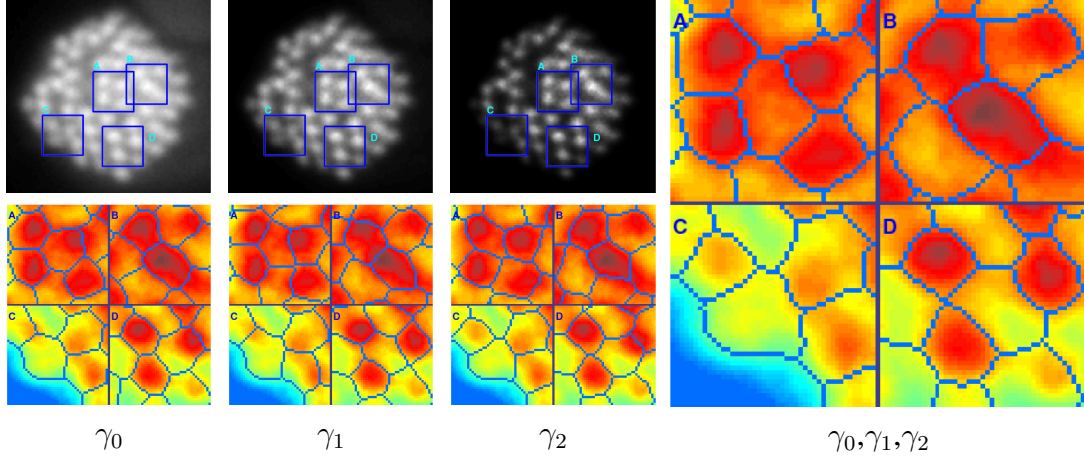


Figure 3.5: Better segmentation is obtained by cutting across the gamma stack instead of a single gamma image. Left shows images and segmentations based on individual γ 's in 4 marked windows. Right shows the segmentations based on all 3 γ 's.

cue stack:

$$A_{n \rightarrow 1}(i, j) = A_n(C_n(i), C_n(j)) \quad (3.15)$$

$$R_{n \rightarrow 1}(i, j) = R_n(C_n(i), C_n(j)) \quad (3.16)$$

$$C_n(i) = \sum_{j \in N(i)} A_n(i, j) C_{n-1}(j) \quad (3.17)$$

where $C_1(i)$ is pixel i 's location in the original image I .

Cutting across the aligned cue stack for the original image is equivalent to cutting a single graph with the following attraction and repulsion:

$$A = \sum_n D_{A,n}^{-1} A_{n \rightarrow 1} + A_{n \rightarrow 1} D_{A,n}^{-1} \quad (3.18)$$

$$R = \sum_n D_{R,n}^{-1} R_{n \rightarrow 1} + R_{n \rightarrow 1} D_{R,n}^{-1}. \quad (3.19)$$

where $D_{A,n}$ and $D_{R,n}$ are the degree matrices for A_n and R_n respectively.

3.3.4 Partial Grouping Constraints

We obtain a crude background mask by intensity thresholding on the original image. This mask is translated into our graph cuts framework as partial grouping constraints where two pixels in the background must belong together in the final segmentation. We form the constraint matrix U from the collections of these pairwise grouping constraints.

3.3.5 Algorithm

Given image I , we compute a segmentation with the following procedure:

1. Build a gamma image stack where $I_n = I^{\gamma_n}$;
2. For each gamma image I_n ,
 - (i) compute attraction A_n and repulsion R_n ,
 - (ii) compute pixel correspondence C_n ,
 - (iii) compute $A_{n \rightarrow 1}, R_{n \rightarrow 1}$ by projecting A_n, R_n to the original image I ;
3. Compute total attraction A and repulsion R by collapsing the stack;
4. Form partial grouping constraints U from a background mask ;
5. Solve the eigenvectors of weights $W = A - R + D_R$ with constraints U ;
6. Obtain a discrete segmentation from the eigenvectors.

3.4 Experiments

We implement our algorithm in MATLAB. The same set of parameters are used for all our images: $\gamma = \{1, 2, 4\}$, $\sigma_a = 0.3$, $\sigma_r = 2\sigma_a$, window sizes 8 and 16 for attraction and repulsion respectively. We choose the number of eigenvectors k according to the expected number of hair bundles in the images.

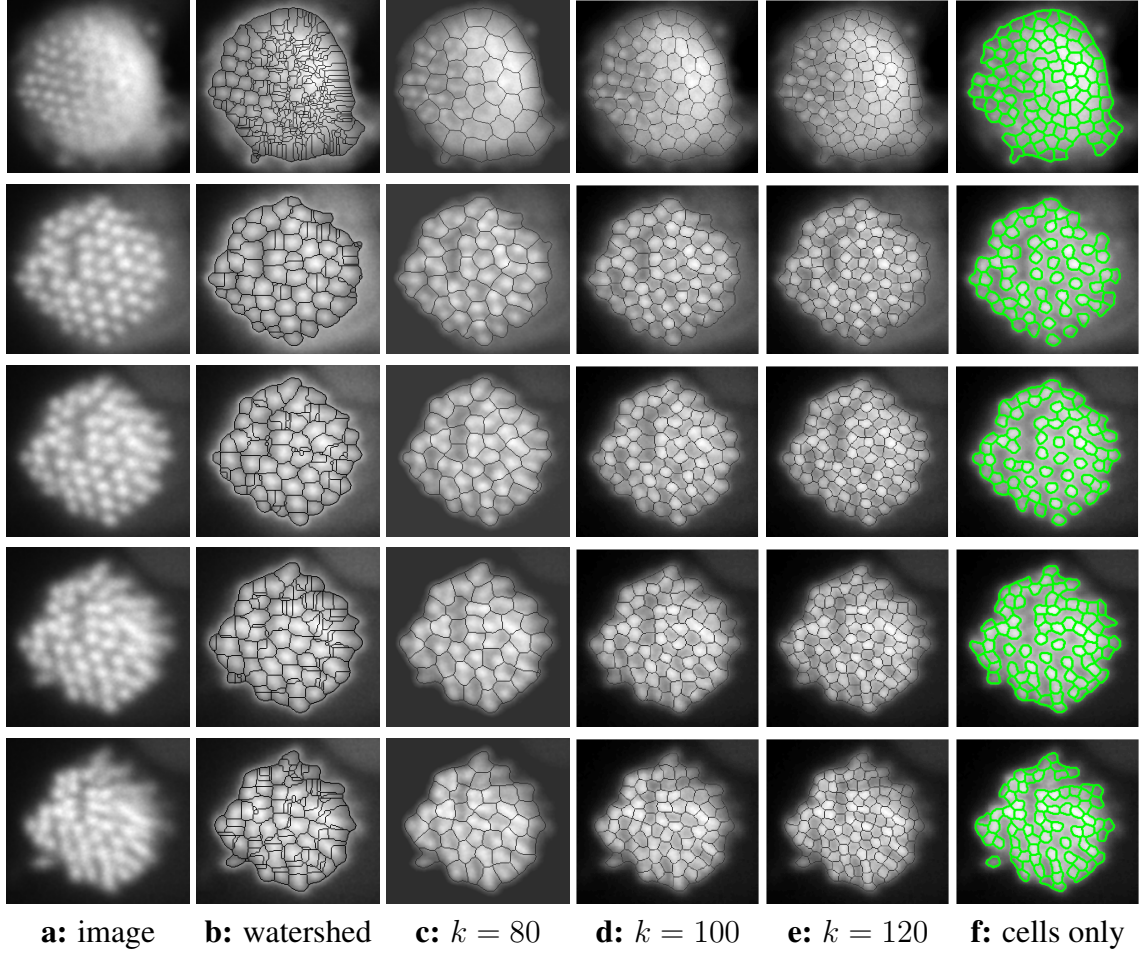


Figure 3.6: Coarse-to-fine stereocilia segmentations. For each image (Column **a**), we compare our results (Columns **c-e**) with watershed (Column **b**) when the number of eigenvectors k increases. Extracted stereocilia (Column **f**) show that our method is robust to local intensity fluctuation, can discover weak peaks, and have precise boundaries.

Fig. 3.5 shows that better segmentation is achieved by integrating cues over the entire gamma stack instead of an individual gamma image. Single peaks originally faint or without clear boundaries are enhanced in gamma transformed images. However, with an increasing gamma, valleys are widened and boundaries become less precise. Cutting across the gamma image stack allows segmenting out weak peaks while retaining precise boundaries throughout the image.

Fig. 3.6 shows our coarse to fine segmentations. When the number of eigenvectors k is small, our segmentations resemble the watershed results. However, our segmentations

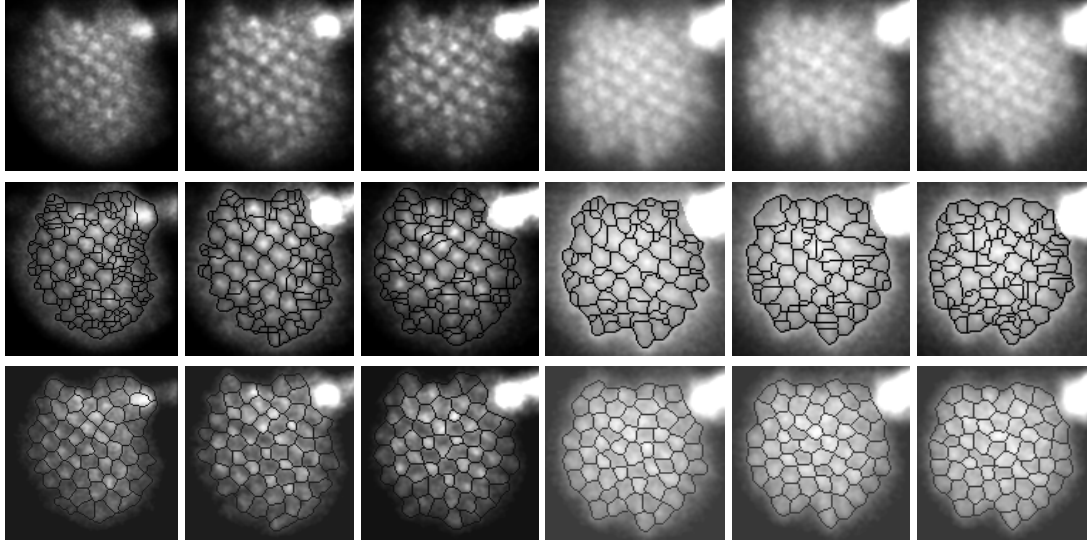


Figure 3.7: Our method works equally well on noisy and low-contrast images. $k = 40$. Rows 1-3 show images, watershed results and our results respectively.

are not disrupted by local intensity fluctuation and do not cut through salient peaks. When k increases, our segmentations locate each peak with tighter delineation. Most noticeably, our method is able to successfully segment weak peaks without utilizing the near regularity of stereocilia layout. Fig. 3.7 shows additional results on images of poor imaging quality.

We measure the goodness of segmentation by scoring it with respect to the ground-truth center locations of stereocilia. Let $\text{disk}(i)$ denote a disk of some fixed radius throughout the haircell bundles, located at stereocilium center i . Let $\text{segment}(i)$ denote the segment of maximal overlap with $\text{disk}(i)$. Our score is a number between 0 and 1, measuring the extent of overlap between the two:

$$\text{score}(i) = \frac{\text{disk}(i) \cap \text{segment}(i)}{\text{disk}(i) \cup \text{segment}(i)}. \quad (3.20)$$

The higher the score, the more precise the segmentation. As the number of eigenvectors increases, our segmentations capture a more precise shape of individual stereocilia. Fig. 3.8 shows with an image example as well as statistics that our method overall scores higher than watershed.

Our method segments the background into multiple valley regions, which are of no

interest to medical researchers. By requiring the mean intensity in the region center to be higher than the periphery, we get rid of valleys and automatically extract the stereocilia, as shown in Fig. 3.6f.

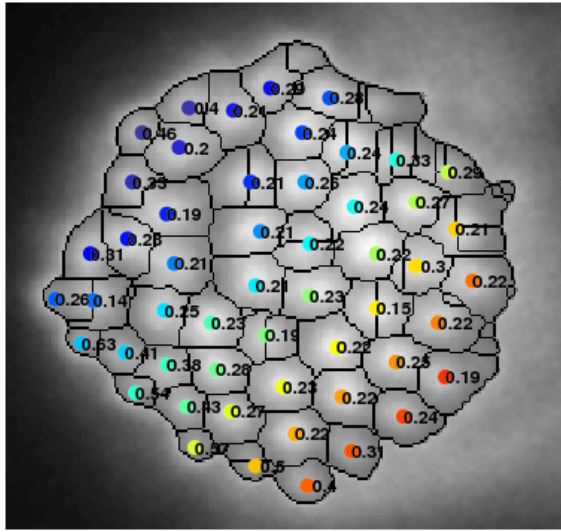
3.5 Summary

The segmentation of medical images appears to be governed by the global intensity level, yet local intensity fluctuation poses considerable challenges to both local methods such as watershed and global methods such as level sets.

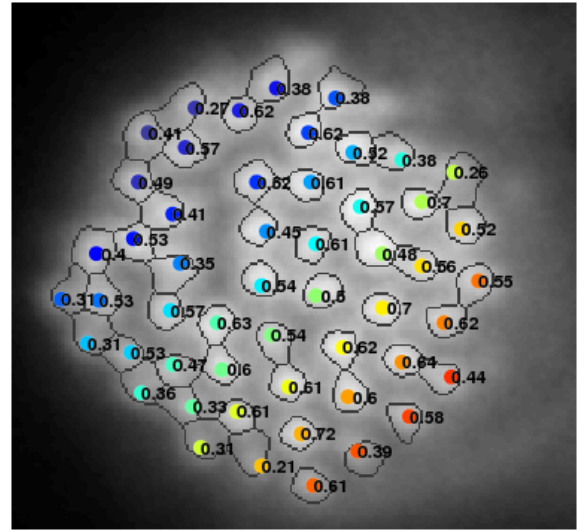
We develop a spectral graph-theoretic method which finds the best segmentation on every gamma transformed version of the original image. The local grouping cues at each gamma image include short-range intensity similarity cues that pull pixels towards stereocilia centers, and long-range intensity difference cues that push pixels away from stereocilia boundaries. When they are projected back to the original image, we can seek the optimal graph cuts across the aligned cue stack which maximize within-group attraction and between-group repulsion. The near-global optimal solution can be found efficiently using eigendecomposition.

Our method has only a few parameters and requires little tuning. We obtain accurate and robust results on a variety of low-contrast images with the same set of parameters, showing the advantage of cutting across the entire gamma stack instead of the original image or any gamma image alone.

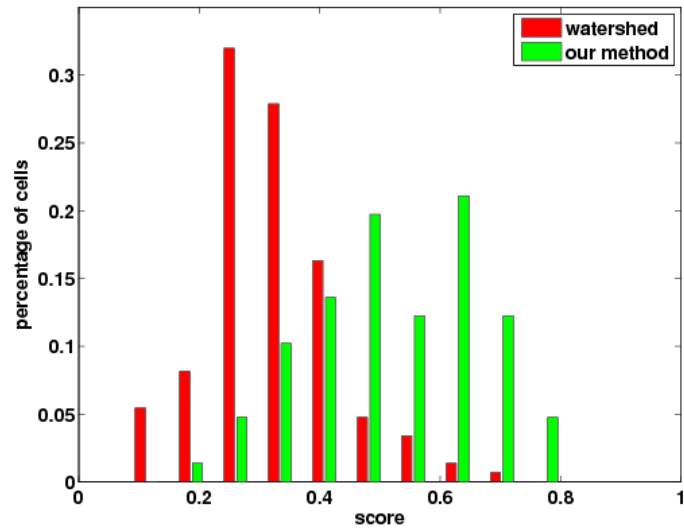
The segmentation issues we investigate in this paper are not restricted to stereocilia images. Our approach of making a global decision based on two types of local cues at different spatial ranges provides a robust and efficient alternative to watershed or level set methods in many medical image applications.



a: watershed



b: our method at $k = 120$



c: distribution of scores

Figure 3.8: Segmentation scores with respect to ground-truth stereocilia centers. These center locations are marked by colored dots. Each number indicates the score of a particular segment that contains a stereocilium center. **a** and **b** show a score example for watershed and our method. **c** shows the distribution of scores from all the images. Our method has a higher score than watershed overall.

Chapter 4

Finding Dots: Segmentation as Popping out Regions from Boundaries

4.1 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. 4.1). 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. 4.1a), large intensity variation (Fig. 4.1b-c), and extensive occlusion and conjunction between dots (Fig. 4.1c-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.

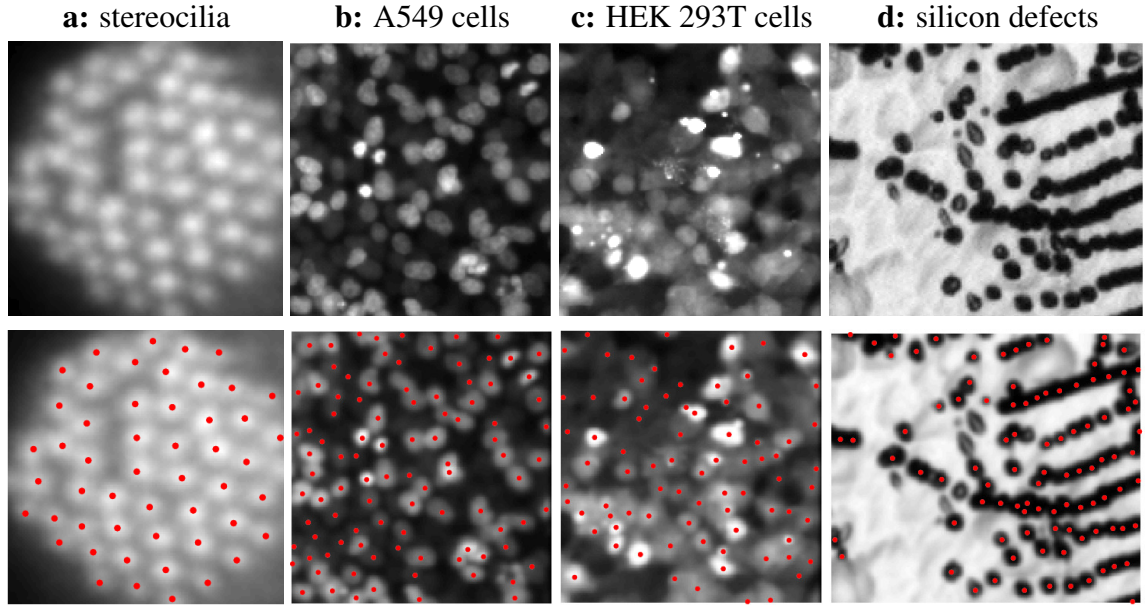


Figure 4.1: Finding dots in an image is a challenging segmentation problem when the image has poor imaging quality, large intensity variation, occlusion and conjunction between dots. These dots could be: **a)** Haircells in hearing research, courtesy of Pathak and Corey at Harvard University, **b,c)** 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.

Image segmentation is conventionally formulated as separating regions of homogeneous features such as intensity and texture [Gra06b, SM00, Gra06a, MS97]. 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 [Mey94, LM98, XP98, ZSS07, WSC09].

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. 4.2a) or object recognition and grasping, it is not necessary for our dot finding (Fig. 4.2b) and many other applications. Where boundaries should be located is flexible, so long as the core of each

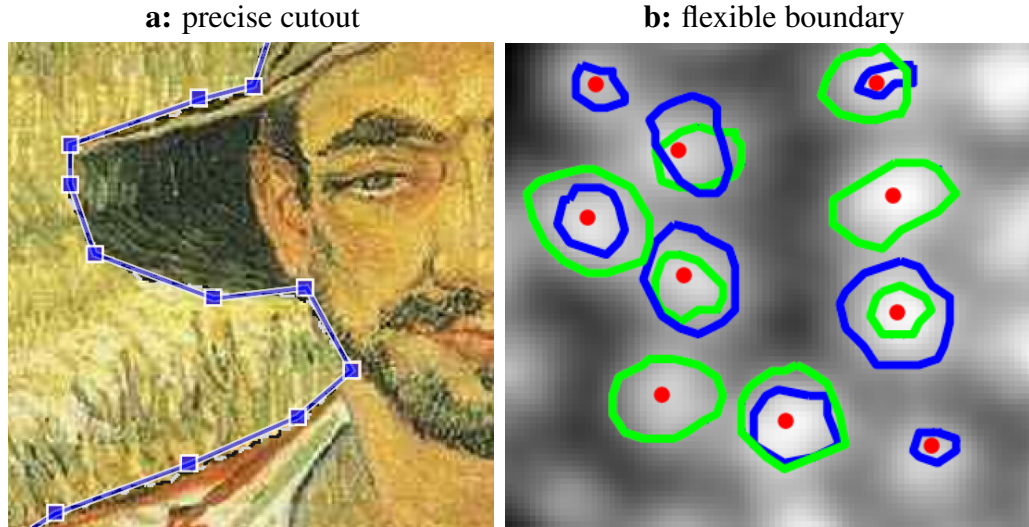


Figure 4.2: Precise boundary delineation is useful for image manipulation but not necessary for our dot finding applications. **a)** Image cutout (Li et al, Lazy Snapping, SIGGRAPH 2004) is an editing tool which produces a segmentation that follows the facial contour precisely. **b)** Finding haircells only requires locating the core of dots where the boundaries between dots could be flexible.

dot is retained in the region.

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. 4.3).

That boundaries form regions of their own has long been observed in [MBLS01], but only as a hazard to real image segmentation which needs to be actively suppressed [MFM04]. 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.

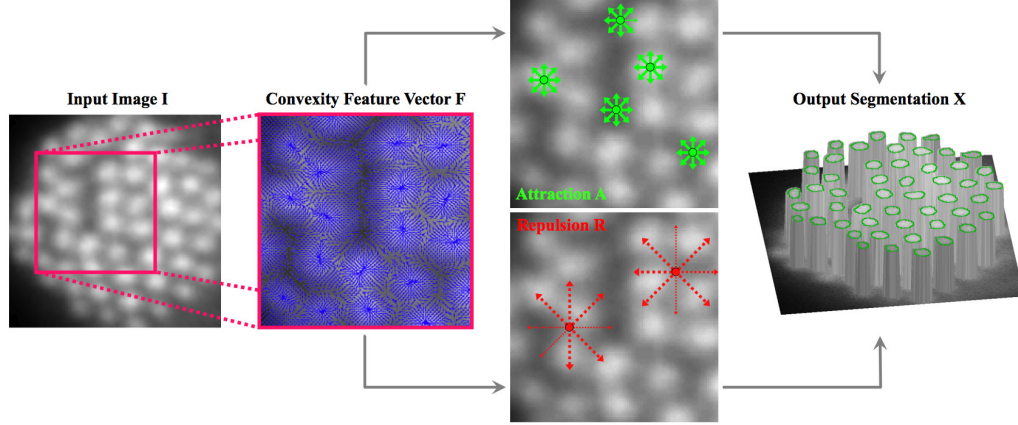


Figure 4.3: Finding dots overview. Given image I , we first compute a flow field *peak direction* vector p which characterizes the geometry of local intensity distributions: A pixel inside a dot (intensity peak) becomes a sink, and a pixel outside the dots (intensity valley) becomes a source. We then compute (green) attraction A between nearby pixels based on similarity of a *convexity feature* vector derived from p , and (red) repulsion R between distant pixels based on dissimilarity. A two-way node partitioning based on both attraction and repulsion pops out all the dots from their backgrounds simultaneously as one foreground region with many disconnected components.

Most evident in Fig. 4.1d, 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. 4.3F).

We formulate our algorithm in the spectral graph cuts framework [SM00], 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 pixels 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. 4.3A, *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. 4.3X).

Our method works better than other 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.

4.2 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 [YS01, YS03].

4.2.1 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 :

$$p(i) \propto \sum_{a \in N(i)} T(i, a)(L(a) - L(i)), |p(i)| = 1 \quad (4.1)$$

$$T(i, a) = \sum_{\substack{I(m_1) \leq I(m_t) \leq \dots \leq I(m_k) \\ m_1 m_2 \dots m_k = \text{line}(i, a)}} I(m_t) \quad (4.2)$$

$p(i)$ 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. 4.4a).

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:

$$F(i, a) = \langle p(i), p(a) \rangle \quad (4.3)$$

where \langle, \rangle 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. 4.4b,c), whereas pixels between dots have both positive and negative values (Fig. 4.4d).

4.2.2 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.

$$S(i, j) = \frac{\langle F(i, :), F(j, :)\rangle}{|F(i, :)| \cdot |F(j, :)|}, \quad j \in N(i) \quad (4.4)$$

$$A(i, j) = e^{-\frac{1-S(i, j)}{\sigma}}, \quad |L(j) - L(i)| \leq r_A \quad (4.5)$$

$$R(i, j) = \frac{1 - S(i, j)}{2}, \quad |L(j) - L(i)| \leq r_R \quad (4.6)$$

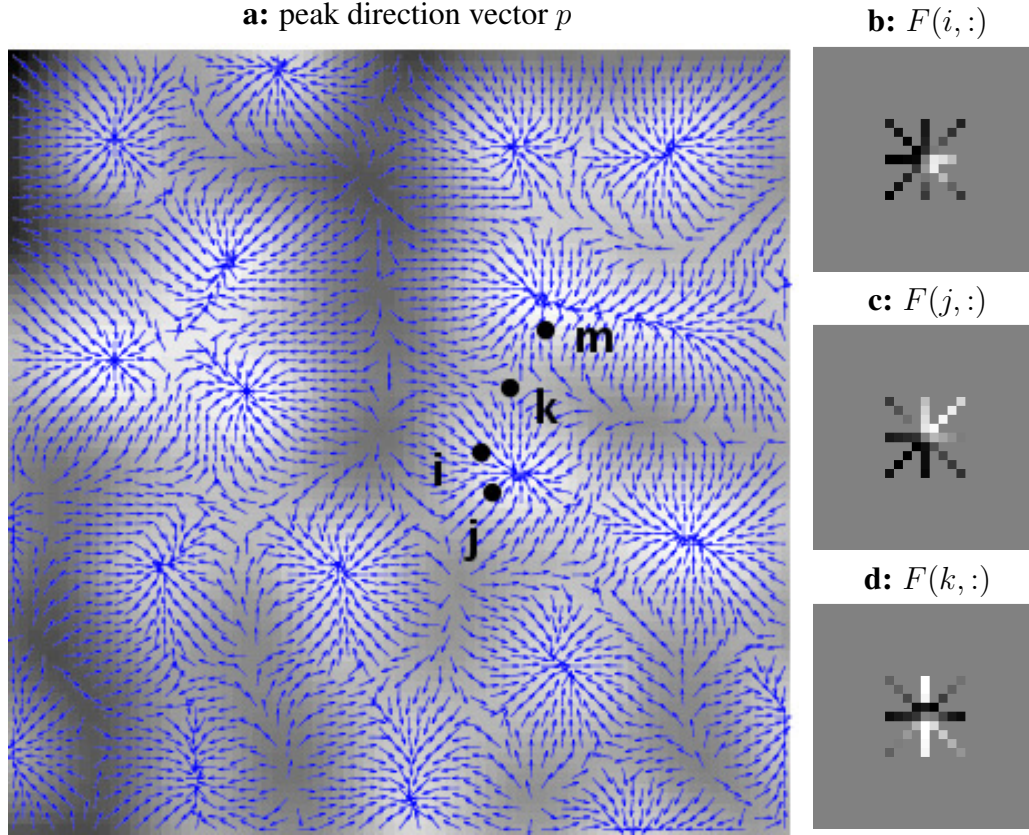


Figure 4.4: Pixel-centric convexity feature vector F . **a)** Peak vector p at each pixel points towards the center of dot the pixel belongs to. The dot centers and boundaries are sinks and sources in the vector field. **b,c,d)** Feature vector F at pixels i, j, k marked in **a)**. $F(i)$ indicates how 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.

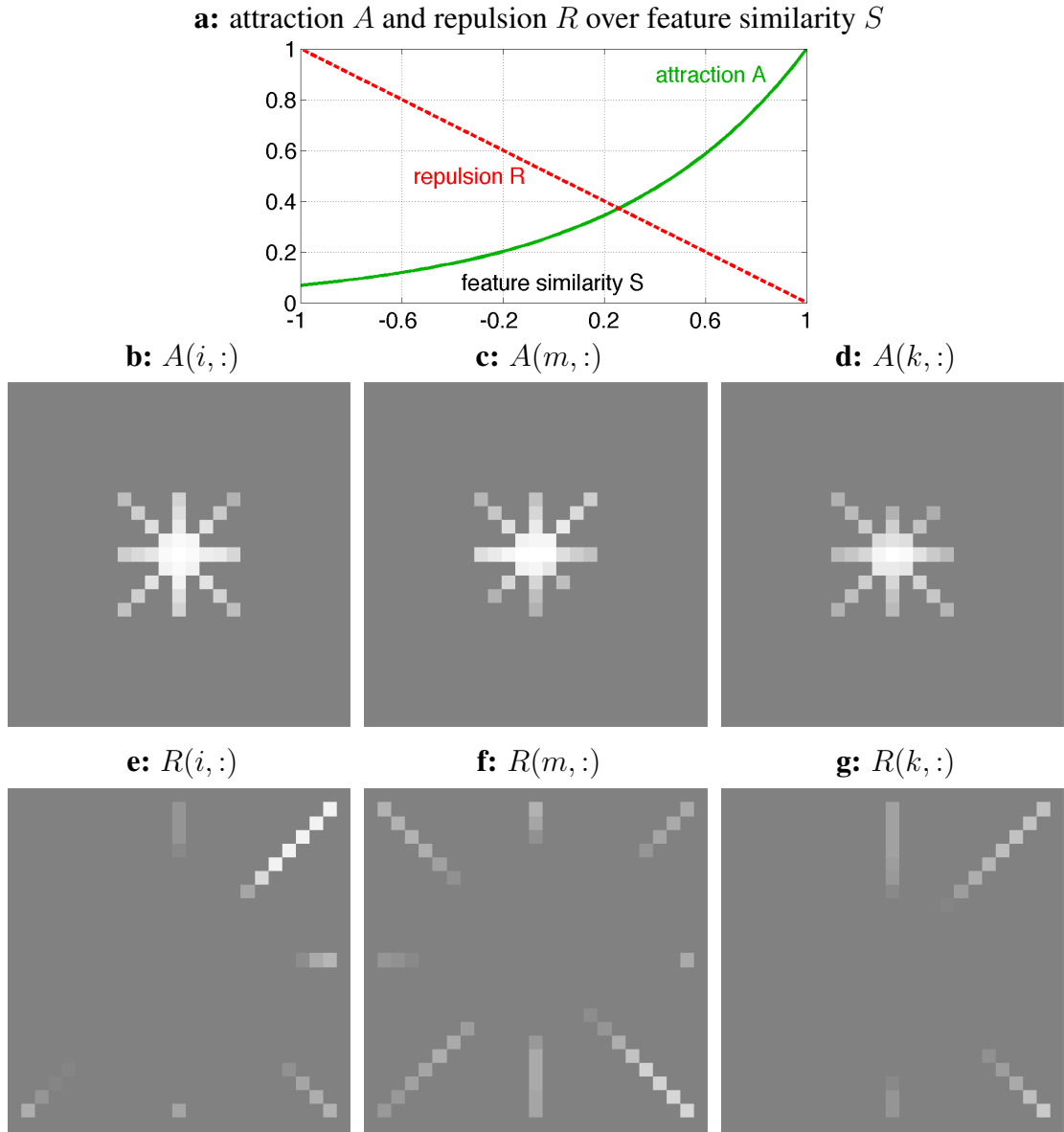


Figure 4.5: Short-range attraction A based on feature similarity S and long-range repulsion R based on feature dissimilarity $1 - S$. **a)** Attraction A is proportional to S and defined for nearby pixels, whereas R is inversely proportional to S and defined only for distant pixels. **b,c,d)** A and **e,f,g)** R at pixels i, m, k marked in Fig. 4.4a. Both i and m repel k , causing them to group together.

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.

4.2.3 Graph Cuts with Attraction and Repulsion

Given attraction A and repulsion R between pixels, we segment the image using the normalized cuts criterion [YS01]:

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

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 $D_W = \text{Diag}(W1_n)$ the diagonal *degree* matrix for any $n \times n$ weight matrix W .

$$\begin{array}{ll} \text{maximize} & \varepsilon(X) = \sum_{g=1}^2 \frac{X_g^T W X_g}{X_g^T D X_g} \end{array} \quad (4.7)$$

$$\begin{array}{ll} \text{subject to} & X \in \{0, 1\}^{n \times 2}, X1_2 = 1_n \end{array} \quad (4.8)$$

$$U^T X = 0 \quad (4.9)$$

$$\begin{array}{ll} \text{where} & W = A - R + D_R, \end{array} \quad (4.10)$$

$$D = D_A + D_R \quad (4.11)$$

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 .

We follow the solution procedure developed in [YS04, YS03] and use their code online to find a near-global optimum to this constrained normalized cuts problem.

4.3 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. 4.9) 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 [BY09], 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. 4.6).

Meanshift: We use a local implementation [BC04]. 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 [AMFM09]. It relies on Pb [MFM04] 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.

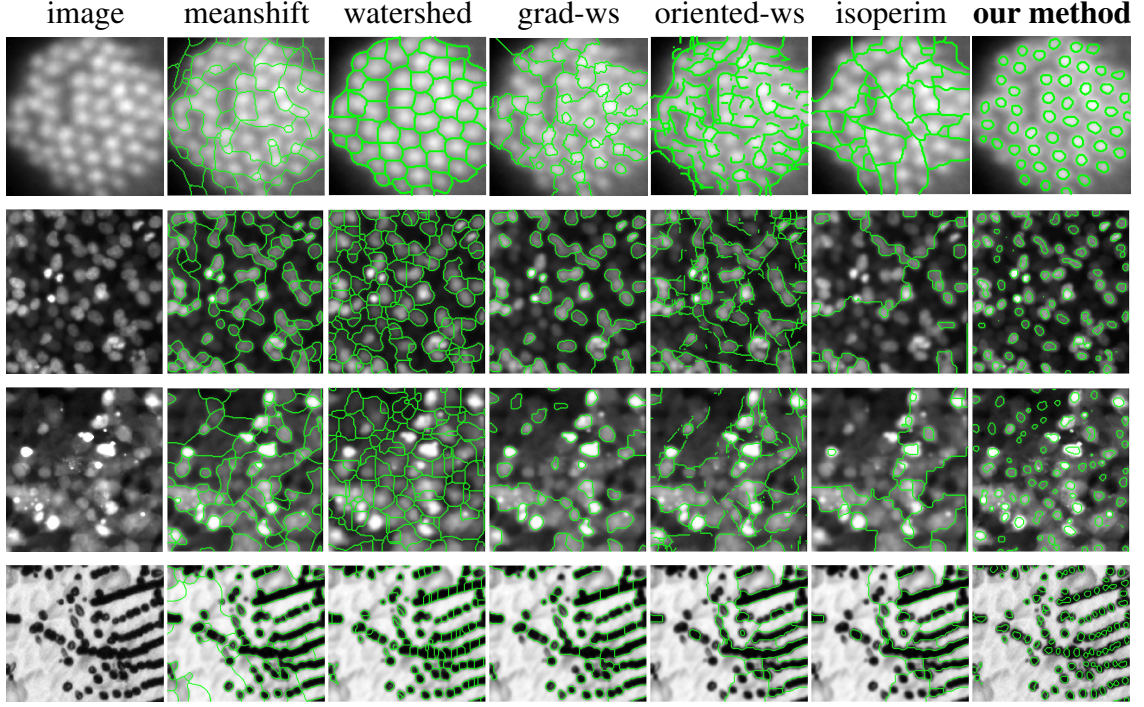


Figure 4.6: Comparison of results on the 4 representative images in Fig. 4.1. Meanshift cannot break up conjoined dots. Standard watershed tends to oversegment, and gradient-based watershed misses many weak dots. Oriented watershed does not produce closed regions. Isoperimetric segmentation tends to 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.

Isoperimetric: We use the implementation provided by [Gra06a]. 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.

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. 4.8, Fig. 4.9).

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

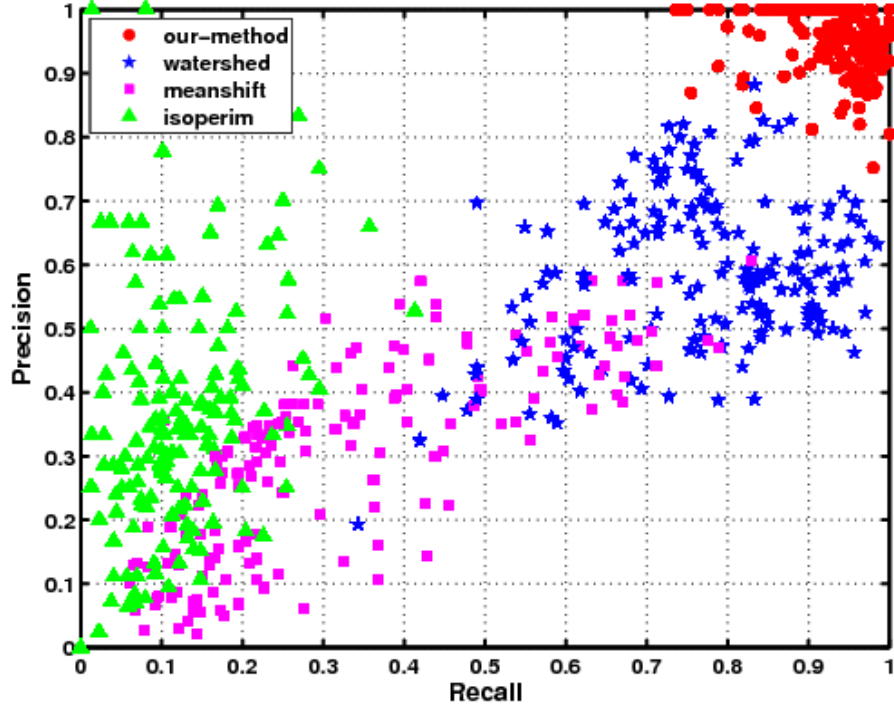


Figure 4.7: Precision-recall statistics for meanshift, watershed, isoperimetric, and our method on 4 categories of microscopic images. The oriented watershed shown in Fig. 4.6 is not included as it often does not produce closed dot regions. Our method (red round dots, upper right corner) has better precision and recall overall.

dot i and segment j . If it is less than a certain radius threshold ρ , we consider (i, j) a matched detection. We define

$$\text{precision} = \frac{\#\{j : \min_{i=1}^m D_{ij} \leq \rho\}}{n} \quad (4.12)$$

$$\text{recall} = \frac{\#\{i : \min_{j=1}^n D_{ij} \leq \rho\}}{m} \quad (4.13)$$

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. 4.7 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. Our method fails to detect some extremely blurry haircells in the periphery which usually only experts can pick out.

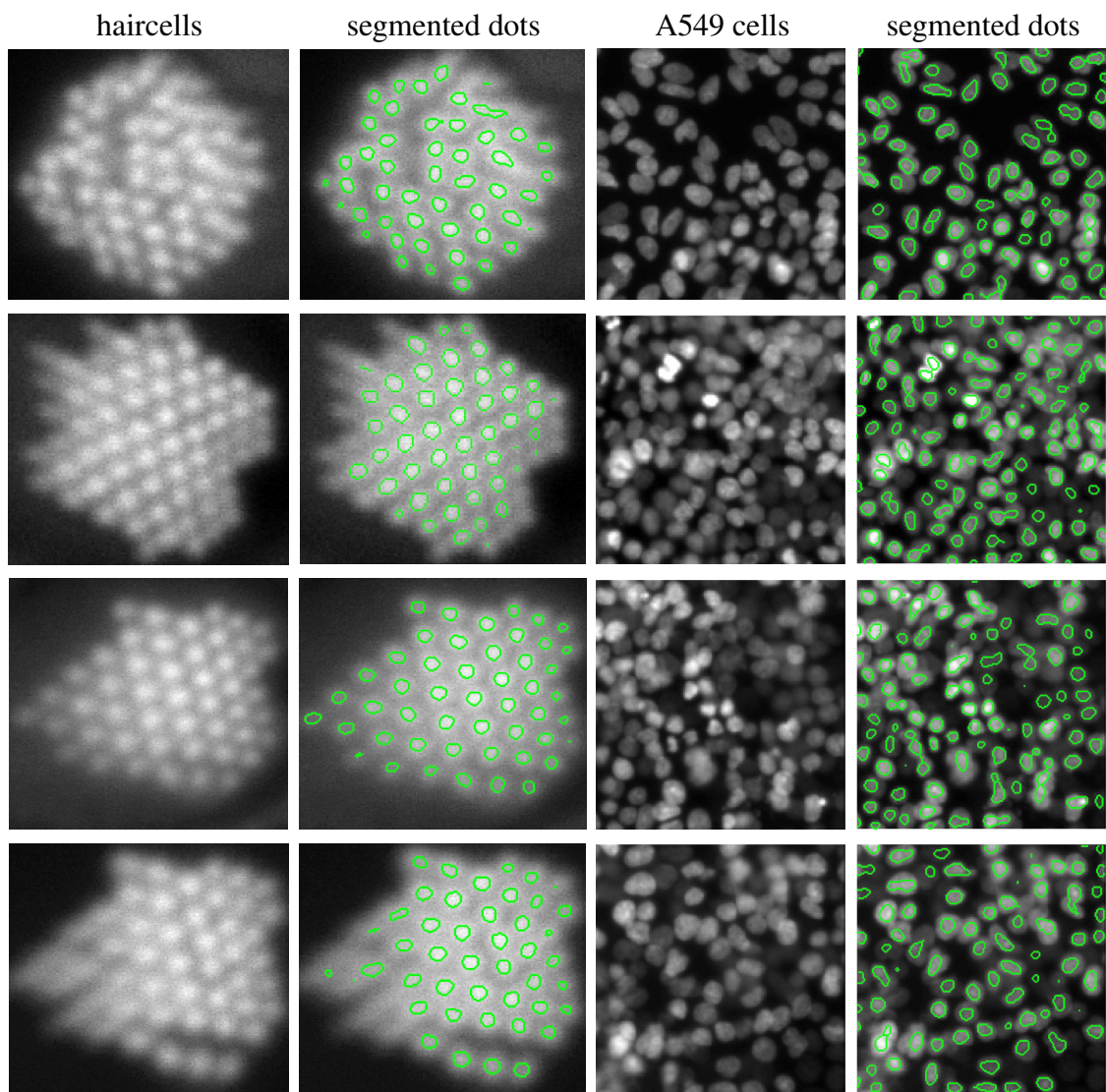


Figure 4.8: Our sample results on haircells and A549 cells. We can segment out clear or fuzzy dots in a near-regular or irregular layout.

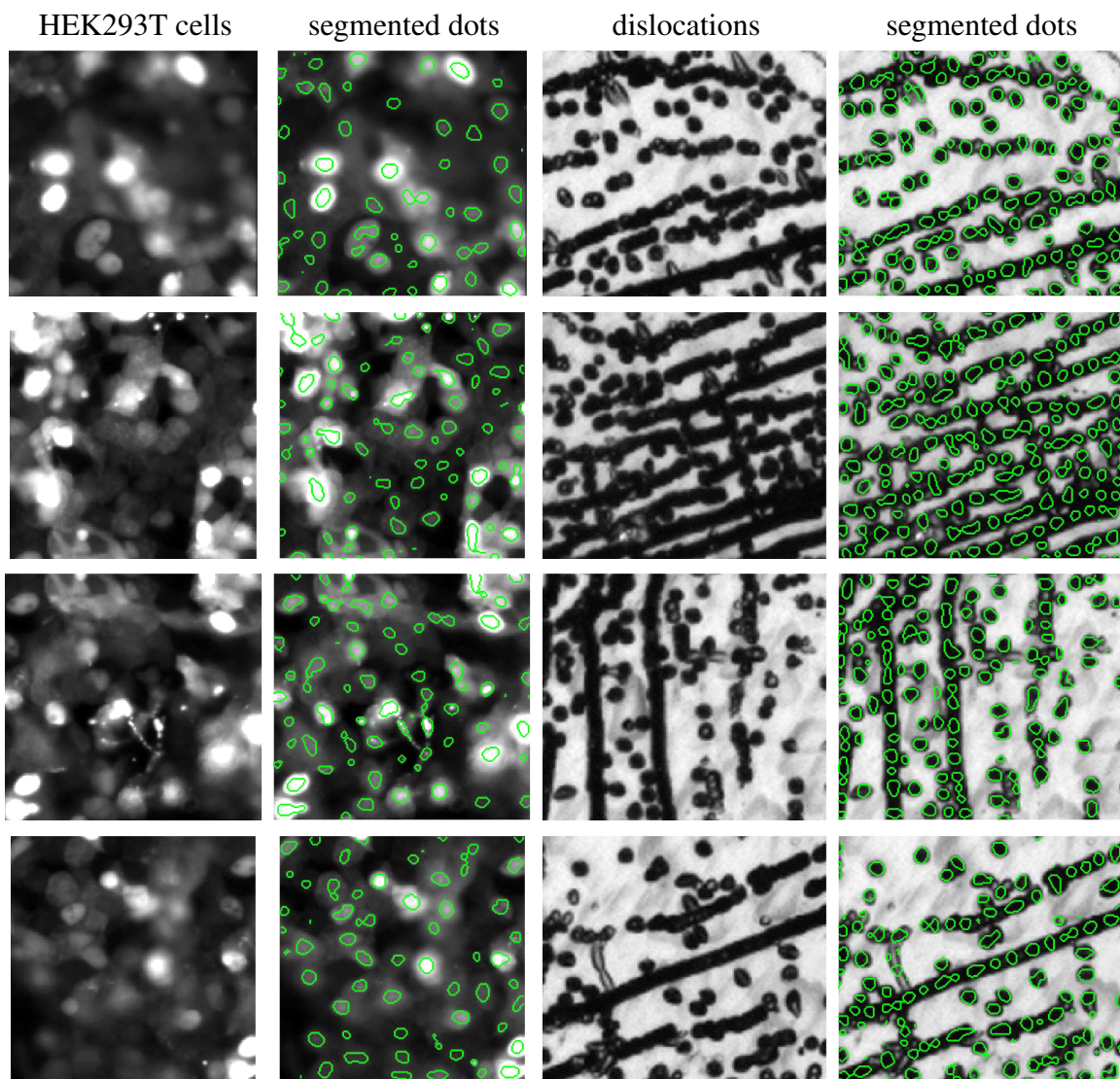


Figure 4.9: Our sample results on HEK293T kidney cells and silicon wafer dislocation Images. We can segment out dots independent of the size of their intensity contrast and even in the presence of partial occlusion and dot conjunction.

4.4 Summary

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 larger spatial range, it plays an active and complementary role to local attraction. Popout would not be possible without repulsion. While the mechanism of attraction and repulsion in spectral graph theory has been elucidated in [YS01], 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.

While our method is designed to pop out all the dots in a microscopic image based entirely on the convexity flow field feature, Fig. 4.10 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 (further explored in Chap. 8). These interesting results open up new possibilities for shape from texture algorithms.

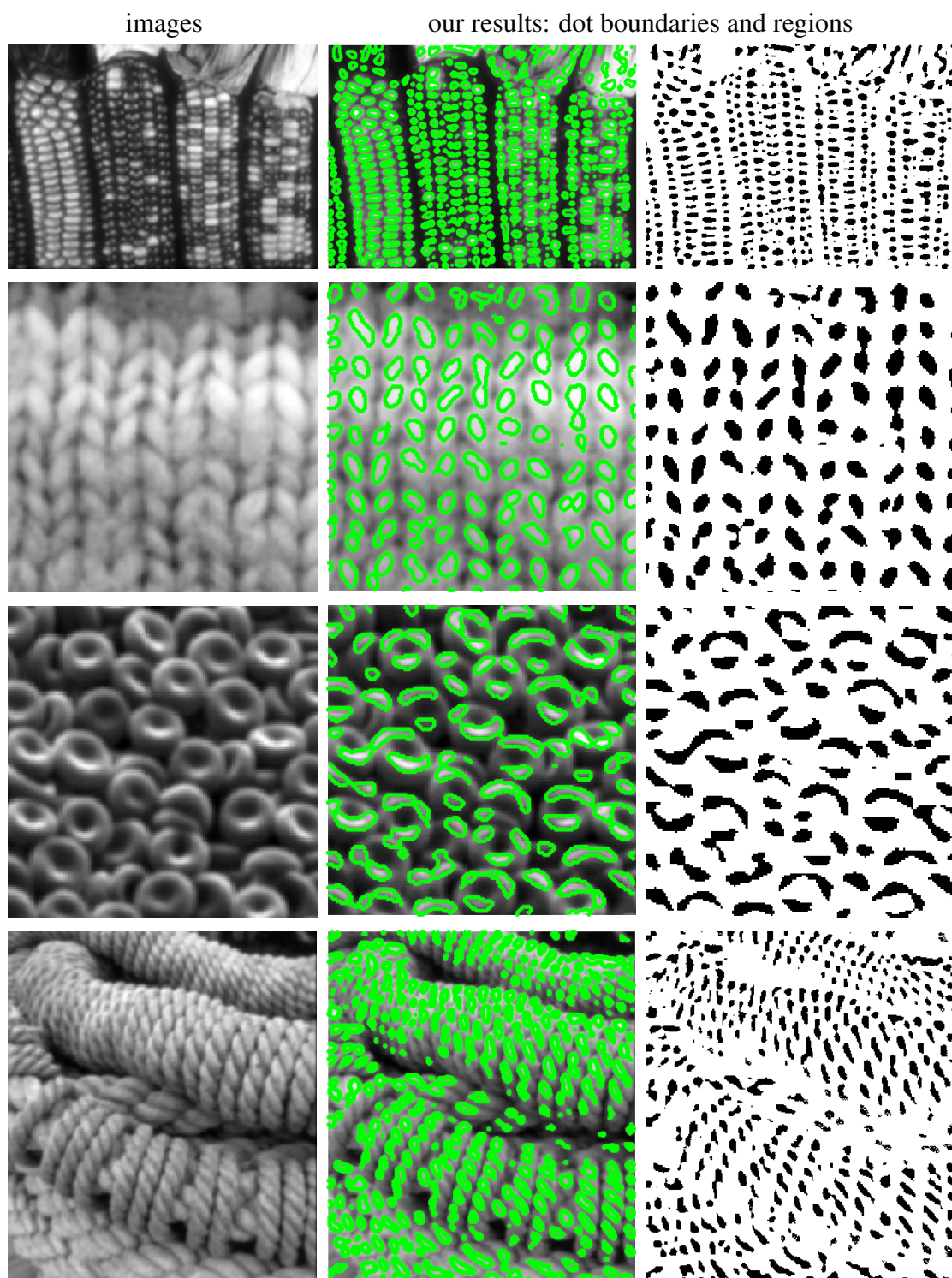


Figure 4.10: Finding dots in natural scene images with the same set of parameters used for microscopic images. These dots, i.e. small repetitive patterns, are useful for recovering shape from texture.

Chapter 5

Pop Out Many Small Structures from a Very Large Microscopic Image

5.1 Introduction

There is often a need in medical research to count, measure, and compare numerous small structures in a large image. Recall from Chap. 1 the examples in Fig. 5.1, which can be very different in nature and visual appearance. They could be from a frog's inner ear (left) or from a drosophila's fly brain region from electron microscopy (EM) data (right). On the left, regions of interest could be the larger scale haircell bundles or the smaller scale individual stereocilia that compose them. While cluster intensities vary across the image, cell intensities peak towards the center of each bundle. On the right, salient regions assume a larger range of shapes and they are densely packed in the image together with new elongated structures. Whether one wants to extract a 15 pixel cilia from the frog's hairbundles or a similar size vesicle in the fly's brain, one has to address complexity issues associate with a 1600×1600 pixels image. In both images, the regions of interest are very small when compared to the large image size.

As we have shown in the previous chapters, finding these small structures is a challenging segmentation problem on its own. Figures 5.2 and Fig. 5.3 displays images from two

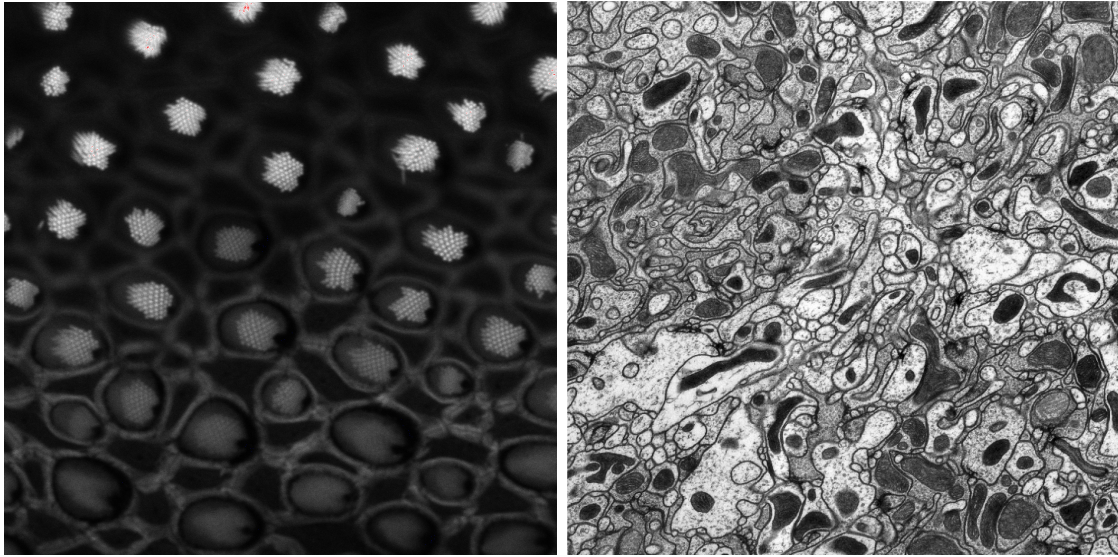


Figure 5.1: Many small structures in a large image. Left: stereocilia bundles of the frog's inner ear, 1600×1600 pixels (Image Courtesy: Medha Pathak and David Corey at Harvard). Right: electron microscopy (EM) data from the medulla brain region of the drosophila fly, 1800×1800 pixels (Image courtesy: Mitya Chklovskii, C Zhiyuan Lu, Rick Fetter, Shinya Takemura and Ian Meinertzhagen at Janelia Farm Research Institute). Cells are approximately 15 pixels in diameter.

human pathology image datasets in cancer research. The appearance variety of these cells illustrate common problems encountered when dealing with these small regions. Fig. 5.2 displays images of epithelial and embryonic cells, with faint boundaries, large intensity variations and occlusions. Fig. 5.3 display histopathological images of tumor-like lesions with textured cells and non-homogenous backgrounds.

The challenge of segmenting many small structures in a very large image is therefore two-fold: fine segmentation granularity when dealing with the size of the small segment and segmentation complexity when dealing with the size of the large image.

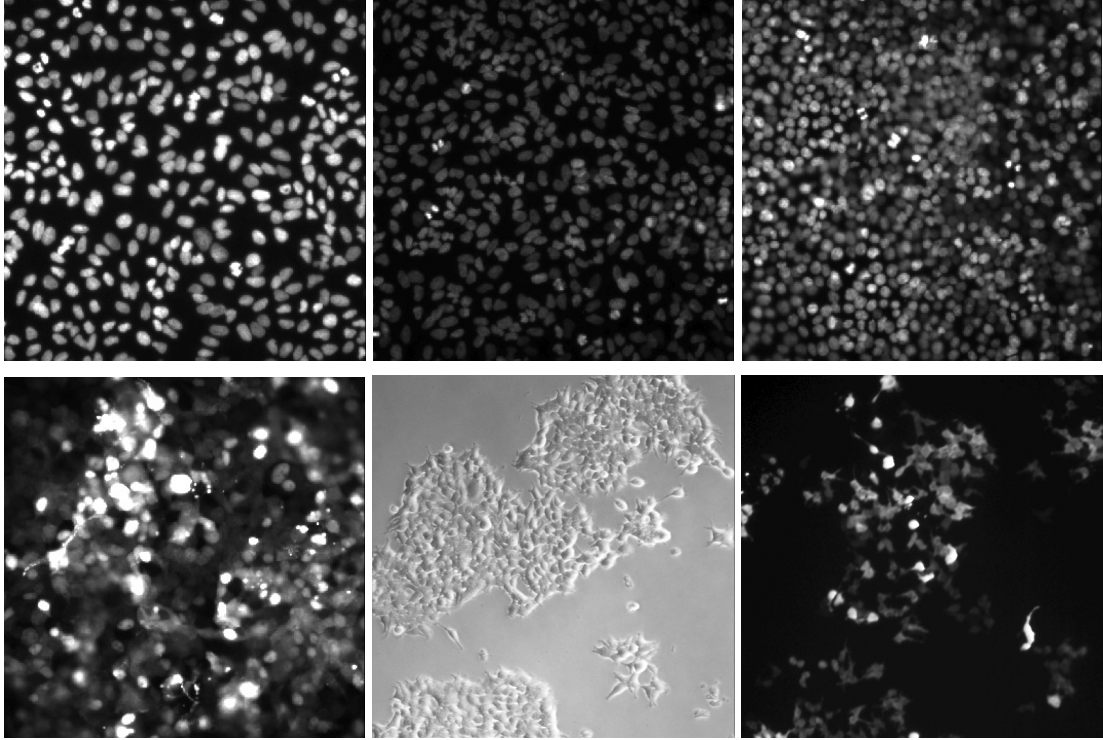


Figure 5.2: Segmenting cells challenges. **Row 1:** Epithelial A549 and embryonic kidney HEK293T cells featuring faint boundaries, varying dot intensities and occlusions. **Row 2:** More HEK cells of different convex geometries often appearing as cell clusters (Image courtesy: Nisha Sosale at UPenn).

5.1.1 Challenge 1: Segmenting many small structures

Segmentation of medical images often appears to be governed by global intensity levels, yet imaging noise and local intensity fluctuation present considerable challenges. Faint regions, similar intensities between adjacent regions and conjoined cells make these images challenging even for a human eye.

Many segmentation approaches for these type of image are based on mathematical morphology and energy-driven methods. In mathematical morphology, the watershed transform [Mey94] is applied to extract an initial set of contours, and markers or seeds are used to refine the contours of interest. The watershed transform is computationally very efficient [Mey05], but finding seeds automatically is application-driven and can be very challenging. Without proper seeds, oversegmentation results, since watershed is easily

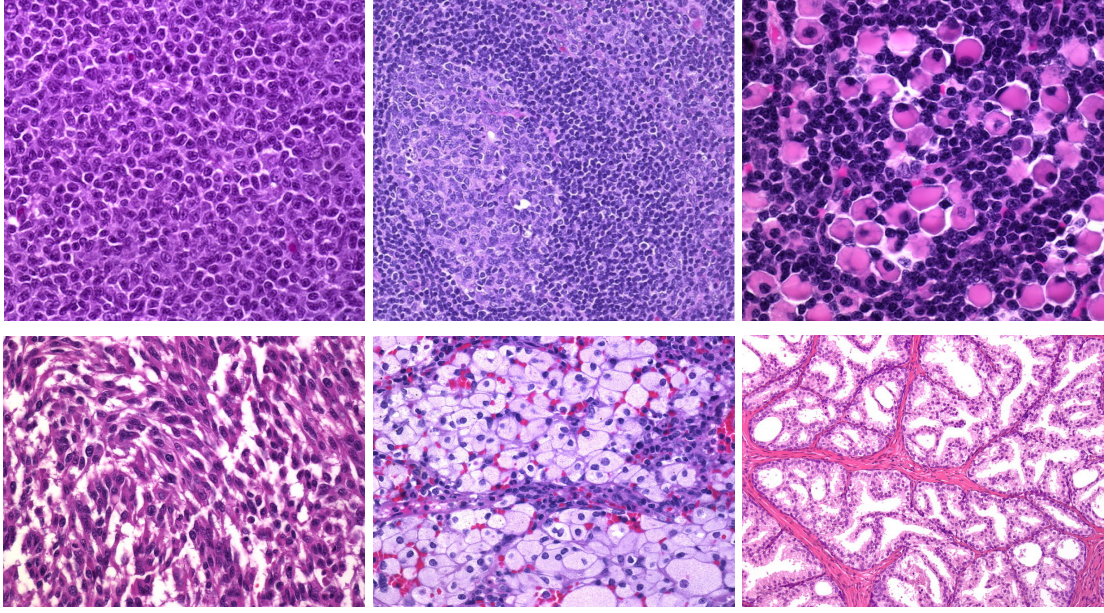


Figure 5.3: Segmenting cells challenges. **Row 1:** Histopathology images depicting tumor-like lesions: cells of various sizes and textures in spleen cell lymphoma. **Row 2:** spleen tumor cells metastatic melanoma, clusters of glands in ovarian cells and clear cell cribriform hyperplasia from prostate illustrating cells on non-homogenous backgrounds (Image courtesy: www.webpathology.com).

disrupted by local intensity fluctuations. Less prone to local noise fluctuations, energy-driven methods involve the minimization of an energy function formulated either on regions [MS89, GG90, ZY96] or contours, such as snakes [XP98] and level set methods [MS97]. These algorithms though are computationally costly and they depend on initial seed choices. Various techniques have been proposed to combine the benefits of watershed and energy-driven methods, e.g. level sets for watershed [THW⁺07] or watersnakes [NWvdBW03] that allows to inject smoothness priors in the watershed formulation.

Graph cuts methods have also been employed to overcome the limitations of watershed algorithms, e.g. segmenting a single connected component with isoperimetric graph partitioning [Gra06a]. In [CBNC09], watersheds are formulated within a graph setup. Their theory together with a larger family of segmentation methods including random walker [Gra06b] are generalized in the theory of power watersheds [CGNT09]. The latter though also depend on initial seeds and are thus not tailored for images with many small conjoined

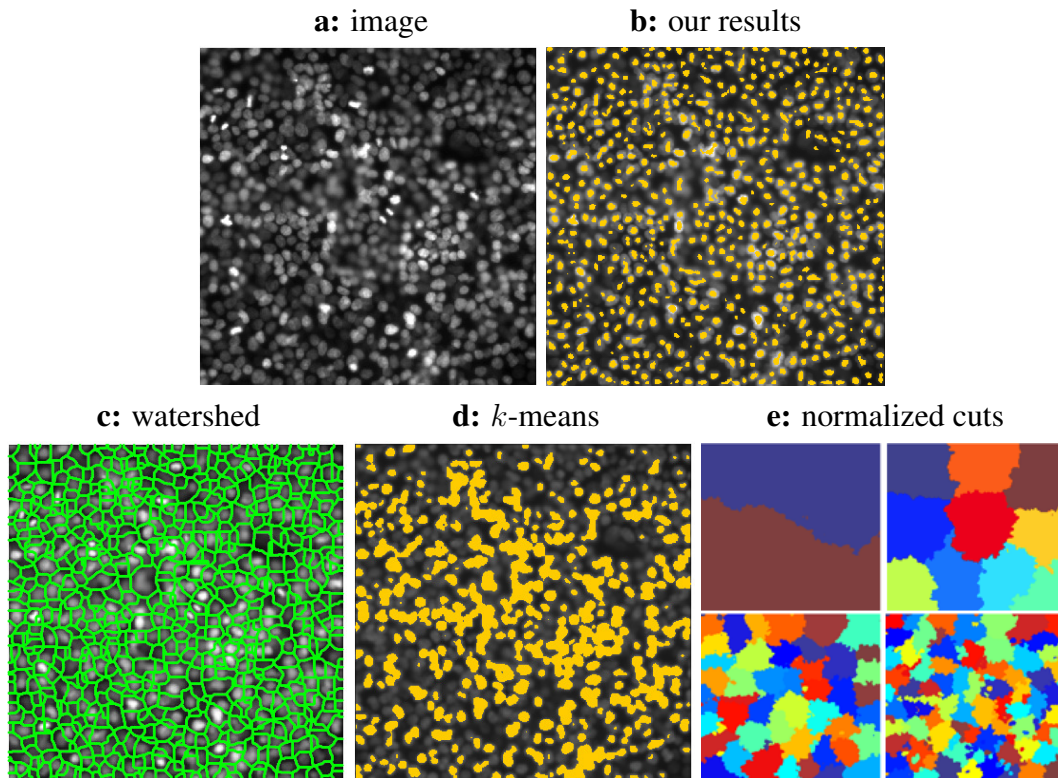


Figure 5.4: Efficiency versus robustness dilemma for segmenting small regions in a large image. **a,b**) Image of epithelial cells and our resulting cell segmentation (gold). **c,d**) Efficient two-way segmentations by watershed, k -means. Watershed oversegments in the presence of local intensity fluctuations while k -means is unable to distinguish cells of similar intensities apart. **e**) Normalized cuts (N-cuts) for 2, 4, 32, 64 regions is meant to segment large regions in natural images. While robust in general, it fails for these structures.

regions with faint boundaries and no clear minima.

5.1.2 Challenge 2: Dealing with a large image

Segmentation methods such as watershed and k -means clustering are efficient but unable to deal with large intensity variation (Fig. 5.4c,d). On the other hand, spectral graph partitioning methods [SM00, Yu05] are prized for their ability to grasp the large structural organization of an image from the global integration of local cues. While this property is desired for understanding a real-scene image, it unnecessarily handles a huge number of

pixels in a large image, since segmenting cells in one region really should not be influenced by cells far from them. It also prevents small structures from being segmented all at once (Fig. 5.4e), since a larger image size leads to larger regions instead of numerous small ones given a fixed number of segments.

The two main approaches to reduce complexity, coarse-to-fine and multiresolution segmentations [Yu05, FH04, CBS05, BZ03, GSBB03, HPB97, Yu04], are not suitable for this task. The former approach speeds up the segmentation by initializing a finer segmentation with the results of a coarser one, whereas the latter integrates features at multiple scales to yield a better segmentation. Since small structures are not present in either coarser-scale segmentations or coarser-scale features, there is no help to be gained from either approach.

5.1.3 Our Solution: Popping out many small structures in a large image

We propose a spectral-graph framework which scales effectively with image size without losing the fine granularity of small segments (as shown in Fig. 5.4b). Our segmentation algorithm is outlined in Fig. 5.5.

We focus on segmenting small convex regions (e.g. cells), which we will refer to as *dots*. Each image is divided into patches and each is segmented independently. Pixels in the image become nodes of a weighted graph, and finding dots becomes dissecting the graph based on weighted connections between nodes. Whereas attraction cues are commonly used to encode affinity between pixels, it is the crucial role of repulsion cues that allows popping out all dots simultaneously from a common background. In order to compute these local cues, unlike real-scene image segmentation [SM00], we do not use single edge features (e.g. large intensity gradients along region boundaries) to delineate regions. Instead, we use distributive local gradient fields to characterize geometrical distinction between region cores in the foreground and region peripheries in the background.

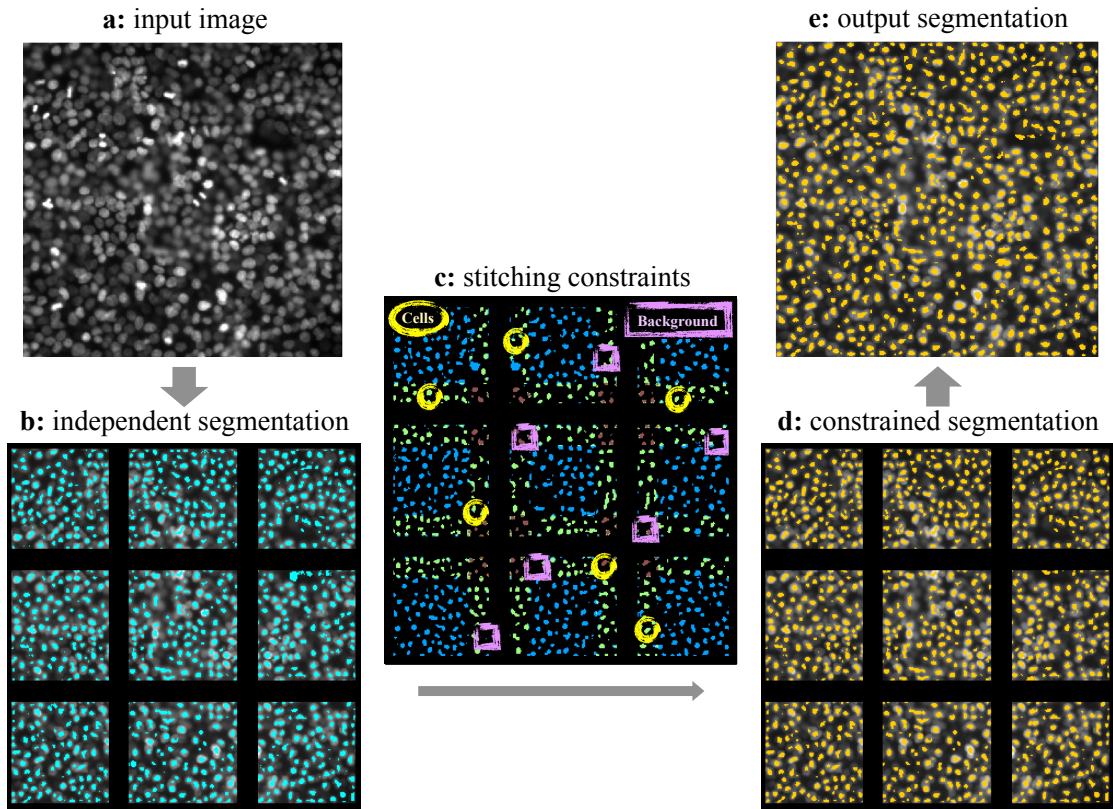


Figure 5.5: Segmentation subject to stitching constraints algorithm overview. Segmenting numerous small structures in a large input image (a) is performed as a series of independent patch segmentation subject to stitching constraints between neighboring patches. The constraints (c) are derived from mutual agreement analysis on adjacent patch segmentations from a previous round. Segmentations between neighboring patches are marked in blue, green, or maroon, if 1, 2, or more than 3 patches agree. Stitching constrains cells (yellow circles) and background (purple squares) to group within each type respectively. The constrained segmentation (d, in gold) improves the initial segmentation (b, in cyan) and can be seamlessly combined to obtain the final output (e).

Similar ideas have been used in [SKRV99] to detect critical points in images with topological numbers, and in the mean shift-generated displacement vectors used in [DTCW01] to guide active contour models.

The individual segmentations are then used to establish the agreement between the patches, which provide pairwise long-range stitching constraints to be respected by each patch. We run the segmentation again on each patch, but now subject to these pairwise constraints on its pixels. The segmentation can be solved efficiently as a constrained eigenvalue problem [YS04]. Since these segmentations have mutual agreement in the overlapping areas, their individual solutions can be collapsed into one segmentation on the entire large image. Segmentation subject to stitching is more than simple stitching. Constraints in the overlapping regions propagate through in the optimization process to also improve the interior segmentation.

This chapter presents the extension [BY11] of our theoretical model on segmentation subject to stitching constraints [BY10b] and finding dots [BY10a], presented in Chap. 4. In Section 5.2, we present a constrained spectral-graph partitioning framework that naturally integrates the stitching constraints with the fine segmentation granularity of the individual patches. In section 5.3, we present a detailed analysis of the parameters used in the algorithm and how they can be adaptively chosen without user interaction. The benefits of stitching are illustrated both on dot structures and on more complex geometries. Finally, we present results and performance comparisons with both general and state-of-the-art domain specific segmentation algorithms, and conclude with Section 5.4.

5.2 Spectral Graph Partitioning Subject to Stitching Constraints

We formulate the image segmentation problem as a constrained graph partitioning problem over a set of overlapping patches. Each patch is represented by a weighted graph, where nodes denote pixels and weights attached to edges connecting two nodes encode

grouping cues between the pixels. Segmenting small structures becomes a two-way node partitioning problem: pixels inside cells form a foreground node set, and those outside form the other background node set.

5.2.1 Segmentation with Stitching: Constrained Graph Partitioning

We first formulate the constrained spectral graph two-segmentation criterion ε [YS01] in terms of pairwise grouping cues encoding short-range (with-group) attraction A and long-range (between-group) repulsion R between background and cells:

$$\max \varepsilon = \frac{\text{within-group attraction}}{\text{total degree of attraction}} + \frac{\text{between-group repulsion}}{\text{total degree of repulsion}} \quad (5.1)$$

For each image patch I , we encode the grouping cues in an $n \times n$ weight matrix W , where n is the total number of pixels, to facilitate the foreground-background segmentation. Let:

$$W = A - R + D_R \quad (5.2)$$

$$D = D_A + D_R \quad (5.3)$$

where $D_M = \text{Diag}(M1_n)$ is the diagonal *degree* matrix for a $n \times n$ matrix M . Note that W could have both positive and negative weights. If we let X denote an $n \times 2$ binary partition matrix, where $X(i, g) = 1$ if pixel i belongs to group g , $g = 1, 2$, the above criterion can be formally written in matrix form as a two-way segmentation using the constrained normalized cuts [YS04, YS03]:

$$\begin{aligned} \text{maximize} \quad & \varepsilon(X) = \sum_{g=1}^2 \frac{X_g^T W X_g}{X_g^T D X_g} \end{aligned} \quad (5.4)$$

$$\text{subject to} \quad X \in \{0, 1\}^{n \times 2}, X1_2 = 1_n \quad (5.5)$$

$$U^T X = 0 \quad (5.6)$$

where 1_n denote an $n \times 1$ vector of 1's.

The near-global optimal solution is given by the eigenvectors of QPQ , where

$$P = D^{-1}W \quad (5.7)$$

$$Q = I - D^{-1}U(U^T D^{-1}U)^{-1}U^T. \quad (5.8)$$

While the eigensolution of QPQ takes longer than that of P (unconstrained version) to compute at each iteration, it often requires fewer iterations. We follow the eigensolution and its discretization procedures developed in [YS04, YS03] and their code online to obtain a binary segmentation.

We next need to address: **1)** How to compute local grouping cues A and R to achieve the desired segmentation; **2)** How to set up stitching constraints U between adjacent patches.

5.2.2 Local Features and Grouping Cues

To derive the attraction and repulsion cues, we start by extracting local features from the image. 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. 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. We characterize cells of small convex bright regions as the sinks of local gradient fields as in [BY10a]. Each pixel is associated with a peak direction vector p that indicates where pixels of higher intensity are located in its convex vicinity. Two pixels are attracted to the same region if their pixel-centric local gradient fields are similar, and repelled into different regions if they are of opposite types (e.g. sources and sinks). Computing the cues can therefore be divided into the following three steps.

Step 1: Consider pixel i and its neighborhood $N_d(i)$ of radius r_d (neighborhoods are taken as log-polar) as in Fig. 5.6a (a simple case of two separate dots, as can be appreciated by viewing the intensity profile in Fig. 5.6d). If neighbor $a \in N_d(i)$ can be reached in a

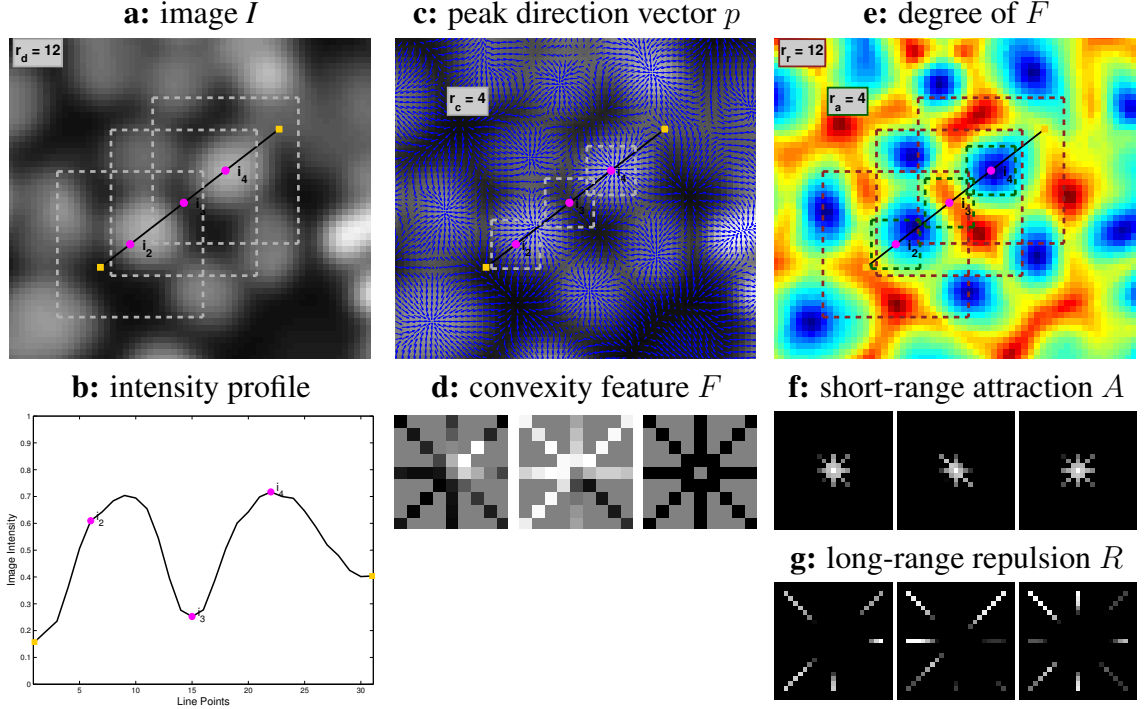


Figure 5.6: Computing pairwise features and grouping cues. **Step 1:** Given **a)** image I , with **b)** a sample intensity profile along the line intersecting a pair of separated dots, an initial radius r_d is used to compute **c)** a *peak direction vector* p at each graph node. This is achieved by taking the average of the direction from i to all its a neighbors, each weighted by the total non-decreasing intensity $T(i, a)$ along the straight line from i to a . **Step 2:** A convexity feature radius r_c is then used to compute **d)** the local *convexity feature vector* F , which characterizes where pixel i lies with respect to closest convex region. F can be visualized by looking at **e)** its degree at each i , i.e. $\sum_{a \in N_c(i)} F(i, a)$. Sinks of the flow (dot centers) result in negative values while sources are positive. **Step 3:** Finally, F is used to compute the pairwise cues: **f)** short-range attraction A and **g)** a long-range repulsion R between graph nodes within radii r_A and r_R of each other.

straight line from i with non-decreasing intensity, a is a higher intensity pixel in the same convex region. Let $p(i)$ be the average direction from its a neighbors, weighted by the total non-decreasing intensity $T(i, a)$ along the straight line from i to a :

$$p(i) \propto \sum_{a \in N_d(i)} T(i, a)(L(a) - L(i)), |p(i)| = 1 \quad (5.9)$$

$$T(i, a) = \sum_{\substack{I(m_1) \leq I(m_t) \leq \dots \leq I(m_k) \\ m_1 m_2 \dots m_k = \text{line}(i, a)}} I(m_t) \quad (5.10)$$

where $L(i)$ denotes the 2D location of pixel i in the image, $I(i)$ the intensity of pixel i , and $|\cdot|$ the L_2 norm of a vector. The peak direction vector $p(i)$ thus points from i towards the core of the cell that i belongs to, i.e., the highest intensity of its local convex region. It measures the direction and distance from pixel i to the center of the cell. $T(i, a) = 0$ if no ascending path exists on the specific line (resulting in $p = 0$ at the center of the dots, i.e. where sinks occur).

Step 2: We define the *convexity feature* vector $F(i, a)$ as the inner product of $p(i)$ and $p(a)$ within a *convexity* neighborhood $N_c(i)$, measuring how much a 's cell center estimate agrees with i 's. The ensemble of $\{F(i, a): a \in N_c(i)\}$ is a pixel-centric vector field (i.e. with the absolute direction of $p(i)$ factored out) that characterizes where pixel i is in the shape of a convex region, and we can use the feature similarity S to establish later pairwise pixel attraction and repulsion grouping cues:

$$F(i, a) = \langle p(i), p(a) \rangle, \quad a \in N_c(i) \quad (5.11)$$

$$S(i, j) = \frac{\langle F(i, :), F(j, :) \rangle}{|F(i, :)| \cdot |F(j, :)|}, \quad j \in N_A(i), N_R(i) \quad (5.12)$$

where N_A and N_R are the attraction and repulsion neighborhoods respectively. $F(i, :)$ shows how much i 's neighbors agree with i on the direction the dot lies in, with $p(i)$ itself factored out. Note that $p(i) \in \mathbb{R}^2$ while $F(i, :)$ is a $2r_c \times 2r_c$ vector. $S(i, j)$ is more likely to be positive for nearby pixels inside the same dot, and negative for distant pixels between different dots, giving rise to two kinds of grouping cues. In Fig. 5.6 we visualize F also by computing its degree at each pixel i , i.e. $\sum_{a \in N_c(i)} F(i, a)$, so that sinks (dot centers)

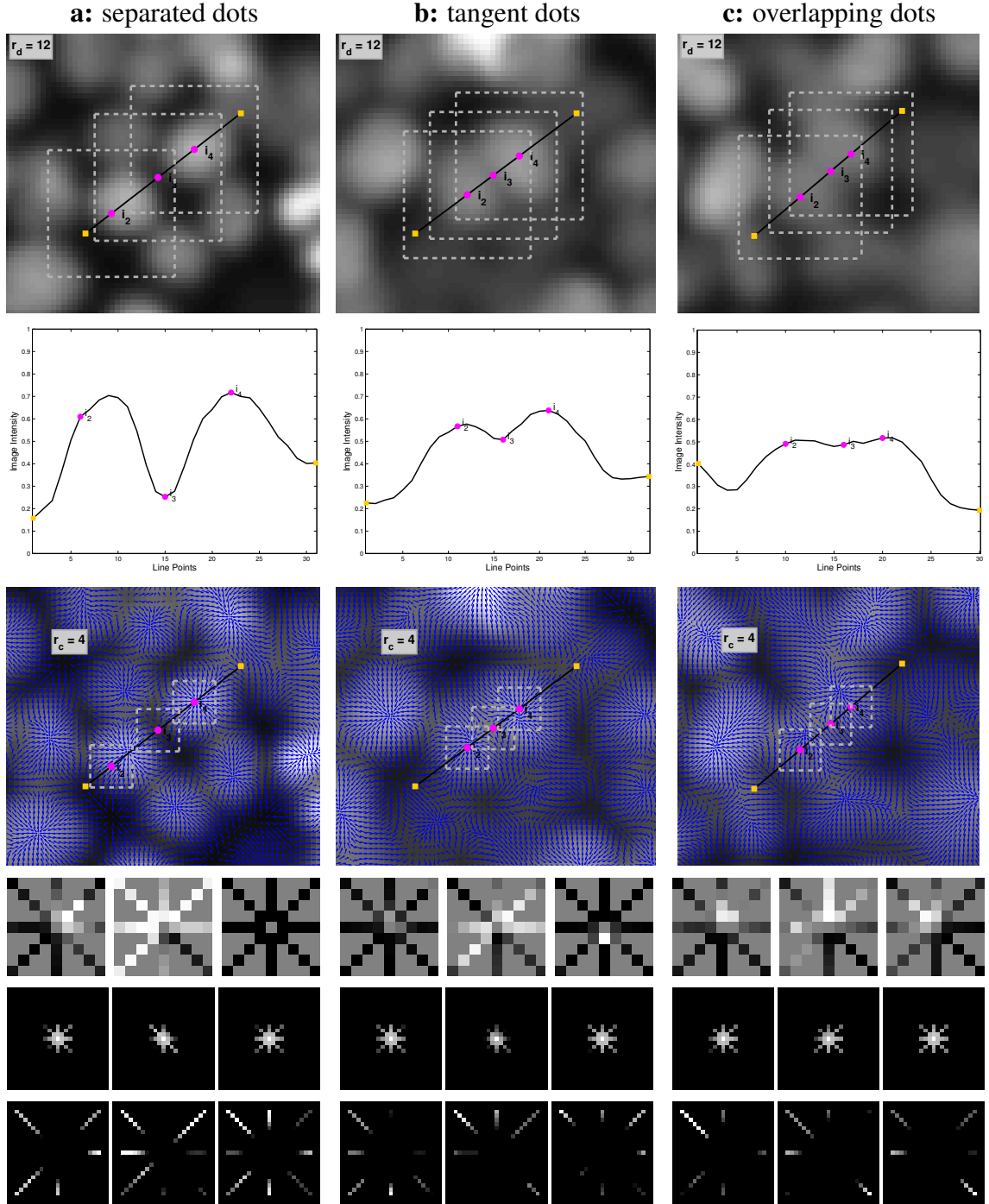


Figure 5.7: Local pairwise cues based on feature similarity for three typical scenarios. The initial image with intensity profiles (rows 1 and 2) of the line joining the two dot centers is used to compute the peak direction vectors (row 3). Three points (off-centered on the left dot, midpoint between the dots and on-center on the right dot) are selected to show feature vector F (row 4), attraction A (row 5) and repulsion R (row 6).

are characterized by negative values and sources by positive ones. We illustrate the sizes of attraction and repulsion neighborhoods N_A and N_R on this last image.

Step 3: Since only convexity cues are used to differentiate dots apart, pairwise grouping cues are based directly on the similarity feature measure S . While the attraction A only operates at short ranges, the repulsion R operates at long ranges pushing apart pixels inside dots from pixels outside and is essential for popping out disconnected regions [YS01, BY10a]. The short range attraction A and long-range repulsion R are then defined as:

$$A(i, j) = e^{-\frac{1-S(i, j)}{\sigma}}, \quad |L(j) - L(i)| \leq r_A \quad (5.13)$$

$$R(i, j) = \frac{1 - S(i, j)}{2}, \quad |L(j) - L(i)| \leq r_R \quad (5.14)$$

Figure 5.7 illustrates how A and R change as two dots become closer together.

5.2.3 Stitching Constraints U

A two-way node partitioning can be described by a $n \times 2$ binary partition matrix X , where n is the total number of pixels, $X(i, 1)$ and $X(i, 2)$ indicating whether pixel i belongs to the inside or outside of a cell.

Our stitching constraints are imposed on the partition indicator X that is to be solved in the optimization. If pixels a and b are known to belong in the same region, we have the constraint $X(a, :) = X(b, :)$, or $X(a, :) - X(b, :) = 0$. All these equations can be described in a linear constraint $U^T X = 0$, where $U(a, k) = 1$, $U(b, k) = -1$ is the k -th constraint that a and b belong to the same region.

The initial first-round patch segmentation does not require any constraints U , although simple intensity thresholding or initial seeds can be introduced. In the second-round patch segmentation, where each patch has been segmented, U comes from a mutual agreement analysis of X in the overlapping regions between neighboring patches: pixels for which two adjacent patches agree on the segmentation become either foreground or background

pixels. Only a sparse set of pairwise constraints are needed to ensure that two neighboring patches will have consistent segmentations in their overlapping areas.

Compared with traditional normalized cuts, the increase in time complexity is negligible if the number of constraints is small [YS04]. Additionally, the space complexity is reduced using patch segmentation with stitching constraints, as the image is broken down into smaller patches and finding numerous small regions becomes possible in a single two-way segmentation.

5.3 Experiments

We implement our algorithm in MATLAB and present results on different datasets of dot-like structures encountered in human pathology studies. The experimental section starts with a detailed analysis of parameter selection followed by details on the selection and effects of stitching constraints. We extend our model to thin and elongated structures to illustrate the benefits of the segmentation subject to stitching constraints beyond the simpler dot structures. We benchmark our method for the epithelial and embryonic cells against other commonly used segmentation algorithms. Finally, we also compare our dot segmentation method with state of the art domain-specific segmentation algorithms presented at the 2010 International Conference for Pattern Recognition (ICPR) as part of the Pattern Recognition in Histopathological Images contest for counting lymphocytes on histopathological images.

5.3.1 Segmentation Parameters

The setup of pairwise cues for convex region detection requires four radii, first to compute image features and then to compute pairwise cues:

r_d : radius of the neighborhood from which the peak direction vector p is computed.

Beyond a certain radius, r_d gives a constant p , so it can be chosen independently of the dot scale.

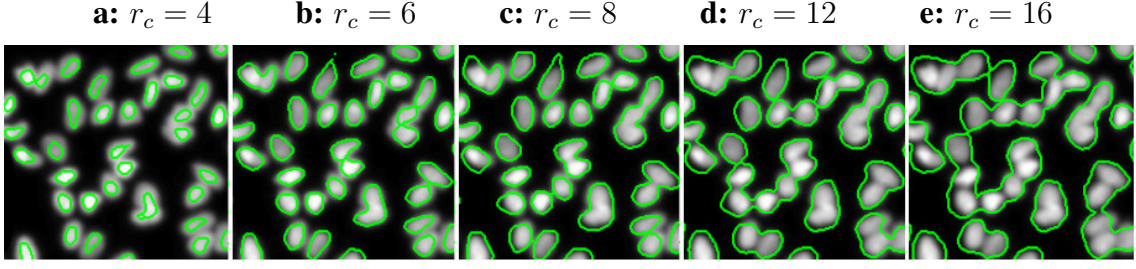


Figure 5.8: Finding larger dots by increasing the convexity feature radius r_c . As the attraction and repulsion radii remain fixed, we display the change of segmentation boundaries as a function of increasing r_c . This shows how choosing a smaller template allows detection of all dots regardless of their exact sizes with the same set of parameters. As the radius increases, the boundary expands to include the entire dot shape after which neighboring regions start merging together.

r_c : radius of the neighborhood used to compute the feature vector F , it encodes the local convexity of the neighborhood. Increasing r_c , while keeping all other parameters fixed, allows to look for larger shapes (Fig. 5.8). As r_c increases, the boundary expands to first include the entire dot and then to merge adjacent dots together.

r_A : radius used to compute short-range attraction cues. Increasing r_A brings nearby pixels together, so it should be at least comparable to the dot size.

r_R : radius that determines the extent of long-range repulsion cues. The absence of repulsion yields a segmentation similar to the traditional normalized cuts result shown in (Fig. 5.4e).

Intuitively, r_c has to entirely contain the dot to set a proper ‘template’ size for the feature vector F computation, while r_A and r_R have to be large enough to capture shape information and surrounding information respectively. r_c , r_A and r_R can be therefore defined in terms of a core radius r_0 that represents the size of the average dot. To compute the core radius, we use the heuristics depicted in Fig. 5.9. Starting with degree of the convexity feature F (illustrated in Fig. 5.6), we threshold the negative values to obtain disconnected components that represent possible dot regions and estimate the radius of each component.

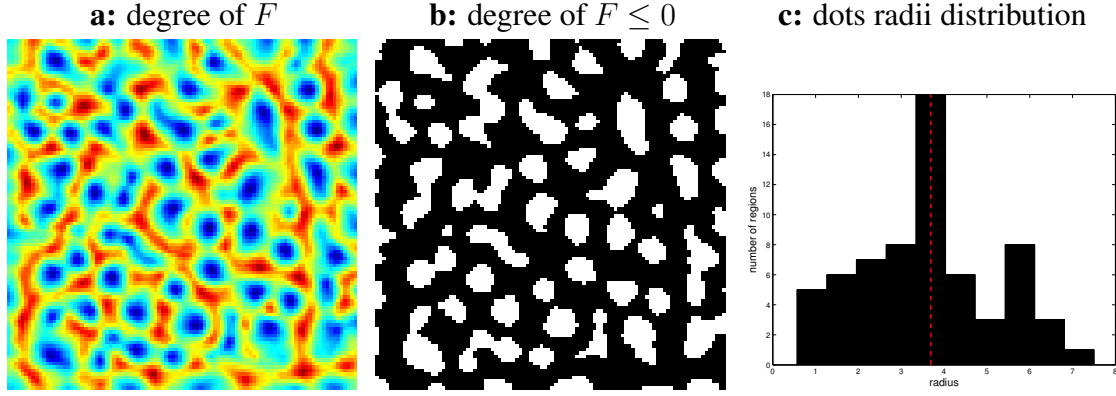


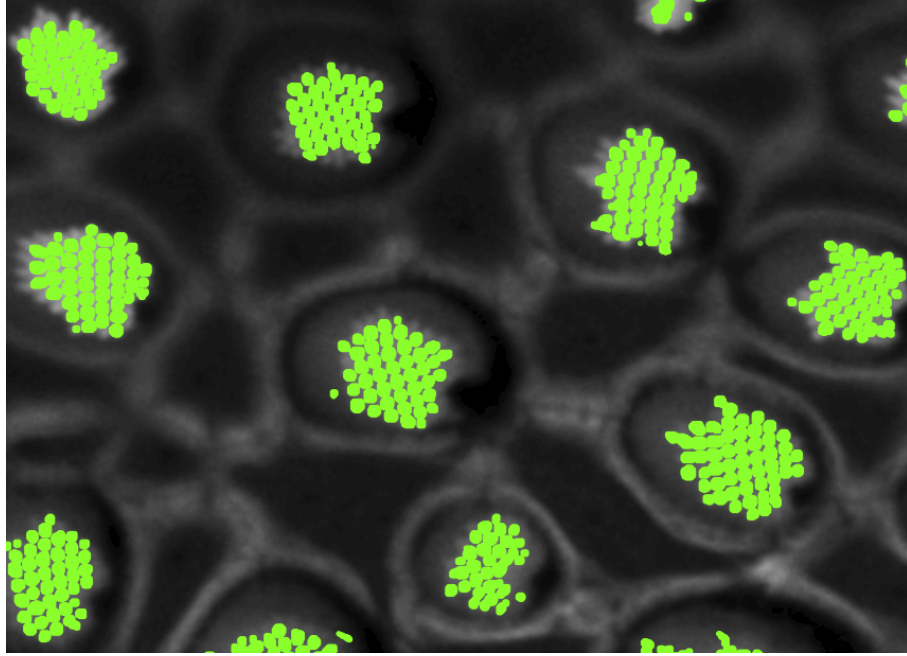
Figure 5.9: Automatic core radius estimation for parameter selection. Starting with **a**) the 2D visualization of the degree of the feature vector F , where negative values (blue) represent the sinks (dots) while positive ones (red) are background. We obtain **b**) disconnected components (i.e. possible dots candidates) by simply thresholding the negative values. After computing a radius estimate for each component, we look at **c**) the radii distributions for each patch (shown in the histogram) and we take the peak as possible core radius estimate (red dotted line).

We take the radius associated with peak of the radii distribution as core radius estimate, restricting the radius to be at least 4 pixels. The rest of the parameters can then be set accordingly: $r_c = \max(4, 1.5r_0)$, $r_a = \max(4, r_0)$ and $r_R = \max(20, 3r_0)$. In special case scenarios, such as the hierarchical dot phenomenon illustrated in Fig. 5.10, parameters have to be manually tuned to extract the desired structures.

5.3.2 Constraint Propagation and Elongated Structures

Segmentation subject to stitching constraints is not simple stitching. The constraints set on the overlapping regions propagate to the interior regions and are able to correct improper initial segmentation, as shown in cyan in Fig. 5.11. A few constraints are sufficient to correct mistakes, such as two cells erroneously segmented as one, spurious segment cleanup and boundary refinement. We choose an overlap size that is able to contain at least one entire dot in order to be informative. For all results, we use 20 pixels of overlap and patches of approximately 256×256 pixels. Clearly, larger overlap would allow more constraints between cells but one must take into account the tradeoff of having to compute

a: segmenting individual cilia: $r_c = 4, r_R = 20, r_A = 4$



b: segmenting haircell bundles: $r_c = 30, r_R = 40, r_A = 10$

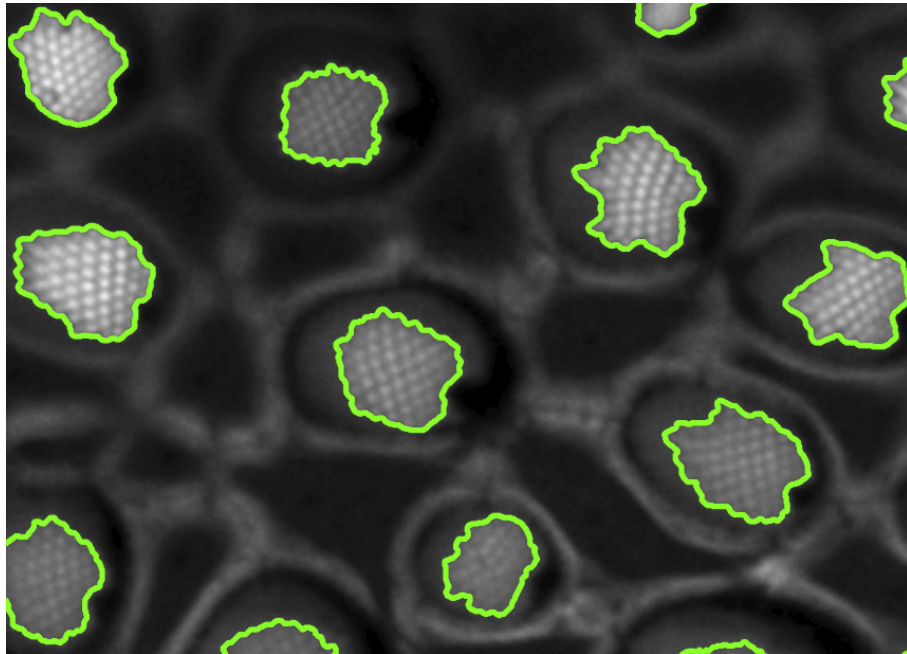


Figure 5.10: Hierarchical dot structures and radii parameters. Changing the template size r_c is not sufficient when looking for significantly larger objects. In order to add proper repulsion between the regions, the overall radius r_R must also be increased. In **a**) we have results with the standard set of parameters and in **b**) with increased repulsion and template radii.

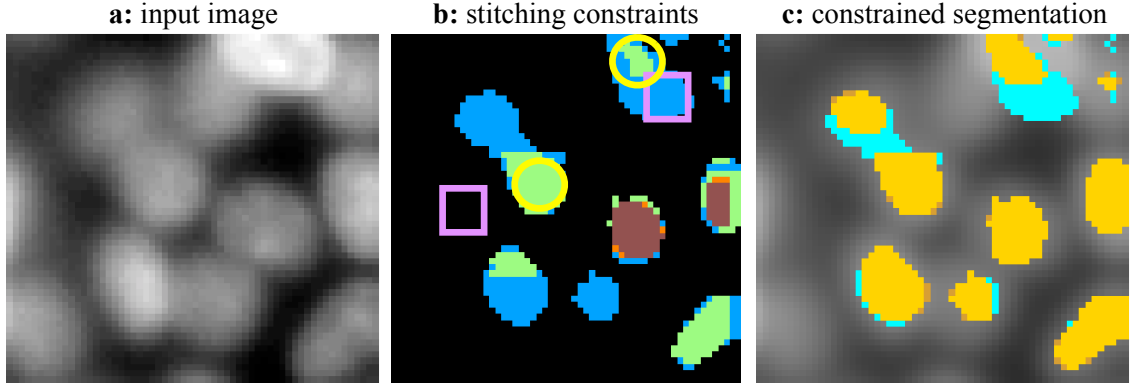


Figure 5.11: Beyond simple stitching. **b)** Constraints are derived from mutual agreement analysis on adjacent patch segmentations from a previous round (blue, green, or orange/maroon indicate 1, 2 or more patches agreement). Stitching constrains cells (circles) and background (squares) to group together respectively. **c)** Constrained segmentation (gold) improves the initial segmentation (cyan) by cleaning up spurious small regions, separating conjoined cells, and refining their boundaries.

more patches and the complexity increase due to increasing the number of constraints.

Recalling the EM data from the medulla brain region of the drosophila fly from Fig. 5.1, there is often also a need to segment small and thin structures. The strength of our method is that it can be extended to a larger set of geometries. The same visual popout applies, since salient regions have a common repelling background, but in order to extract the more complicated geometries, new attraction and repulsion cues have to be defined.

To avoid breaking each line into many small convex regions, we add an additional pairwise component $\tilde{S}(i, j) = \exp(-\frac{1}{\sigma}(\Phi(i) - \Phi(j))^2)$ based on the degree $\Phi(i) = \sum_{a \in N_c(i)} F(i, a)$ of the feature vector F , introduced in Section 5.2.2. The pairwise attraction and repulsion cues are now a function of:

$$T(i, j) = \beta S(i, j) + (1 - \beta) \tilde{S}(i, j) \quad (5.15)$$

where β a constant parameter and $S(i, j)$ is the similarity feature measure from Eqn. 5.12.

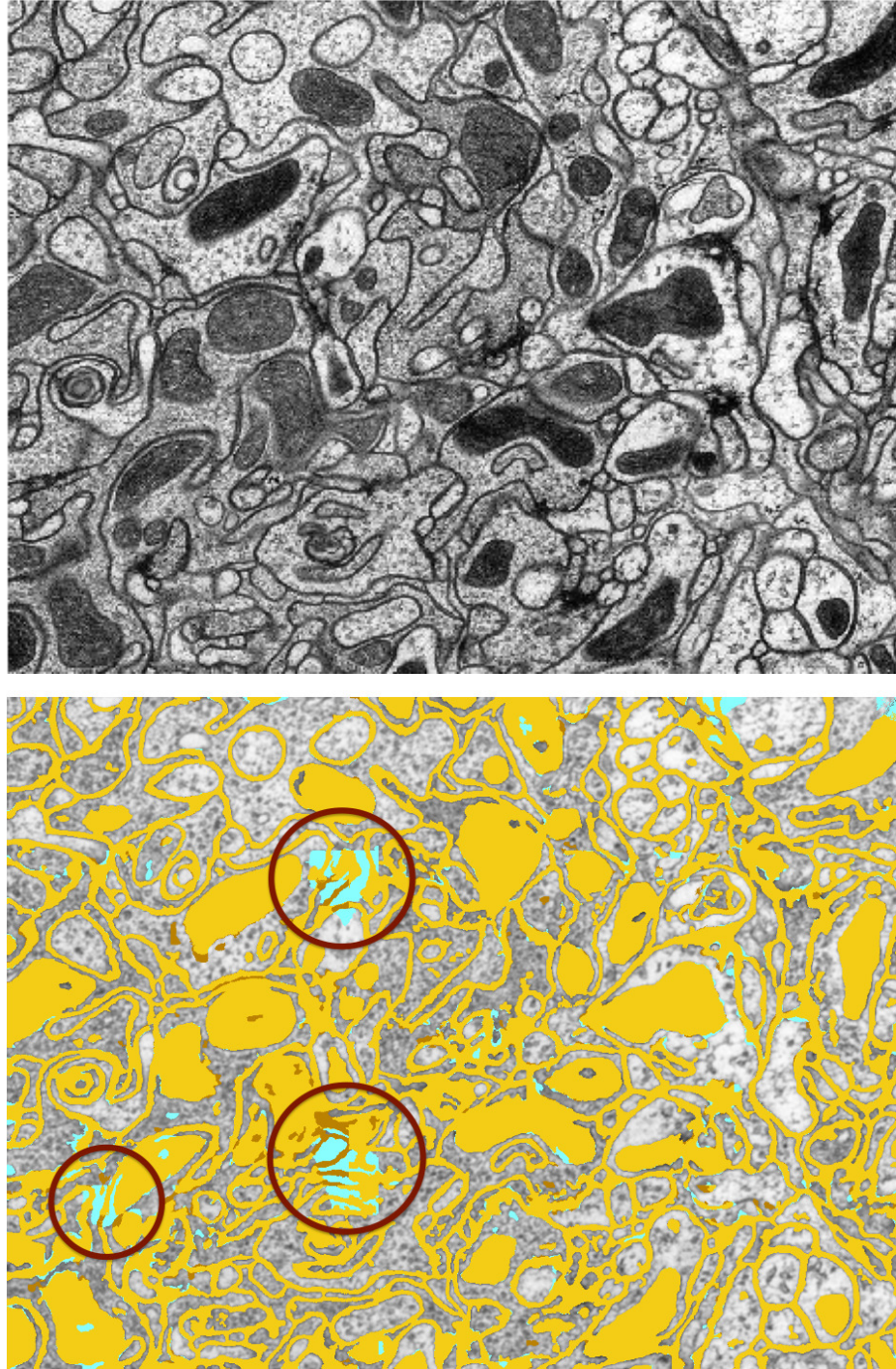


Figure 5.12: Thin structures and constraint propagation. The initial segmentation (cyan) is corrected by the constrained segmentation (brown). The final constrained segmentation (gold and brown) is more than simple stitching. Its effectiveness can be better seen for membrane structures that have a geometry that inherently propagates throughout the image to interior regions which are farther away and not located within the overlapping regions.

The pairwise attraction and repulsion cues are then defined as:

$$R(i, j) = |\alpha - T(i, j)|, \quad |L(j) - L(i)| \leq r_A, T < \alpha \quad (5.16)$$

$$A(i, j) = |T(i, j) - \alpha|, \quad |L(j) - L(i)| \leq r_R, T \geq \alpha. \quad (5.17)$$

For the images shown, we empirically chose parameters $\sigma = 0.25$, $\beta = 0.35$ and $\alpha = 0.35$. As before, attraction A only operates at short ranges and repulsion R operates at long ranges, to give a total effective weight W is $A - R + D_R$.

The benefits of the stitching are highlighted in Fig. 5.12. The initial (cyan) segmentation is improved after the constraints are enforced to obtain a final (gold) segmentation that corrects segmentation mistakes also outside the overlapping regions. The yellow segmentation denotes where the segmentations, before and after the constraints, agree. The elongated structures then allow the stitching benefits to be more pronounced as the constraints propagate within the regions to the interior of the image. The final results for the original 1800×1800 pixel image, obtained using a 8×8 stitching grid, are shown in Fig. 5.19. In Fig. ??, a post-processing step, dividing regions with larger diameters, is added to discriminate between membranes (gold) and all the other structures (blue).

5.3.3 Benchmark with General Segmentation Methods

To compare our results with other commonly used segmentation algorithms, we run our method on images (a selection was given in Fig. ??) from two datasets of dot-like structures encountered in human pathology: 1) 512×512 pixels bright field and fluorescent images of epithelial and embryonic cells; and 2) 600×900 pixels histopathology images from various tumor-related lesions. Figures 5.15, 5.16, 5.17 and 5.18 illustrate sample results of our segmentation method.

We benchmark our 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

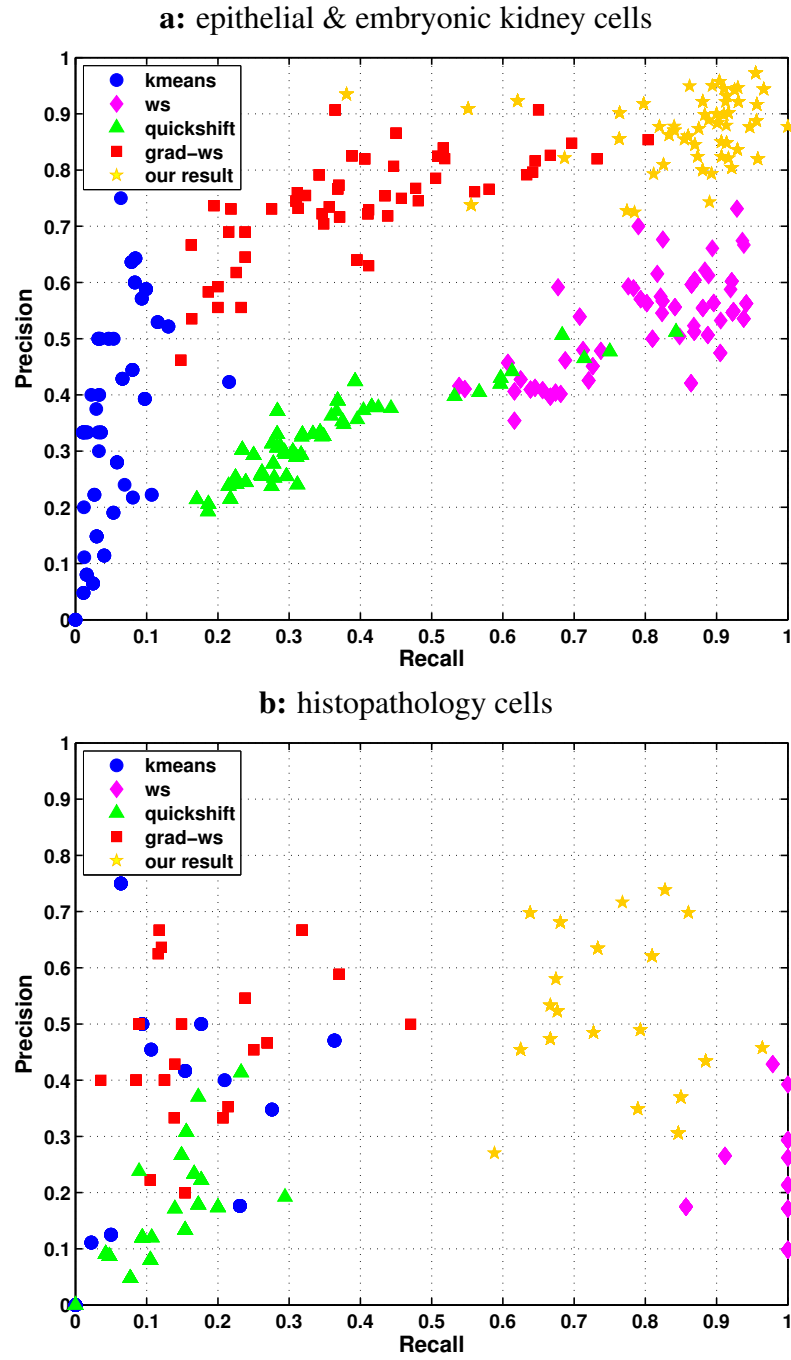


Figure 5.13: Precision-recall statistics for **a)** epithelial cells images and **b)** histopathology images. Our method (yellow square dots, upper right corner) has better precision and recall overall when compared with quickshift, k-means, gradient-based and standard watershed.

matched detection. We then define:

$$\begin{aligned} \text{precision} &= \frac{\#\{j : \min_{i=1}^m D_{ij} \leq \rho\}}{n} &= \frac{\# \text{ nearest dots within radius } \rho}{\# \text{ segments}} \\ \text{recall} &= \frac{\#\{i : \min_{j=1}^n D_{ij} \leq \rho\}}{m} &= \frac{\# \text{ nearest segments within radius } \rho}{\# \text{ dots}} \end{aligned}$$

The precision measures the proportion of true dots among all the segments, and the recall measures the proportion of segments among all the true dots. We compare our method with the following segmentation methods:

k-means: We use MATLAB's built-in function. It clusters pixels based on their intensity values, thus has trouble separating conjoined like-intensity cells and increasing k only leads to clustering instability.

Watersheds: Watershed is directly applied to either the intensity image (MATLAB's built-in function) or the gradient magnitudes (with radius 5) of the image. While the standard watershed results tend to be over-fragmented in the presence of local intensity fluctuation, the gradient-based watershed results tend to miss many dots of weak contrast but improves on the precision.

Quickshift: We use the online code by [VF10]. Analogous to meanshift, quickshift can be used to partition an image into a set of superpixels. It enhances intensity differences, but it is sensitive to scale choices and cannot break up dots based on convexity.

The precision-recall statistics in Fig. 5.13 shows that our method works better than others at segmenting small regions in terms of both precision and recall. Our stitching constraints can be appreciated by comparing the quality of segmentation without and with constraints: While there is no significant improvement in the recall, there is an average improvement of 0.04 in the precision.

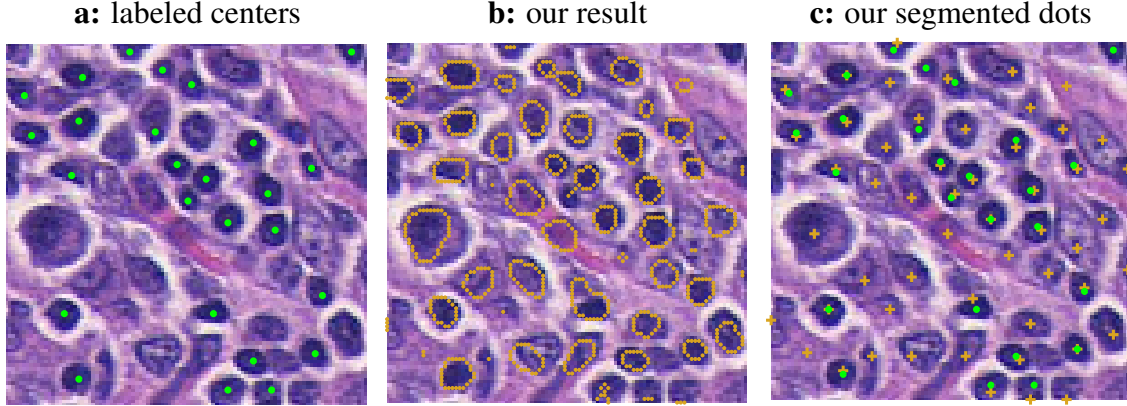


Figure 5.14: Sample results from the ‘Counting Lymphocytes on Histopathology Images’ contest dataset [FBX⁺09, BGA⁺10], courtesy of Anant Madabhushi at Rutgers. **a)** Labeled ground-truth was provided for comparison. **b)** Our results: boundaries of our computed segments and **c)** the centers (gold pluses) extracted from them together with the ground truth (green dots). Performance details are given in Table 5.1. We display our results on the inverse images, for clearer display. As in the contest, the goodness of the boundaries was not measured as specific boundaries cannot be identified to begin with. Our algorithm is not domain specific, which explains why we pick up many more dots that are not identified by experts as lymphocytes. A post-processing step, for example based on color, can be applied to refine the results.

5.3.4 Benchmark with Domain Specific Methods

We also compare our dot segmentation method with state of the art domain-specific segmentation algorithms presented at the 2010 International Conference for Pattern Recognition (ICPR) as part of the Pattern Recognition in Histopathological Images contest for counting lymphocytes on histopathological images. For this benchmark we use an additional set of 100×100 pixels histopathology images, very kindly provided by the contest organizers. Given the 100×100 pixels dimensions, we did not divide the images and simply ran the segmentation once. Each image took approximately 2 seconds to segment. Sample results are shown in Fig. 5.14. Finally, we compare performances in Table 5.1 with the algorithms presented at the contest:

Method 1: [KSG10] based on extracellular matrix segmentation, i.e. mean shift clustering for color approximation followed by hsv space thresholding. Texture features

Group	m_d	s_d	m_N	s_N
[KSG10]	3.04	3.40	14.01	4.4
[PRT10]	2.87	3.80	14.23	6.3
[GGP10]	7.60	6.30	24.50	16.2
[CVR10]	8.10	6.98	26.67	12.5
our results ($\rho = 5$)	3.22	3.92	5.40	3.68
our results ($\rho = 4$)	2.84	2.89	8.20	4.75
our results ($\rho = 3$)	2.20	1.80	12.90	5.38
our results ($\rho = 2$)	1.12	0.71	16.75	7.47

Table 5.1: Our results compared with the finalists of the PR in HIMA (ICPR 2010) contest performance ranking provided online, for the ‘Counting Lymphocytes on Histopathology Images’ dataset. The ranking of performance, where m and s denote the mean and standard deviation, respectively. For all numbers, smaller numbers represent a better result. The criteria are (a) the Euclidean distance d between the ground truth and the result provided by the participants; (b) the absolute difference between the true number of cells and N the number of cells found. We show our results for tolerances $\rho = 2, 3, 4, 5$ pixels from the center. Our results are very intuitive. Decreasing the tolerance allows to find less true dots (higher m_N) but with higher precision (smaller d). For each choice of radii, we perform well when compared with the other proposed methods.

are extracted from the cells and then used to train a SVM classifier to find the lymphocytes.

Method 2: [GGP10] based on connected components. A first processing step involves thresholding and morphological operators to improve the quality of the images for recognition; a second recognition step then extracts the lymphocytes with a template matching method.

Method 3: [CVR10] starts with a segmentation based on multi-phase level sets, followed by morphological operations to clean-up the image of small spurious regions. Features are then used to identify the target cells.

Method 4: [PRT10] based on the estimation of a mixture of Gaussians for determining the probability distribution of the principal image component. Lymphocyte are detected after post-processing to eliminate small regions, using a transferable belief model for knowledge modeling.

The criteria for evaluation are (a) the Euclidean distance between the ground truth and the segmented lymphocytes; (b) the absolute difference between the true number of cells and the number of cells found, denoted by N . m and s denote the mean and standard deviation, respectively. As before, we consider (i, j) a matched detection if the segment center is within a ρ threshold from the ground-truth dot (we use tolerance radii $\rho = 2, 3, 4, 5$ pixels). For all numbers, smaller numbers represent a better result. We are not able to distinguish lymphocytes from the other dot structures present because our method is not domain-specific. For fairness we stress that we did not have different training and testing sets. We used one image to test our automatic core radius estimator and then ran the algorithm on the entire image set. For each tolerance radius, we perform well when compared with the other proposed methods. As intuitively expected, decreasing the tolerance allows to find less true dots (higher m_N) but with higher precision (smaller d) and vice-versa, so the threshold can be fixed according to desired balance.

5.4 Summary

Segmenting small structures in a large image presents a scale dilemma between the image size and the segment size. Our approach resolves this by decoupling the two sizes in constrained patch segmentations. Although segmentation subject to stitching constraints could work with any patch segmenter, a spectral graph partitioning formulation naturally integrates stitching constraints together with the foreground/background patch segmentation. The experimental section features a detailed parameter analysis to segment dots of different shapes and sizes, and illustrations of the different cases that arise in segmenting dots. We benchmark our results with general segmentation algorithms and we also compare our method with domain-specific algorithms presented at the ‘Counting Lymphocytes in Histopathology Images’ contest at ICPR 2010. Our method outperforms state-of-the-art results in both precision and recall by identifying all faint and conjoined cells simultaneously in a two-way segmentation, without need for post-processing.

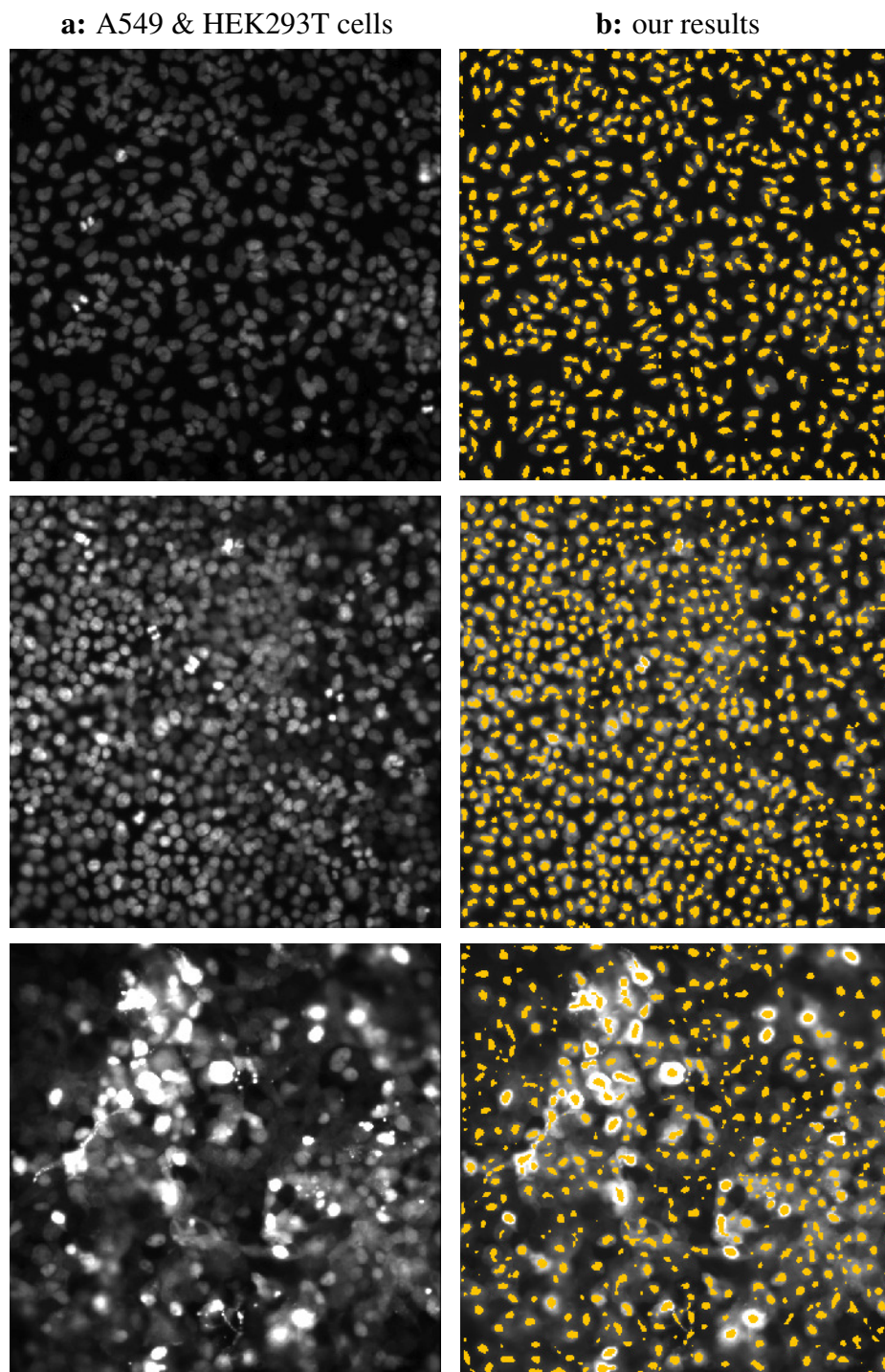


Figure 5.15: Results (b) on human alveolar basal epithelial A549 cells and embryonic kidney HEK293T cells (a). A quantitative measure of our segmentation is given in Fig. 5.13.

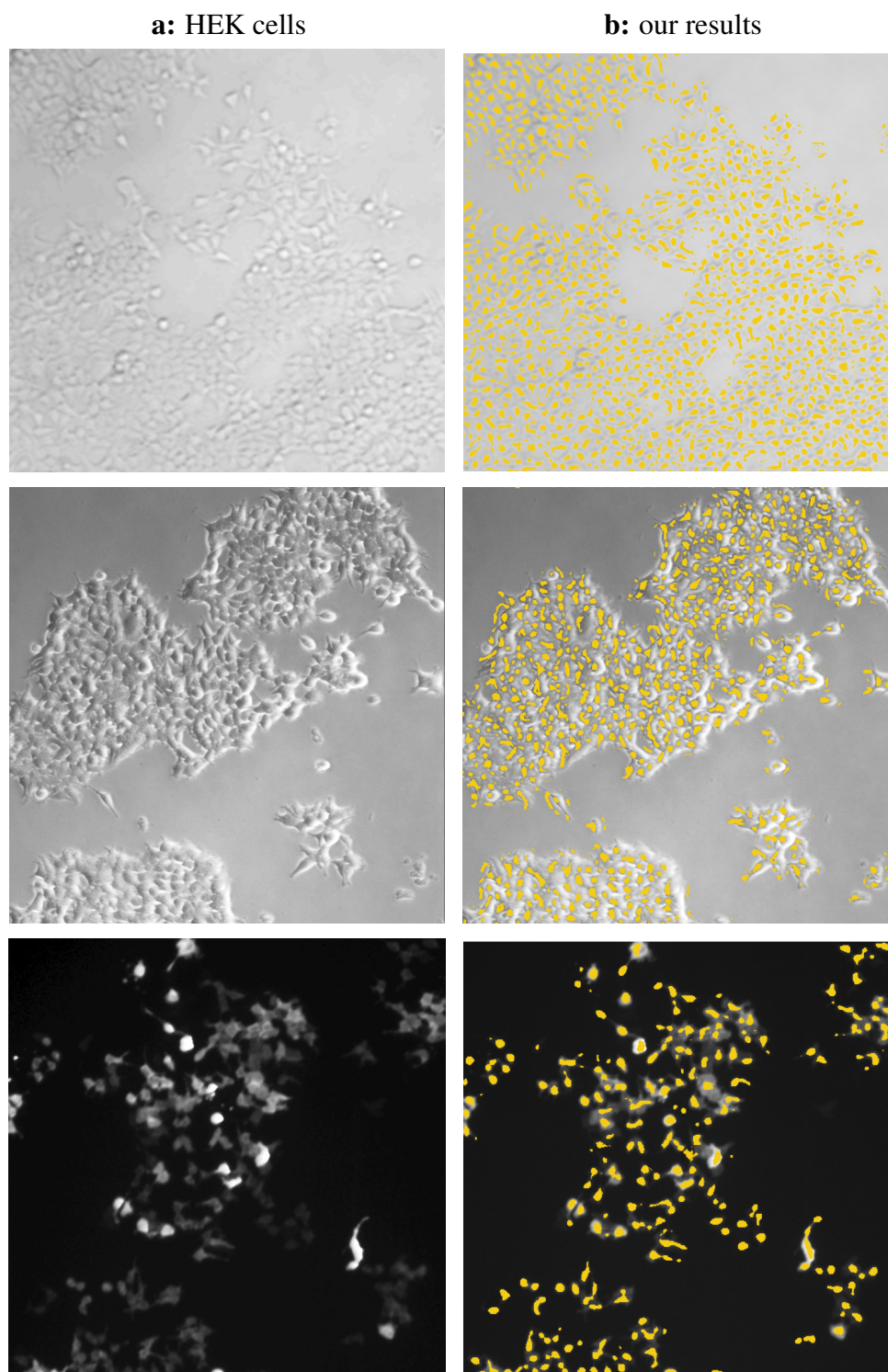


Figure 5.16: Results (b) on HEK cells (a) with a variety of ‘convex shapes’. Our method pops out all the cells in these images with the same parameters. Minimal post-processing with morphological operations allows to mask out the ‘flat’ background regions.

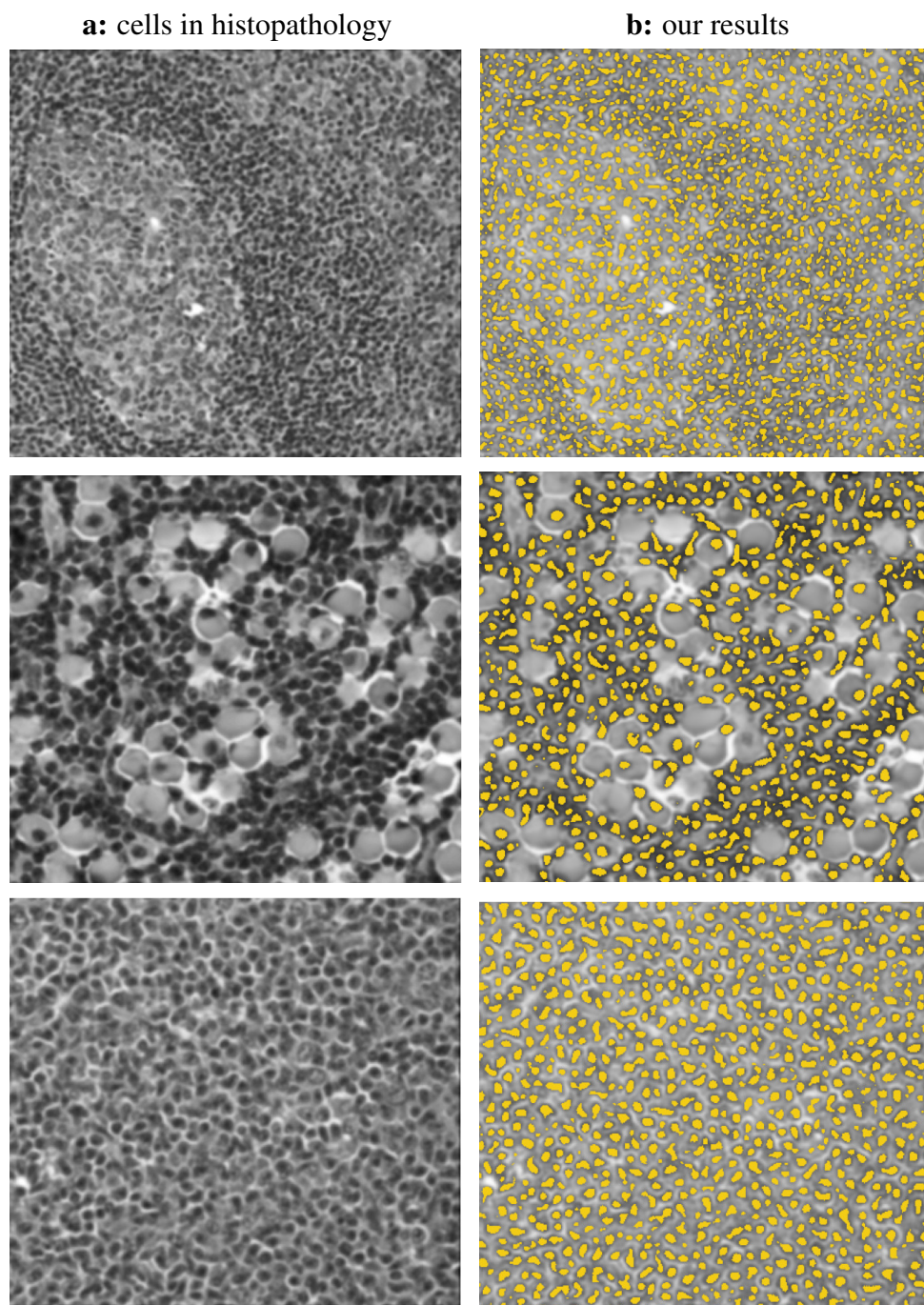


Figure 5.17: Results (b) on histopathology images (a) depicting tumor-like lesions in the spleen (top,bottom) and in mediastinum (center) containing ‘dots’ of various sizes and textures. Our method pops out all the cells in these images with the same parameters and no post-processing.

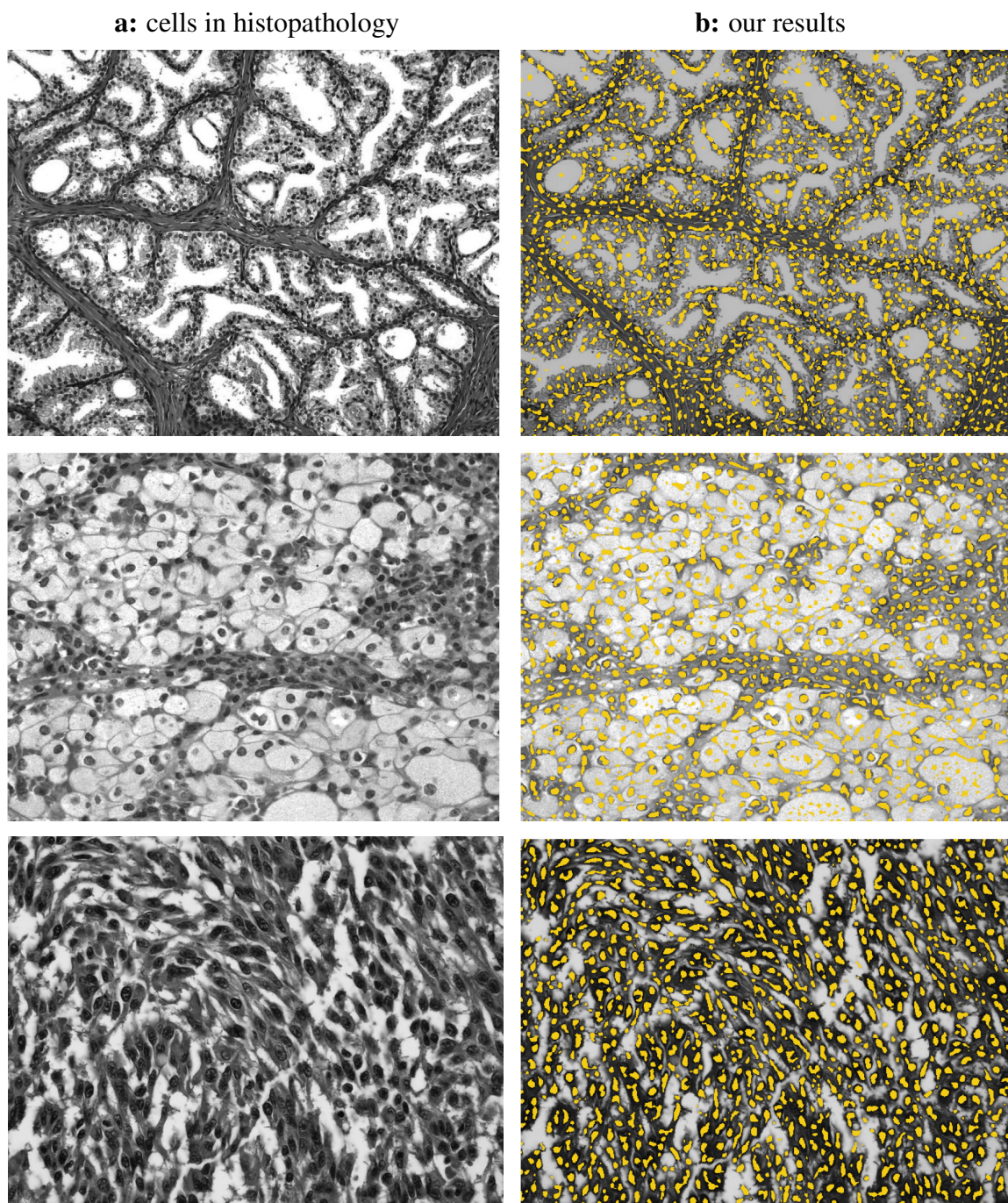


Figure 5.18: Results (b) on histopathology images (a) depicting tumor-like lesions in the ovaries (top), spleen (center) and prostate (bottom) containing ‘dots’ within a variety of background structures. Our method pops out all the cells in these images with the same parameters and no post-processing. Images were darkened for final result display only.

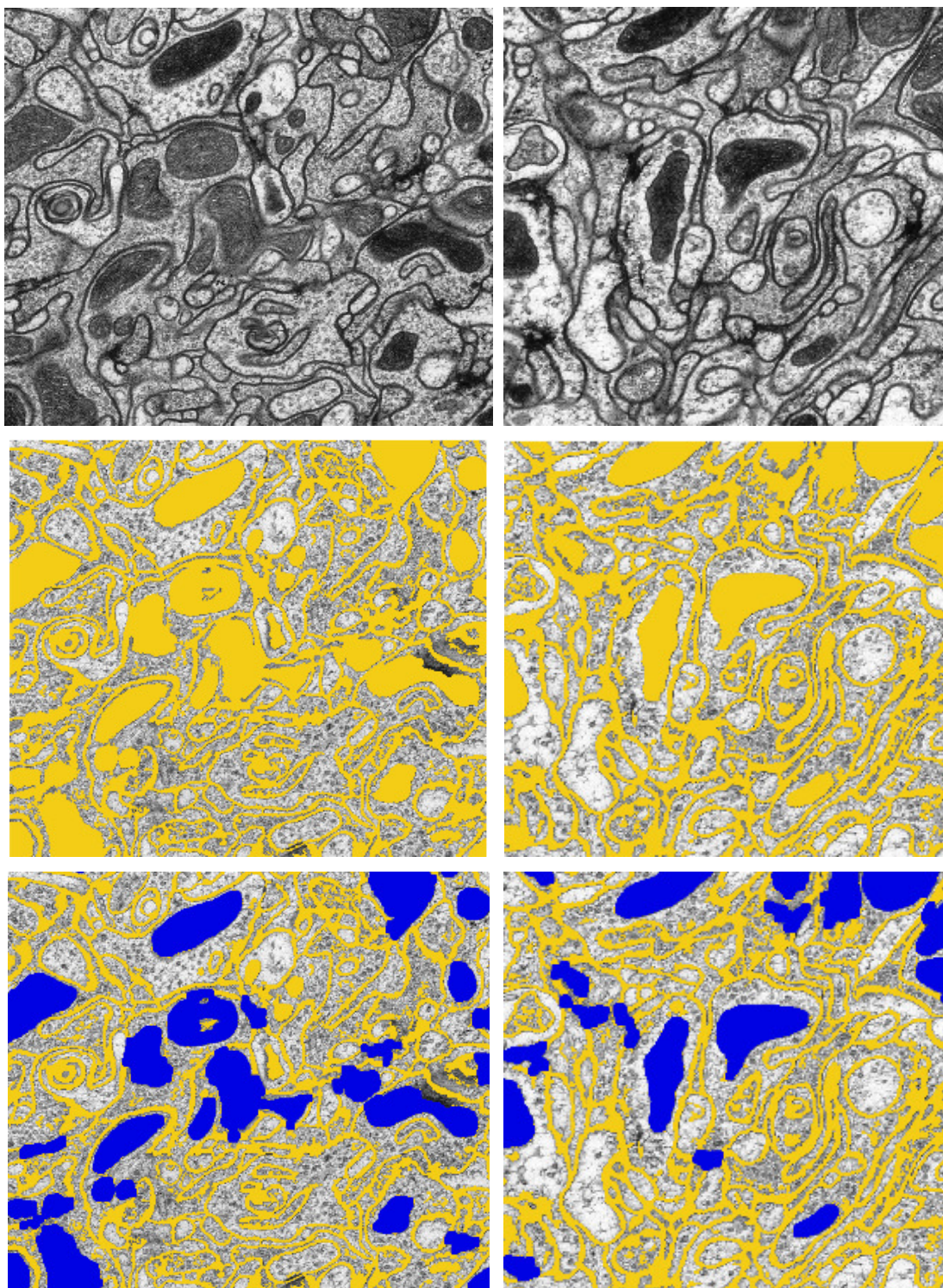


Figure 5.19: Results on EM brain sections of drosophila fly. Extending the weights beyond the 'dots' structures shows the potential of our method to extract a wider shape variety. Displayed are 900×900 cutouts of the original 1800×1800 pixels image. Post-processing with a simple sequence of morphological operations yields (bottom row) thin membranes in gold and mitochondria in blue.

Chapter 6

Shape Extraction through Region-Contour Stitching

6.1 Introduction

We consider the problem of extracting salient shapes from an image, which is an important part of any image analysis. This problem has been studied in the context of region segmentation [SM00, MAFM08], contour grouping [TZ06, US88, MWTX03, FB03, GM96], as well as a combination of them [WKS05, Jac96]. For many applications, the key remaining challenge is detecting faint contours and boundaries along low contrast regions.

Traditional approaches that rely only on local edge detection in order to detect all faint contours boundaries have the side effect of generating many additional spurious edges. On the one hand, contour grouping techniques, such as edge linking [GZW07] and more recent methods such as untangling cycles [ZSS07], are often used as a way to prune out spurious edges and complete missing ones. The downside is that these methods depend on fragile local cues, and fail to take into account the global information provided by the regions along the boundary. On the other hand, region segmentation such as Normalized Cuts (Ncut) [SM00] has the ability of cleaning up spurious edges and bridging gaps among

missing or faint contours. However, in order to find all the salient segments, oversegmentation is needed which leads to fragmentation along the salient shape boundary.

The intuition behind our method, as shown in Figure 6.1, is to combine the Ncut segmentation of an image (shown in Fig. 6.2) with a *flow* on its edge map [BS08]. There are two key insights: (1) region information provides edge boundaries with more global information of how they should be connecting with each other. From this more robust flow information, one can extract salient contours more easily even under low contrast and clutter; (2) we only use soft segmentation eigenvectors, which capture the likelihood of region segmentation. This allows us to retain edges which do not line up with the segmentation boundaries and could be washed out in the discretized segmentation. In these cases, the contours extract the low contrast region information from the soft eigenvectors.

6.2 Graph Setup

6.2.1 From Filter Responses to Edge Mask

Motivated by human vision, we define the image gradient using *oriented edge* energy:

$$OE_{\Phi,\alpha} = (I * f_{\Phi,\alpha}^e)^2 + (I * f_{\Phi,\alpha}^o)^2, \quad (6.1)$$

where $f_{\Phi,\alpha}^e, f_{\Phi,\alpha}^o$, are a quadrature pair of even and odd symmetric filters at orientation Φ and scale α .

Most algorithms detect edges by applying non-maximum suppression on $OE_{\Phi,\alpha}$ across different orientations Φ to obtain edge orientation Φ^{max} , and then localize the edge by checking zero-crossing in $f_{\Phi^{max},\alpha}^e$. Such edge detection procedure often destroys edges around junctions. At the junctions, the boundary pixels could have multiple orientations. Forcing them to make a hard choice on a single edge orientation leads to erroneous orientation estimates. We compute the edge zero crossings for each oriented $f_{\Phi,\alpha}^e$. To remove spurious edge detections (those that extend from the boundary into the surrounding flat image region), we apply non-maximal suppression across the orientations allowing only

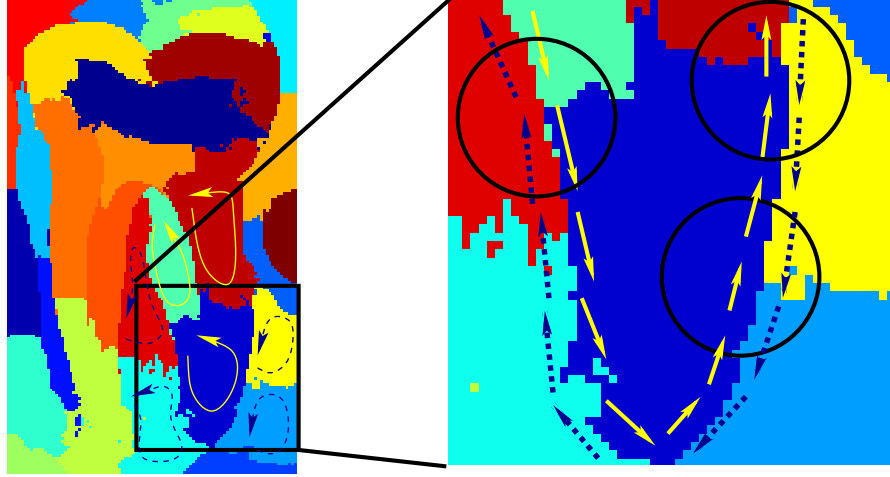


Figure 6.1: The intuition behind our method: each segmentation region induces directional flows on the region boundaries. Right: each image pixel could have multiple induced flow directions, one for each nearby segmentation region. Contour grouping using the induced edge flow results in the stitching of fragmented segmentation regions.

pixels NOT on a zero crossing to suppress their neighbors. We initialize the edge mask \mathcal{E} to include the surviving zero crossings.

6.2.2 Region Segmentation using Normalized Cuts

We use the NCut [SM00] graph partitioning setup to extract image region information. The set of points in the image space are represented as a weighted undirected graph $G^{region} = \langle V^{region}, W^{region} \rangle$, where the nodes V^{region} of the graph are the image pixels and the W^{region} is a similarity function between pairs of nodes computed by Intervening Contours [MBLS01] directly using $f_{\Phi, \alpha}^e, f_{\Phi, \alpha}^o$. The weight matrix W^{region} is used to compute the NCut soft eigenvectors which will be denoted by $\{v^{(1)}, \dots, v^{(n)}\}$. Each eigenvector can be seen as an image itself, with the brightest pixels corresponding to the hard segmentation region labeling $\{S^{(1)}, \dots, S^{(n)}\}$.

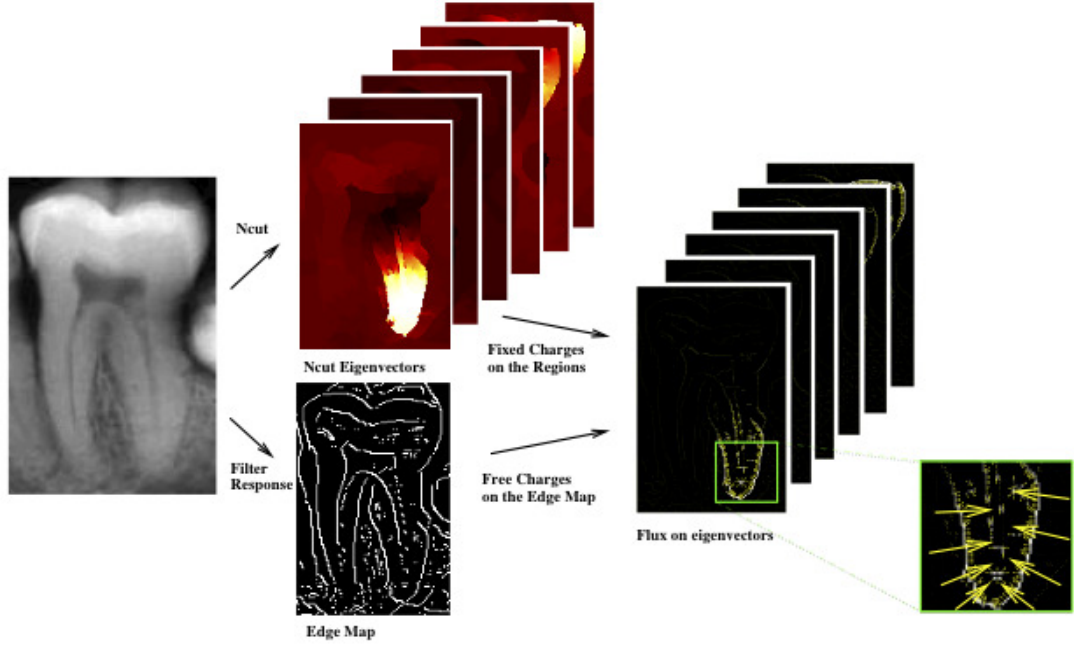


Figure 6.2: Region Segmentation and Lorentz force on edges. We compute the Ncut image segmentation eigenvectors to obtain a set of soft segmentation maps, one for each region. Using the soft segmentation map as negative particle charges, we compute the force on the initial image edges. Its direction encodes the local shape of the segmentation region, while the magnitude reflects the saliency of the image region, containing more global information than local image contrast.

6.2.3 From Region Segmentation to Lorentz Edge Flow on the Edge Mask

Our method contrasts with previous contour grouping approaches in the sense that the magnitudes and orientations associated to each edge point are not given by the local geometrical properties of the contour itself, but rather they are induced from more global information computed by using both the edge mask \mathcal{E} and the region segmentation soft eigenvectors $\{v^{(1)}, \dots, v^{(n)}\}$ with corresponding labels $\{S^{(1)}, \dots, S^{(n)}\}$.

Jalba et al. [JWR04] introduced the charged particle model to recover shapes by modeling the image as a charged electric field. In their model, the free charges in the image

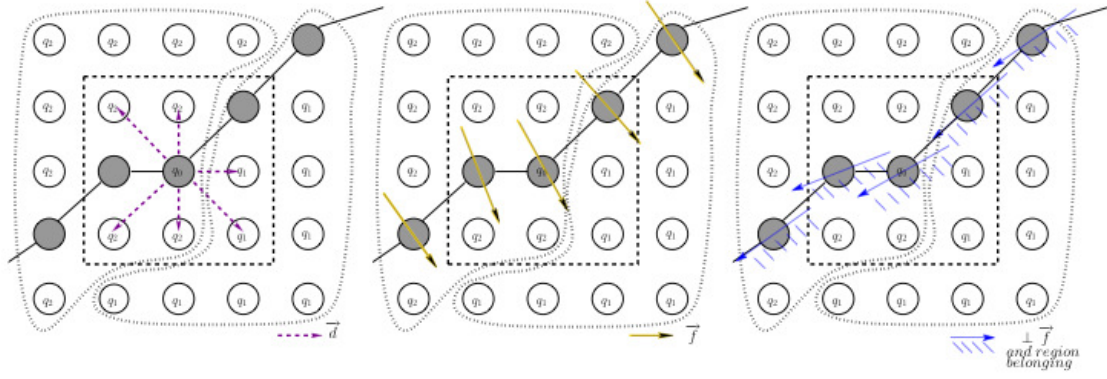


Figure 6.3: Computation of the Lorentz force F_l from each of the soft region segmentation eigenvector. Left: Initial image edge detections superimposed on the segmentation eigenvector (resulting in the free particles p_i shown with darker color). Dotted lines indicate the region boundary, white circles the fixed region particles e_i and d denotes the distance vector $r_i - R_k$. Computation of F_l tolerates imprecision in the region segmentation boundary. Center: an example of resulting Lorentz force F_l . Right: The corresponding directed Lorentz flow after a 90 degree rotation. The flow also encodes contour-region membership (indicated by the one sided edge flow).

regions are attracted to the contours by an electric field in which image pixels are assigned charges based on gradient magnitude of the image. In this work, the exact contours are not known a priori and finding the contours, viewed as static entities, relies on the repulsive forces within similar regions. Inspired by this, we invert the setup and exploit the initial region segmentation information to fix the electrical particles within the regions instead and let the electric field defined on them create a flow on the previously found edge map. For each eigenvector $v^{(i)}$ we view the N image pixels as charged particles with corresponding negative electric charges q_i given by the pixels' negative eigenvector magnitude value. To each eigenvector, we superimpose the initial edge map \mathcal{E} and we keep as fixed particles e_k only pixels that do not lie on the edges, the edge points instead will be considered as free particles p_i . It is important to note that the contours found from the region segmentation and the edge map contours do not need to coincide (as is in the example illustrated in Fig. 6.3) and in fact the purpose of combining them is to allow one to correct the mistakes of the other.

For each free particle p_i (which will later correspond to nodes of our graph), with position vector r_i , we are only interested in the Lorentz force F due to the electric field generated by the fixed charges e_k as this force captures the external attractive interactions. Following the simplifications in [JWR04] we omit the magnetic field. The Lorentz force F_l at each particle p_i can then be written as:

$$F_l^{p_i} = q_i \sum_{k: R_k \neq r_i}^M \frac{e_k}{4\pi\epsilon_0} \frac{r_i - R_k}{\|r_i - R_k\|^3}, \quad (6.2)$$

where the summation is over the M fixed charges e_k with grid vector positions R_k that fall within a fixed radius of the free charge p_i , and ϵ_0 is the electrical permittivity of free space. Note that the summation only accounts for interaction from the fixed charges and does not consider forces generated by all other free particles taken in isolation.

We use the Lorentz force F_l induced from the soft region segmentation to compute a *flow* on the initial image edges. Each contour point in the original edge mask can have up to n copies (n being the number of NCut segments) in the line graph nodes $V^{contour}$ explained in the next section. We define the flow vector to be a 90 degree rotation of F_l . This *flow*'s orientation, denoted by F_l^θ or simply θ , captures global shape information of the regions, and its magnitude F_l^{mag} or simply m , reflects the region saliency instead of image contrast. To prune the number of nodes further, we threshold the magnitude of F_l and apply non-maximal suppression across the *flow*'s orientation.

6.2.4 Contour Graph Weight Setup

How do we utilize Lorentz force flow to detect salient contours? We will develop a graph formulation for this task as shown here. The first step is to define a graph $G^{contour} = \langle V^{contour}, W^{contour} \rangle$ which consists of a set $V^{contour}$ of n nodes, and a directed weight matrix $W^{contour}$ with dimensions $|V^{contour}| \times |V^{contour}|$. For the rest of the paper, $W^{contour}$ will be simply referred to as the directed graph \vec{W} . For each pair of nodes (v_i, v_j) where $v_j \in Nbhd(v_i) = \{(i, j) : \|(x_i, y_i) - (x_j, y_j)\| \leq \delta\}$ for a fixed distance δ , the weight

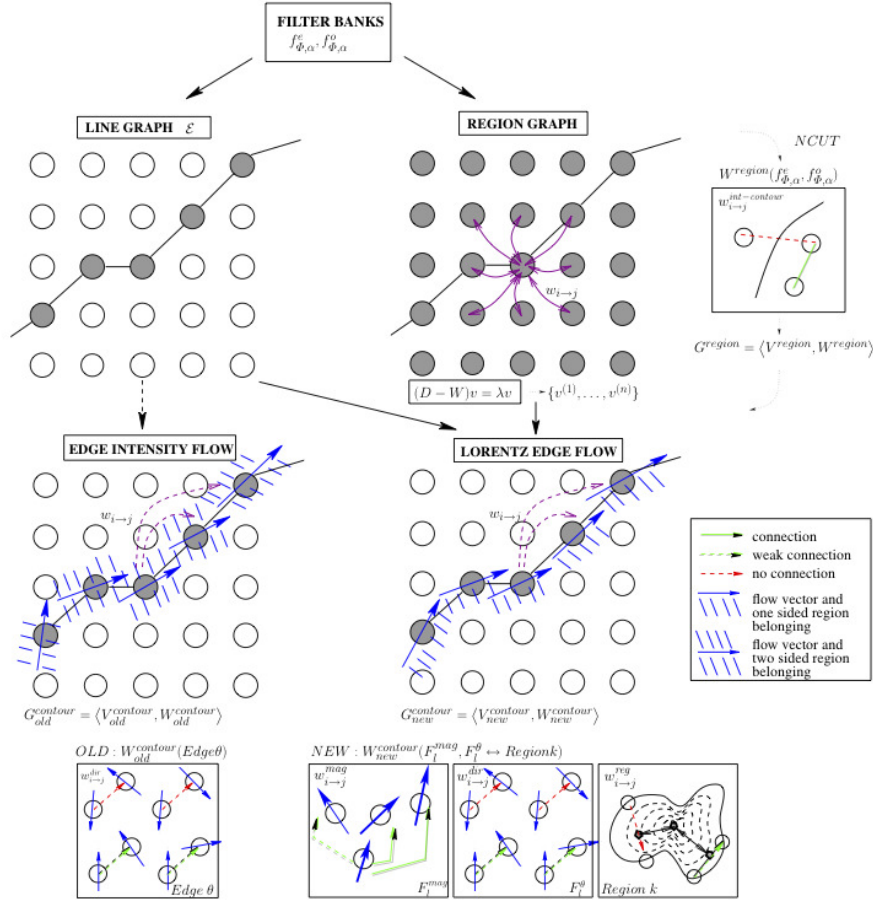


Figure 6.4: Schematic representation of our method. The phase and magnitude of the edge filter responses are used to obtain the initial line and region graphs (first row). In the line graph, the nodes are image edge pixels \mathcal{E} computed from the filter responses (section 2.1). In the region graph, the nodes are all image pixels, the graph is undirected and the weights are computed by Intervening Contours (illustrated on the right). The region segmentation eigenvectors computed by using NCuts are then used to compute the Lorentz Edge Flow on the edge mask \mathcal{E} . Our line graph, here denoted by a *new* subscript, has nodes that are duplicated copies of the edge mask \mathcal{E} with the flow vectors given by the Lorentz Edge Flow. Bottom Row: in contrast with previous contour grouping algorithms, denoted by a *old* subscript, which use image gradient orientation to compute the weights (lower left figure), the new vector flow (lower right figure) includes the global properties given by the segmentation eigenvectors $\{v^{(1)}, \dots, v^{(n)}\}$ and inherits an implicit region belonging. The graph weights (section 2.4) are computed by using the new flow’s magnitude (proportional to the thickness of the arrows) and orientation as well as information of the region labeling (lower right boxes). The three different cases are best viewed in color.

$w_{i \rightarrow j}$ of the edge connecting them is defined by:

$$w_{i \rightarrow j} = w_{i \rightarrow j}^{mag} w_{i \rightarrow j}^{dir} w_{i \rightarrow j}^{reg}, \quad (6.3)$$

where w^{mag} , w^{dir} and w^{reg} are the contour magnitude and direction and region cutting components. Flow directionality will further impose restricting conditions on which nodes i and j will be actually connected in the final weight matrix. Since there can be multiple nodes (each with its own (θ, m) pair) per image pixel, the weights of edges connecting two nodes corresponding to the same pixel are set to zero. For graph nodes (v_i, v_j) with corresponding flow magnitudes m_i and m_j and orientations θ_i and θ_j respectively, the three components are as follows:

contour magnitude $w_{i \rightarrow j}^{mag}$: The higher magnitudes correspond to contours bounding a region in the current eigenvector. To enhance also smaller magnitude contours, we choose a magnitude component that prefers similar magnitudes:

$$w_{i \rightarrow j}^{mag} = \exp(-|m_j - m_i|/\sigma_m). \quad (6.4)$$

contour bending $w_{i \rightarrow j}^{dir}$: the directionality of the edgels is measured by how much bending is needed in order to complete a curve between two nodes v_i and v_j . The weight is given by the co-circularity conditions in terms of the angles α_i and α_j between the nodes orientation and the distance vector between them:

$$w_{i \rightarrow j}^{dir} = \exp\left(-\frac{1 - \cos(2\alpha_i - \alpha_j)}{1 - \cos(2\sigma_d)}\right). \quad (6.5)$$

We restrict the bending amount by allowing only connections between nodes whose vectors satisfy $\cos \alpha_i \geq 0$ and $\cos \alpha_j \geq 0$. The orientations of both nodes cannot be simultaneously perpendicular with respect to the direction vector between them, i.e. $\cos \alpha_i \cos \alpha_j \neq 0$. We fix $\sigma_d = \pi/4$.

region cutting $w_{i \rightarrow j}^{reg}$: to avoid jumping between disconnected contours around the same region (hence cutting through the region rather than going around it) we look at the distance transform Δ of the regions to their boundaries (highest values will occur at the medial axis of the regions). For each region store the maximum Δ values as

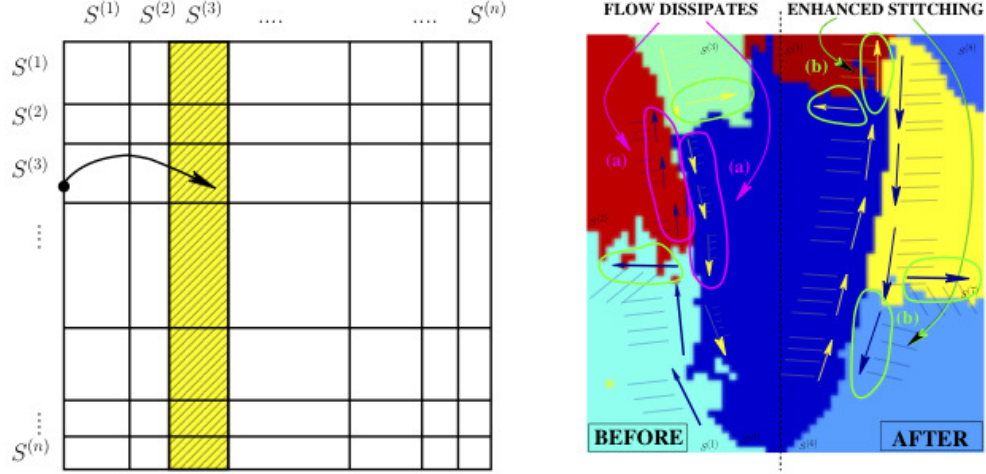


Figure 6.5: Graph normalization for directed Contour Random Walk setup. Left: Graph nodes are grouped according to the region segmentation labels S^k they arise from. We normalize the graph weights by the sum of connections reaching each region label S^k separately. This gives a boost to contours flowing between regions. Right: Effects of this normalization. When the flows belonging to each S^k are normalized to one, fainter flows that would otherwise not be able to compete with the stronger neighbors are enhanced (case b, indicated in green) instead of dissipated (case a, indicated in magenta). The ‘feather’ length on each arrow indicates the flow magnitude.

$\{\Delta_{\max}^{(1)}, \dots, \Delta_{\max}^{(n)}\}$. Let l_{ij} denote the set of points on the line between graph nodes (v_i, v_j) . We then compare the maximum point on this line, $\max(\Delta(l_{ij}))$, with the highest value Δ_{\max} of the region $S^{(k)}$ in which it occurs:

$$w_{i \rightarrow j}^{\text{reg}} = \exp\left(-\max(\Delta(l_{ij})) / (\sigma_r \Delta_{\max}^{(k)})\right), \quad (6.6)$$

hence penalizing cutting through the interior of the labeled regions $\{S^{(1)}, \dots, S^{(n)}\}$.

6.3 Salient Contour as Persistent Cyclic Random Walk

We generate a directed random walk matrix $\vec{P} = D^{-1}\vec{W}$ by normalizing the connections from each node. Normalizing the random walk matrix is known to provide a segmentation criterion robust to leakages. In contrast to the previous approaches which normalize \vec{W} by its total weighted connections [SM00], we choose to normalize the connections

within each eigenvector separately hence allowing the random walk to bifurcate. This is an important feature of our algorithm; it allows contours to flow on in different region segments without penalty. Since the connections of the flow in each region add up to one, this normalization effectively enhances faint contour flows that arise from low contrast regions, as well as minor flows through regions which would otherwise be suppressed by the stronger neighboring options. Recall that we can view \vec{W} as column blocks with $\vec{W} = [\vec{W}_{S^{(1)}}, \vec{W}_{S^{(2)}}, \dots, \vec{W}_{S^{(n)}}]$ are ordered in terms of the original region labels $\{S^{(1)}, \dots, S^{(n)}\}$. The normalization for each column block $[\vec{W}_{S^{(\beta)}}]$ for $\beta = \{1, \dots, n\}$ (illustrated in Fig. 5) will then have the form $D_\beta^{-1}[\vec{W}_{S^{(\beta)}}]$ where D_β is a diagonal matrix with entries $D_\beta(i, i) = \sum_j [\vec{W}_{S^{(\beta)}}(i, j)]$.

According to our graph setup, finding salient image contours amounts to searching for cycles in this directed graph. How would salient cycles appear in this random walk and how would they be distinguishable from generic clutter? We first notice an obvious necessary condition. If the random walk starting at a node comes back to itself with high probability, then there likely is a cycle passing through it. We denote the returning probability by $Pr(i, t) = \sum_\ell Pr(i, t \mid |\ell| = t)$. Here ℓ is a random walk cycle with length t passing i . However, this condition alone is not enough to identify meaningful structures. Consider the case where there are many distracting branches in the random walk. In this case, paths through the branches will still return to the same node but with different path lengths. Therefore, it is not sufficient to require the paths to return only; they have to return in the *same period* t .

Salient contours can be thought of as 1D cycles, structures that have a 2D geometry but are topologically 1D, i.e., a set of edgels with a well defined ordering and connections between them strictly follow it. 1D cycles have a special returning pattern probability $Pr(i, t)$. A random walk step on a 1D cycle tends to stay within the cycle, while moving a fixed amount forward in the cyclic ordering. Our task is to separate these *persistent cycles* from all other random walk ones. To quantify this observation, [ZSS07] introduces

a *peakness* measure of the random walk probability pattern:

$$R(i, T) = \frac{\sum_{k=1}^{\infty} Pr(i, kT)}{\sum_{k=0}^{\infty} Pr(i, k)}, \quad (6.7)$$

which computes the probability that the random walk returns at steps of multiples of T . $R(i, T)$ being high indicates the existence of *ID cycles* passing through node i . The key observation is that $R(i, T)$ closely relates to complex eigenvalues of the transition matrix P , instead of real eigenvalues [ZSS07]:

Theorem. (*Peakness of Random Walk Cycles*) $R(i, T)$ depends on P 's eigenvalues:

$$R(i, T) = \frac{\sum_j \Re(\frac{\lambda_j^T}{1-\lambda_j^T} \cdot U_{ij} V_{ij})}{\sum_j \Re(\frac{1}{1-\lambda_j} \cdot U_{ij} V_{ij})}, \quad (6.8)$$

where U_{ij} and V_{ij} are respectively the left and right eigenvectors of \vec{P} . This theorem shows that $R(i, T)$ is the average of $f(\lambda_j, T) = \Re(\frac{\lambda_j^T}{1-\lambda_j^T} \cdot U_{ij} V_{ij}) / \Re(\frac{1}{1-\lambda_j} \cdot U_{ij} V_{ij})$. For real λ_j , $f(\lambda_j, T) \leq 1/T$. For complex λ_j , $f(\lambda_j, T)$ can be large. For example, when $\lambda_j = s \cdot e^{i2\pi/T}$, $s \rightarrow 1$, $U_{ij} = V_{ij} = a \in \mathbb{R}$, $f(\lambda_j, T) \rightarrow \infty$. It is the complex eigenvalue with proper phase angle and magnitude that leads to repeated peaks.

6.4 Circular Embedding for Contour Grouping

The above analysis shows that salient contours correspond to *persistent* cycles in random walk, and their persistency can be computed from the eigenvalues of the random walk. It has been shown in [ZSS07] that the eigenvectors are an approximate solution to the following ideal cost for circular embedding of salient contour grouping. A circular embedding is a mapping between the vertex set V of the original graph to a circle plus the origin: $\mathcal{O}_{circ} : V \mapsto (r, \theta) : \mathcal{O}_{circ}(i) = x_i = (r_i, \theta_i)$, where r_i is the circle radius which can only take a positive fixed value r_0 or 0. θ_i is the angle associated with each node. The ideal embedding encodes both the *cut* and the *ordering* of graph nodes, and maximizes the

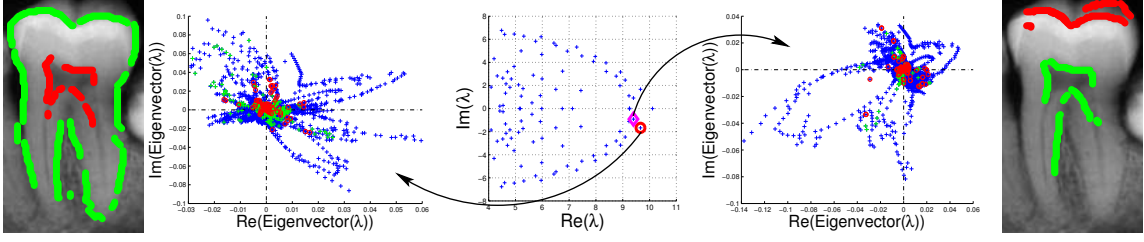


Figure 6.6: Examples of paths found by sampling. Center: The top eigenvalues sorted by their real components. For the eigenvector associated to each eigenvalue, we sample several contours. Displayed are two eigenvectors with two sampled contours each.

following score:

$$C_e(r, \theta, \Delta\theta) = \sum_{\substack{\theta_i < \theta_j \leq \theta_i + 2\Delta\theta \\ r_i > 0, r_j > 0}} \vec{P}_{ij} / |S| \cdot \frac{1}{\Delta\theta}, \quad (6.9)$$

where S is a subset of graph nodes and $\Delta\theta = \overline{\theta_j - \theta_i}$ is the *average jumping angle*.

Optimizing this score is not an easy task. Moreover, we are not only interested in the best solution of eqn(7.4), but in all the locally optimal solutions, which give all the 1D structures in the graph. We find a relaxation by setting $u = x$, $v = u \cdot e^{-i\Delta\theta}$. We set $c = t_0 e^{-i\Delta\theta}$ to be a constant. Eqn. (7.4) can be rewritten as maximizing $\Re((u^H \vec{P} v \cdot c) / (u^H v))$ with $u, v \in \mathbb{C}^n$ and is equivalent to the following optimization problem:

$$\max_{u, v \in \mathbb{C}^n} \Re(u^H \vec{P} v) \quad s.t. \quad u^H v = c. \quad (6.10)$$

This problem leads exactly to \vec{P} 's complex eigenvectors as shown in [ZSS07].

6.5 Contour Cuts as a Hermitian eigenvalue problem

In their recent extension, [KGS11] have shown that solving the eigenvalue problem $Px = \lambda X$ is just an approximation of a new generalized problem. While we summarize their new work in the the following sections, we refer to [KGS11, Gal00] for proofs and details.

In the new framework, each contour point x_j is represented by a complex number $x_j = r_j \exp(i\theta_j)$ which allows to encode ‘ordering’. Let $F = \Pi P$, where $P = D^{-1}W$, with

$D = \text{diag}(\sum_j W_{ij})$, and $\Pi = \text{diag}(\pi)$ the diagonal matrix of the stationary distribution P . Letting $H(\delta) = \frac{F \exp(-i\delta) + F^T \exp(i\delta)}{2}$, the new contour cut criterion becomes

$$\text{Ccut}(x) = \frac{\text{internal cut}(x) + \text{external cut}(x)}{\text{volume}(x)} = \frac{x^*[\Pi - H(\delta)]x}{x^*\Pi x} \quad (6.11)$$

where the right hand side can be written as a generalized Rayleigh quotient $R_{\Pi-H(\delta),\Pi}(x)$. This new formulation has also a random-walk interpretation in terms of the original distribution P :

$$\text{Ccut}(x) = \frac{x^*[\Pi - H(\delta)]x}{x^*\Pi x} = \frac{\sum_{(i,j) \in B} \pi_i P_{ij}}{\sum_{i \in C} \pi_i} \quad (6.12)$$

where, as in the original framework, P is the probability of moving a step forward on the contour. Therefore, minimizing $\text{Ccut}(x)$ is equivalent to minimizing $R_{\Pi-H(\delta),\Pi}(x)$ or, alternatively, maximizing the Rayleigh quotient $R_{H(\delta),\Pi}(x)$ and finding all the critical points corresponding to different contours, is achieved by solving

$$\max_x \frac{x^* H(\delta) x}{x^* \Pi x} \quad (6.13)$$

such that $x_j = r_j \exp(i\theta_j)$, $r_j \in \{0, 1\}$ and $\theta_i = O(j)\delta$.

It has been proved in [KGS11] that finding the eigenvectors of $\Pi^{-1}H$ is equivalent to solving for the eigenvector of the Hermitian matrix $\Pi^{-1/2}H(\delta)\Pi^{-1/2}$.

Theorem. *The critical points of the relaxed contour cut problem*

$$\max_x \frac{x^* H(\delta) x}{x^* \Pi x} \quad (6.14)$$

such that $x_i \in \mathbb{C}$, can be found by searching over δ and finding the eigenvectors of the corresponding matrices $\Pi^{-1}H(\delta)$; any eigenvectors for which $x^ H(\delta) x = |x^* F x|$ are critical points with respect to both x and δ .*

The original eigenvalue problem $Px = \lambda x$ can be seen as only an approximation of this more general problem, as the solutions of $\Pi^{-1}H(\delta)x = \lambda x$ are given as a combination of the eigenvectors of P and $\Pi^{-1}P^T\Pi$. Indeed, [KGS11] prove that if $P = D^{-1}W$ is a normal matrix, the critical points are given by the eigenvectors of P .

The contour results presented in this chapter rely on the original work by [ZSS07] and are found by solving the eigenvalue problem in terms of the matrix P .

6.6 Computational Solution

One way to extract contours is to find the maximal covering cycle in the complex eigenvectors space using a modified shortest path algorithm. We compute line fragments by local edge linking to reduce complexity and construct a directed graph where each node represents a fragment. Two nodes are connected by a directed edge according to the embedding phase angle (which specifies their ordering) and spatial connectivity.

For any two nodes in the graph, we seek to compute shortest path between them, which represents a contour hypothesis. We use dynamic programming to compute at each node v_j for all its parents v_i the recursive functions: $B_j(l_i) = \max_{l_j} (A(l_j) + L(l_j) + d(l_i, l_j) + \sum_{k \in C_j} B_k(l_j))$ and the best fragment l_j^* at which the max occurs. Here, A is the area spanned by l_i in the embedding space, d the phase overlap of l_i and l_j , L is a measure of the fragment's length and leaf nodes only consider the terms A , d and L . The optimal path L^* is obtained by picking the fragment that maximizes $(A(l_r) + L(l_r) + \sum_{k \in C_r} B_k(l_r))$ at the root node v_r and then backtracking the values l_j^* at each node v_j until a leaf node is reached.

To discover more contours, we sample a set of paths around the optimal one. We compute this path by sampling over the marginal distributions, as in [FH05], given recursively by the functions $S_j(l_i) \propto \sum_{v_c \in C_j} p(l_i, l_j) S_c(l_j)$, where $p(l_i, l_j)$ is the joint probability $p(l_i, l_j) \propto \exp(-(A(l_i) + d(l_j, l_i) + L(l_i)))$. For leaf nodes, $S_j(l_i)$ only considers the term $p(l_i, l_j)$. At the root node v_r , we sample from $\propto \sum_{v_c \in C_j} S_c(l_r)$. All contour fragments for nodes v_j thereafter are sampled from the marginals $S_j(l_i)$ until a leaf node is reached.

6.7 Experiments

Examples of different contours extracted from different eigenvectors for the tooth image are shown in Fig. 6.6. We compare our extracted contours with NCut on several objects of the Berkeley Segmentation Database [MFTM01] in Fig. 6.7. We select from the extracted samples the contours enclosing the regions that best match the object by shape context

Algorithm 3 Global Contour Flow Stitching

- 1: From the initial filter responses, use the magnitude and phase to obtain (a) an initial Edge Mask \mathcal{E} (section 2.1) and (b) NCut Region Segmentation (section 2.2) eigenvectors.
 - 2: Use the NCut eigenvectors to compute the Lorentz Edge Flow F_l on \mathcal{E} (section 2.3).
 - 3: Define a new directed contour graph $G^{contour} = \langle V^{contour}, W^{contour} \rangle$ in which the nodes are duplicated copies of the edge mask \mathcal{E} with the flow vectors given by the F_l .
 - 4: Compute graph weights using the new flow's magnitude and orientation as well as information of the region labeling: $w_{i \rightarrow j} = w_{i \rightarrow j}^{mag} w_{i \rightarrow j}^{dir} w_{i \rightarrow j}^{reg}$ (section 2.4).
 - 5: Solve for the eigenvectors of this new directed graph: $V(D - \vec{W}) = V\lambda$ (section 3).
 - 6: Extract maximum covering cycles using a modified shortest path algorithm in the complex embedding space (section 5) hence extracting corresponding salient contours in the image.
-

on the boundaries. Piecing segments into an object from the oversegmentation would require searching through exponentially many combinations of the fragmented regions. Our contour grouping can draw samples from limited salient contours (the number of samples is quadratic in the size of contour fragments). Our contours improve efficiency of segmentation and fix small leakage problems.

6.8 Summary

The algorithm summarized in Algorithm 3 extracts salient closed contours by exploiting region segmentation. The results produce segments that withstand varying contrast on the object boundaries and are therefore, less fragmented. We contrast our method with two related approaches:

1. Region Stitching using Hard Segmentation: The naive approach to region stitching would be to oversegment an image and use contour grouping over the region boundaries to stitch the segments back together. The main disadvantage is that the amount of segmentation needed is unknown a priori and unnecessary overfragmentation increases computation and reduces region saliency. Moreover, this contour grouping would find cycles through the various region boundaries independent of regions' size. Our approach only makes use

of a few salient region segments and enables the region boundary map to be juggled using the soft eigenvectors information and the initial edge mask.

2. Contour Grouping using Soft Segmentation: The method in [MAFM08] indeed uses the global information to extract contours and localize junctions. However, as this method does not involve any grouping, it does not resolve ambiguity of contour grouping in the places where three regions merge (i.e. junctions). Our approach introduces a normalization that enables the contour to flow through the junctions given by the region segment boundaries resulting in long contours. Since the contour has an inherent region belonging associated to it, the extracted contour is also guaranteed to enclose a region.

In this chapter we highlight how salient contours enclosing objects can be detected by combining the complementary power of region segmentation and contour grouping. Regions bridge the gaps between contours due to faint boundaries. Contour flows stitch oversegmented regions into large and salient ones, which can greatly simplify tasks such as object extraction and detection. Results on real images have shown great potentials of our approach on extracting salient object boundaries.

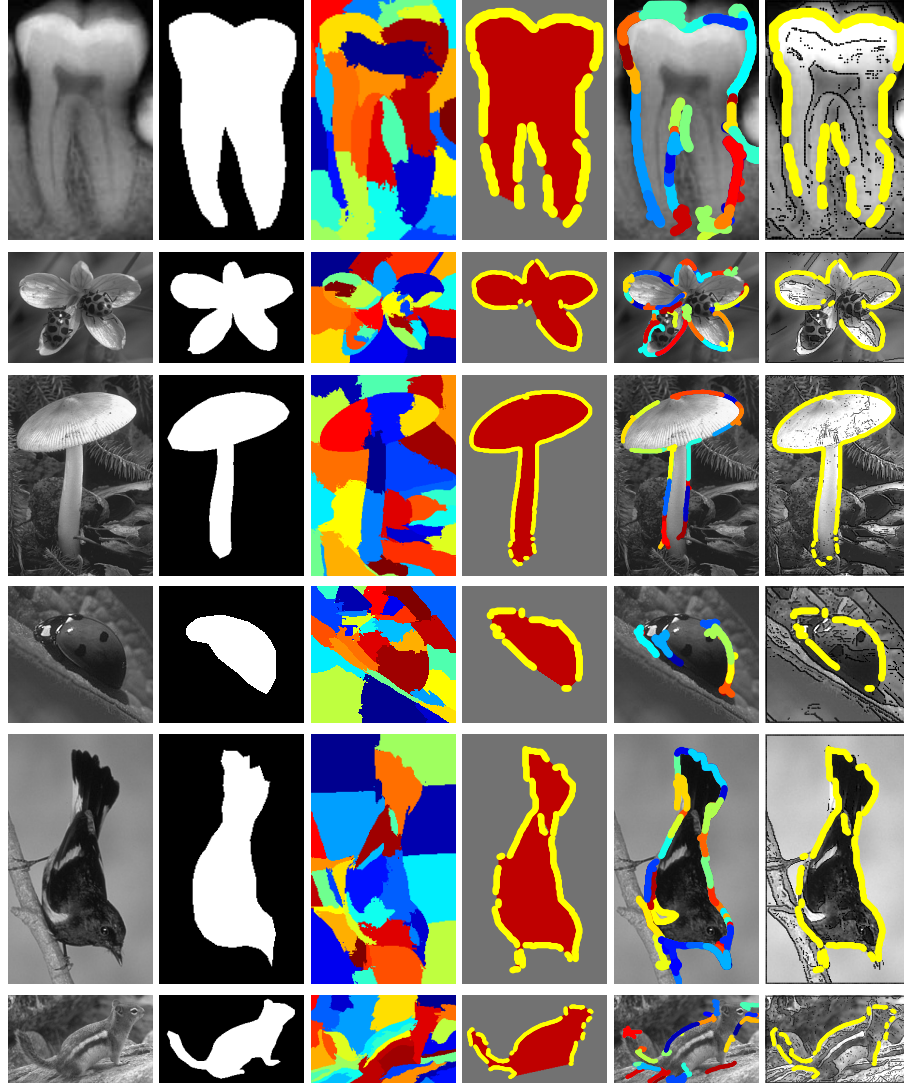


Figure 6.7: Extracted contours for several object images taken from the Berkeley segmentation dataset. From left to right: original image, ground truth object silhouette, ncut region segmentation, segmentation given from our extracted contour, untangling cycles contour grouping result (each contour with a different color), and finally, our extracted contour (yellow) on the original edge map (black). Note that the contour can be disconnected since we do allow jumping in the shortest path algorithm. The jumping is guided by the phase angle of the eigenvector in the embedding space, which allows to follow the contour throughout the boundary even if image continuity is broken. For all the images we used 30 eigenvectors for the Ncut segmentation and 20 for the contour grouping one.

Chapter 7

From Dots to 3D Tubes: Structural Correspondence as A Contour Grouping Problem

7.1 Introduction

Extracting 3D tubular cell structures across a stack of 2D image slices (Fig. 7.1) requires establishing cellular correspondences between images (Fig. 7.2), and we approach it as a contour grouping problem.

The most straightforward approach is to consider all the images in a stack simultaneously and solve a 3D pixel segmentation problem. There are many 3D image segmentation methods, e.g. level sets [OS88], clustering algorithms [CA79, WL93] and region growing [HS85], etc. While this formulation has an output format that naturally describes 3D tubes, it is unclear how grouping cues within and across individual 2D images can be properly integrated. Another major issue is scalability. The number of pixels increases with the increasing image resolution and number of slices, often rendering the computation infeasible. An alternative approach is to solve a series of 2D pixel segmentations independently and then combine the results to obtain the 3D structures [MS97, Mey94, CGNT09, SM00].

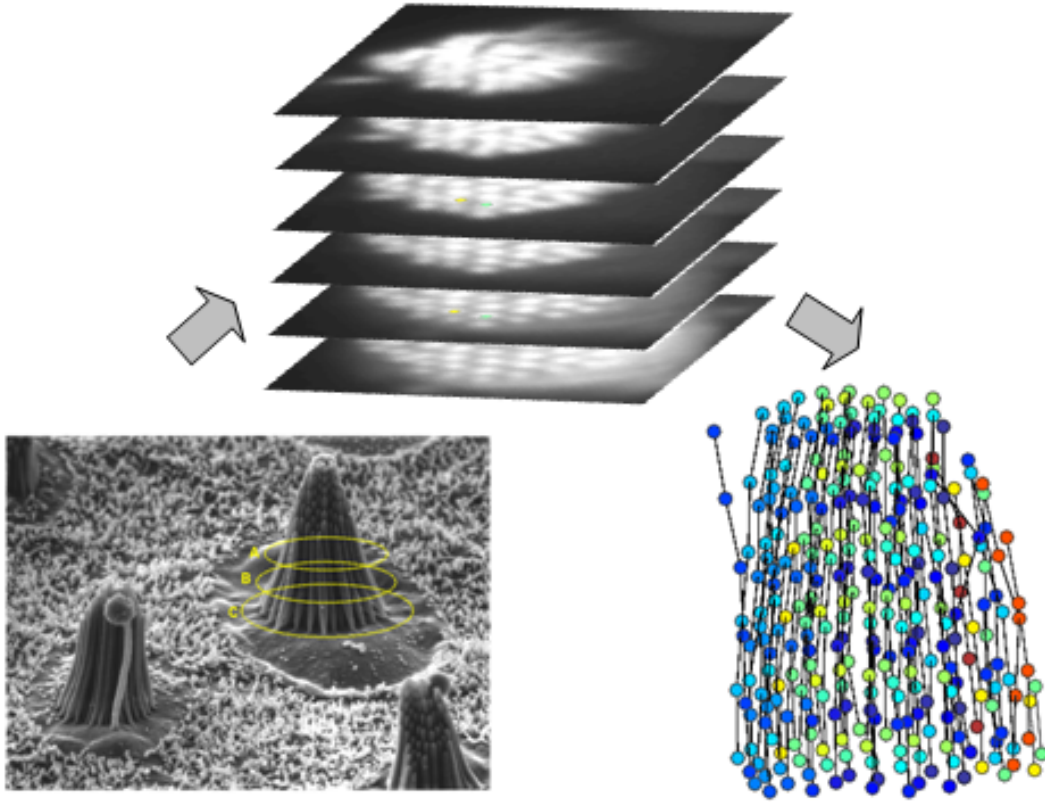


Figure 7.1: 3D Stereocilia segmentation. Hair cells are composed of tens of stereocilia organized in an organ-pipe-like formation of increasing height. We propose to solve the correspondence between cells across the 2D image stack as a contour grouping problem.

Since cells of different tubes often look alike, there are few good features to distinguish them. In practice, it is problematic to identify and cluster 3D tubes of varying lengths.

Both these approaches need to address the structural correspondence problem, which is explicit in the 2D segmentation approach and implicit in the 3D segmentation, e.g. in the derivation of motion cues linking the deformation of one image slice to the next image in the stack.

We propose to solve the structural correspondence between slices in the 3D space by solving a contour grouping problem [BY10c]. Here the *contours* are imaginary closed contour cycles cutting through the stack, going through the centers of the regions that have correspondences in the 3D space. We derive grouping cues between cells in adjacent slices based on their ability to relate in the 3D space. Those that form a long 3D tube in the space

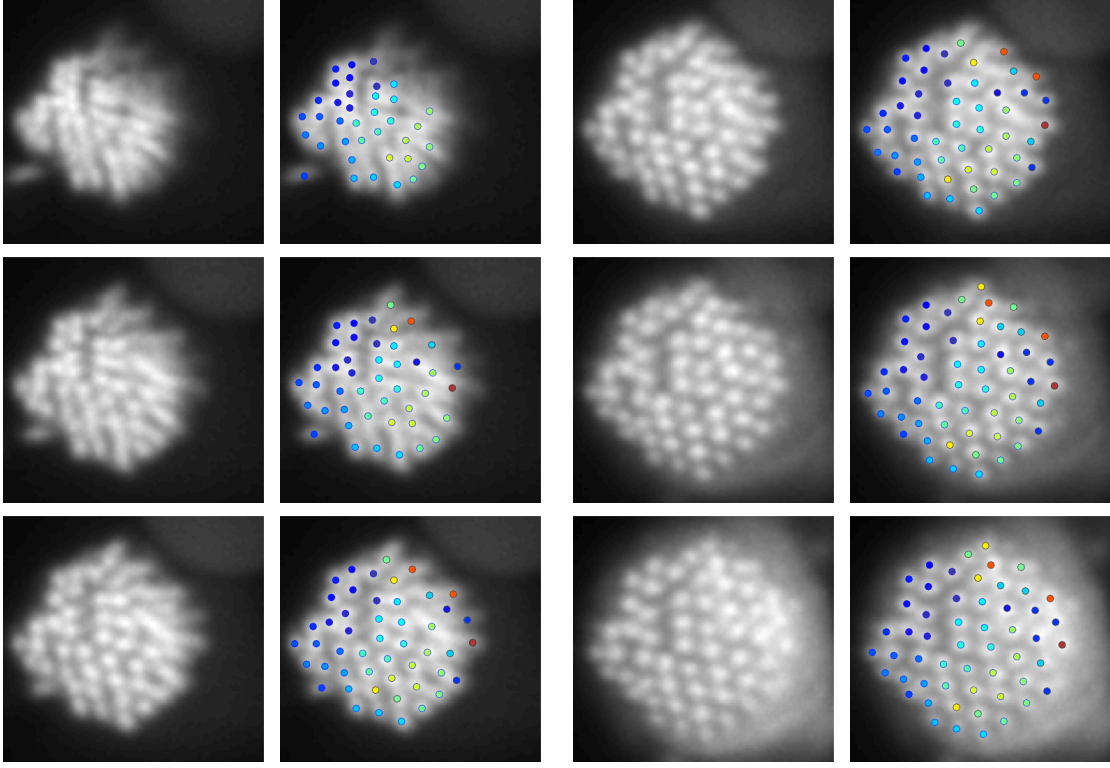


Figure 7.2: Extracting 3D tubes by finding correspondences (coded in color) across image stacks. In our contour grouping framework, missing correspondences simply result in shorter contours.

become the most salient contour, while those of shorter lengths become less salient. We solve the contour grouping problem in the spectral graph-theoretical framework [ZSS07]. The separation by the contour length is reflected in complex eigenvectors of different magnitudes, from which these 3D tubes of various lengths can be extracted.

The most appealing strength of our formulation is that missing correspondences no longer poses a special and difficult problem. They simply lead to shorter contours which are extracted from cycles of shorter lengths.

7.2 Contour Grouping for 3D Correspondence

To find the 3D tubes transversing a 2D image stack, we formulate the cellular correspondence as a contour grouping problem. Each 3D tube is represented by a contour starting

from the cell in the lower stack, transversing the stack and turning backward once the end of the hair cell is reached, returning back to the initial slice.

Our method is inspired by the *Untangling Cycles* [ZSS07] model for contour grouping, which extracts contours by searching for salient cycles of a random walk, within a spectral graph partitioning framework. In our setup, nodes of the weighted graph no longer represent edge pixels of possible 2D contours on an image, instead, they represent segment centers from each individual stack.

7.2.1 Untangling Cycles for Contour Grouping

Within a single image, modeling contour grouping as a spectral graph partitioning problem can be seen as finding persistent cycles in a random walk along a weighted graph $G(V, E)$, where weights W correspond to ‘edge’ nodes V , derived from an initial edge map, and edges E between nodes are given by spatial proximity in the original image. We generate a directed random walk matrix $\vec{P} = D^{-1}W$ normalized with respect to the outgoing connections from each node, i.e. D is a diagonal matrix with entries $D(i, i) = \sum_j [W(i, j)]$. The criterion is more robust to contour leakages.

The idea behind *Untangling Cycles*, is that paths within the random walk have to return to their initial starting point (to guarantee closeness of the contour) and have to do so in the same period t (to guarantee the repeated transversing of a salient contour rather than an accidental returning from surrounding clutter). In this context, salient contours can be thought of as 1D cycles, with a special returning pattern quantified through a *peakness* measure $R(i, T)$ of the random walk probability pattern at steps of multiples of T .

The key observation [ZSS07] is that $R(i, T)$ closely relates to complex eigenvalues of the transition matrix P , instead of real eigenvalues, showing that it is the complex eigenvalues with proper phase angle and magnitude that lead to repeated peaks. A random walk step on a 1D cycle tends to stay within the cycle, while moving a fixed amount forward in the cyclic ordering. Therefore, the link between finding image contours and distinguish them from clutter amounts to searching for *persistent cycles* in this random

walk.

7.2.2 Graph Setup for Structural Correspondence

We start with stack of images and their respective segmentations and assign to each segment a node. With this new set of vertices, nodes can now have the same image spatial coordinates, while belonging to a different stack number. Edges between nodes are added if the corresponding segments are lying in adjacent images, while, within the same stack, only a self returning edge is added at each node. The two frameworks are sketched in Fig. 7.3. The weight associated to each edge has two components.

First, we recall the single image scenario, where each node had an associated direction to it, and a ‘good’ contour was one maximizing smoothness. We still seek smoothness in terms of the 3D tubular structure, so that intuitively each tube does not bend too much while transversing the stacks. If we imagine each node with an arrow pointing downwards, measuring bending between two nodes can be simplified by projecting them onto one single plane and measuring the spatial distance between them. Letting the positions of node i and j be d_i and d_j respectively on the $x - y$ planes of the individual images, we define

$$\xi(i, j) = \exp(-|d_j - d_i|/\sigma) \quad (7.1)$$

We have empirically found that the introduction of a complex component θ to the weights allows better cycle discrimination in the embedding space. ψ encodes the number steps taken in the random walk, so that jumping between stacks, in terms of the n images stacks $\{I^{(1)}, I^{(2)}, \dots, I^{(n)}\}$, can be written as

$$\psi_{i \rightarrow j} = t - s, \quad (7.2)$$

$$i \in I^{(s)}, j \in I^{(t)}$$

$$s, t \in 1, \dots, n$$

where the unit step is given by $\psi = 1$. The final weight between two nodes will then be

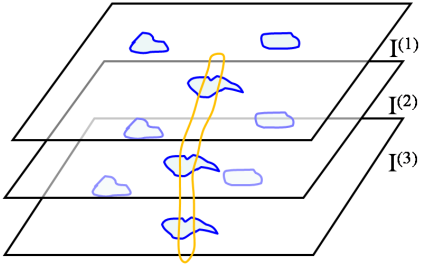
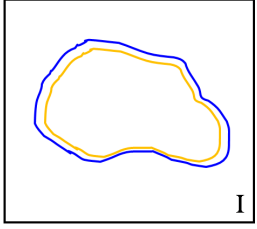
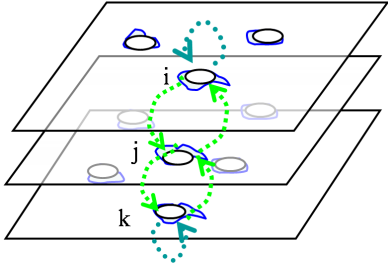
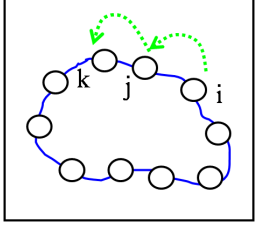
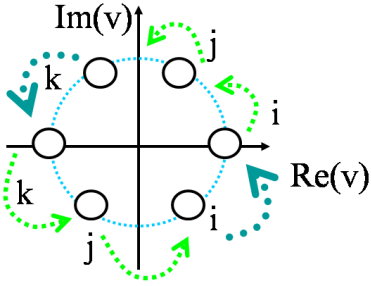
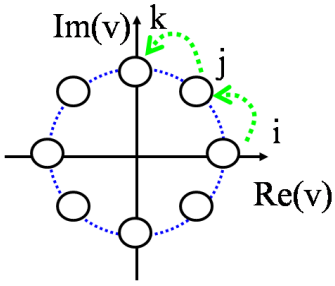
	structural correspondence in 3D	contour grouping
a: image input		
b: graph setup		
c: complex eigensolution		

Figure 7.3: Structural correspondence as a contour grouping problem. We contrast our method with the traditional Untangling Cycles model. In order to solve structure correspondence and find the cycles (**a**, yellow), nodes in the graph (**b**) are no longer edge pixels; instead, they represent segments on the individual images. In order to have close contours we add a return link to each outer stack node (dark green). Cycles through the 3D tubular structures then correspond to cycles in the complex eigenvectors v of a random walk matrix (**c**).

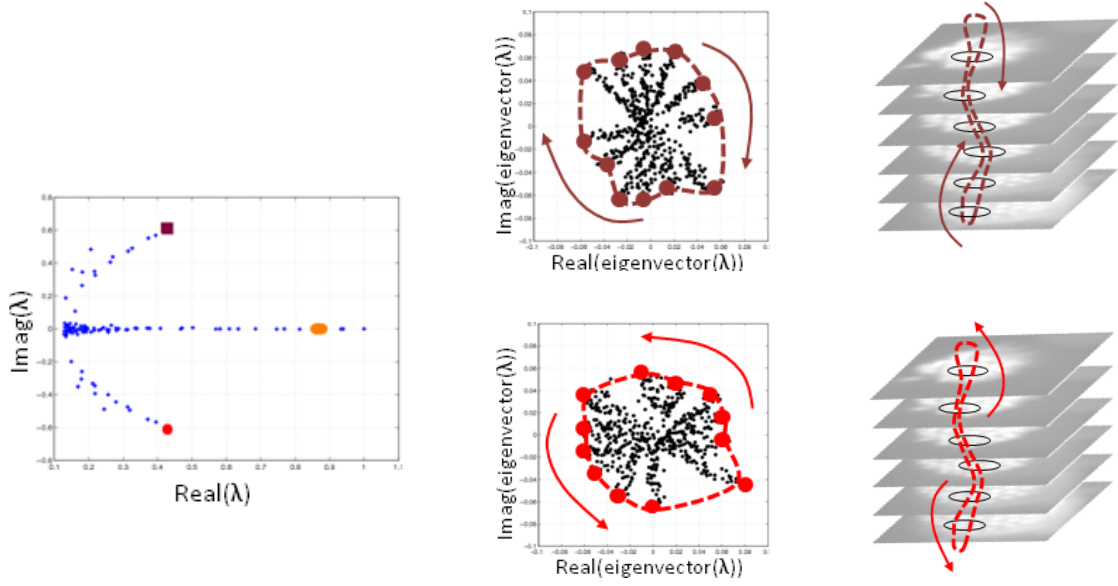


Figure 7.4: Embedding Space. While real eigenvalues (orange ellipse) collapse eigenvectors on the real line, eigenvectors corresponding to complex eigenvalues with large phase angle and real components (purple square, red circle) encode cycles information: each cycle corresponds to a cycle in the original graph, which itself encodes a cycle through the image stacks. Positive and negative phase angles encode clock and anti-clockwise cycles respectively.

given by:

$$w_{i \rightarrow j} = \begin{cases} \xi(i, j) + (\psi_{i \rightarrow j})\mathbf{i} & i \neq j \\ \xi(i, i) + \mathbf{i} & i = j = 1, n \\ \xi(i, i) * 0.1 + \mathbf{i} & i = j = 2, \dots, n - 1 \end{cases} \quad (7.3)$$

In order to have close contours we add a return link to each node Fig. 7.3. Adding a returning edge at each layer guarantees that cycles of shorter lengths will also be able to be picked up in the random walk, hence dealing with missing correspondence throughout the stacks.

7.2.3 Circular Embedding for Random Walk Cycles

In order to understand the intuition behind the eigenvectors of the random walk matrix P , [ZSS07] showed that the eigenvectors are an approximate solution to an ideal cost for circular embedding of salient contour grouping.

A circular embedding (Fig. 7.4) is a mapping between the vertex set V of the original graph to a circle plus the origin: $\mathcal{O}_{circ} : V \mapsto (r, \theta) : \mathcal{O}_{circ}(i) = x_i = (r_i, \theta_i)$, where r_i is the circle radius which can only take a positive fixed value r_0 or 0. θ_i is the angle associated with each node. The ideal embedding encodes both the *cut* and the *ordering* of graph nodes, and maximizes the following score:

$$C_e(r, \theta, \Delta\theta) = \sum_{\substack{\theta_i < \theta_j \leq \theta_i + 2\Delta\theta \\ r_i > 0, r_j > 0}} \vec{P}_{ij} / |S| \cdot \frac{1}{\Delta\theta}, \quad (7.4)$$

where S is a subset of graph nodes and $\Delta\theta = \overline{\theta_j - \theta_i}$ is the *average jumping angle*.

Setting $u = x$, $v = u \cdot e^{-i\Delta\theta}$, and $c = t_0 e^{-i\Delta\theta}$, we can rewrite Eqn. (7.4) as maximizing $\text{Real}((u^H \vec{P} v \cdot c) / (u^H v))$ and it is equivalent to the following:

$$\max_{u, v \in \mathbb{C}^n} \text{Real}(u^H \vec{P} v) \quad \text{s.t. } u^H v = c. \quad (7.5)$$

which leads exactly to \vec{P} 's complex eigenvectors.

As in the previous chapter, our results pre-date the untangling cycles extension proposed by [KGS11, Gal00]. In section 6.5, we briefly summarized their work proving that solving the eigenvalue problem $Px = \lambda X$ is just an approximation of a new generalized problem. Although we introduce new complex weights, the contour results presented in this chapter rely on the original work by [ZSS07] and are found by solving the eigenvalue problem in terms of the matrix P .

7.3 Experiments

We have 20 stacks of microscopic haircell images. For each stack, we selected 6 image slices. The parameters used were kept constant for all stacks: connectivity radius of 25 pixels and $\sigma = 5$. The choice of radius relates to the amount of shifting allowed between the images. Cells centers were found using the 2D segmentation method of [BY10a]. We allowed cells to connect only up to two slices forward. Cylindrical tubes representing hair cells in 3D are then picked up individually as cycles in the eigenvectors Fig. 7.4. Given n stacks, cycles of length 6 represent cells that can be seen throughout the slices, while cycles of length 5, 4, \dots represent the ones with missing correspondences, i.e. disappearing cells in the upper stacks.

Our implementation in MATLAB takes about 1 second to find the correspondences in a stack of about 60 tubes. Our method is summarized in Algorithm 4. We show results of two different image stacks in Fig. 7.5 and Fig. 7.6, illustrating the main two challenges present in these type of cells. Hair cell bundles are formed by an organ-pipe like structure of tubular cells, with the radially outer ones of shorter lengths when compared to the central ones. Depending on each hair bundle and imaging technique, the difference in shrinking of the cells can be seen either as cells shifting in space, possibly assuming ellipsoidal cross-sections before dissipating in the next image layer, or as cells concentrically shrinking, if the images and the hair bundle actually align with respect to its center. Fig. 7.5 and Fig. 7.6, each of one image stack, illustrate how two different eigenvectors contain the information of different length cycles.

Algorithm 4 Structural correspondence algorithm.

Given an image stack with their associated segments:

1. Build a graph $G(V, E)$ where the nodes V correspond to the segments throughout the stack.
 2. Compute the weight matrix by Eqn. 7.3.
 3. Solve the complex eigenvectors of the associated random walk matrix \vec{P} .
 4. Extract contours from cycles in the embedding space [ZSS07].
-

7.4 Summary

We present a contour grouping approach to extract tubular structures across image stacks. The key insight is to view the structural correspondence problem as finding closed contours across the image stack. We formulate it in the spectral graph partitioning framework, where the random walk matrix is constructed from complex graph weights capable of encoding stack ordering. While the resulting eigenvectors correctly encode the tubular structure information for all cells, regardless of their lengths throughout the stacks.

What’s most appealing about our method is that cycles found can handle missing correspondences in the form of disappearing, shrinking, and shifting cells. In addition, this particular choice of contour grouping with complex weights allows all salient cycles of the same length to be extracted from the same eigenvector.

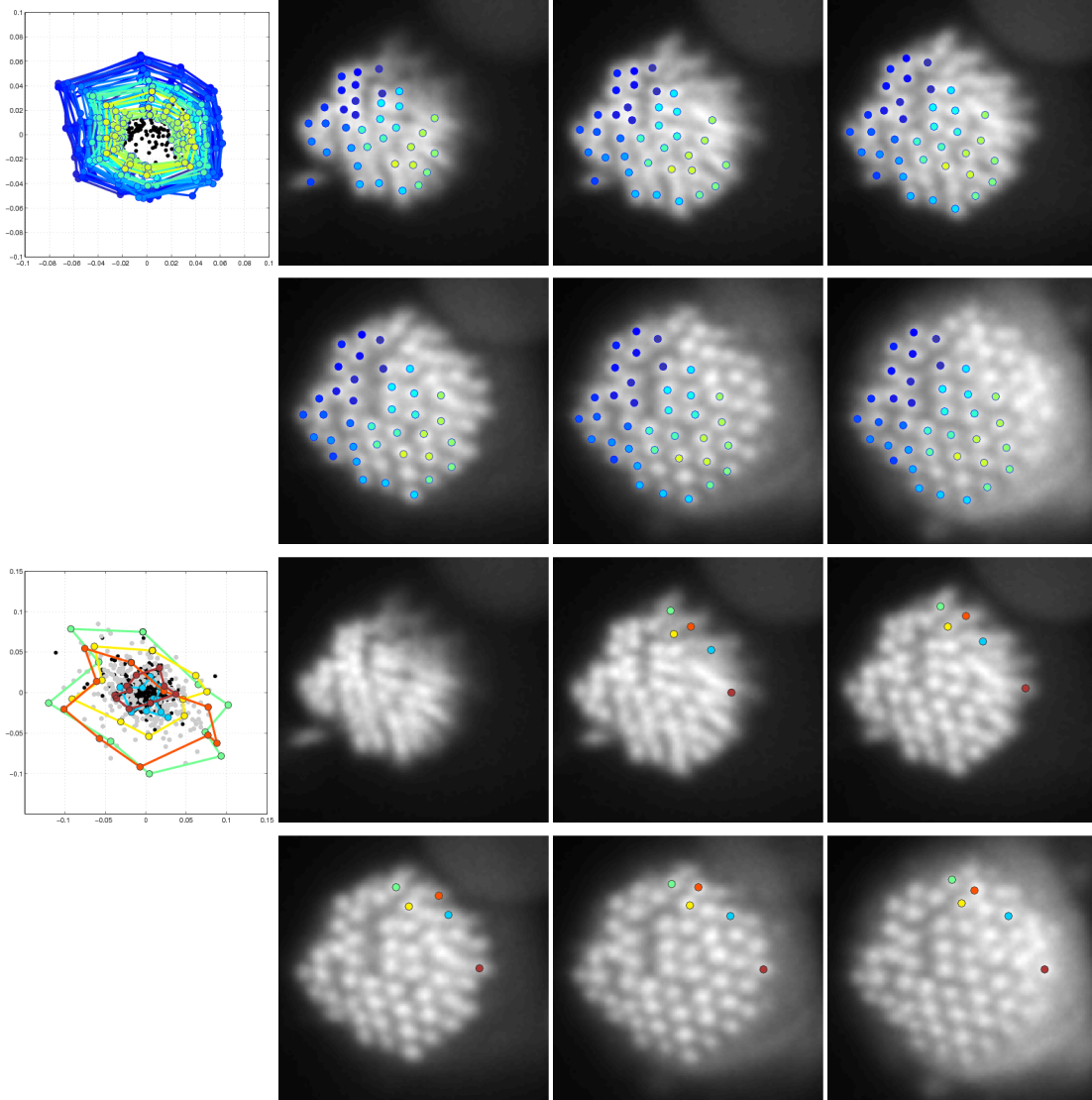


Figure 7.5: Extracting 3D tubes shrinking across image stacks. Sample cycles extracted from two eigenvectors: above, cycles of lengths 6 and below, of length 5. Each cycle in the embedding space is color-coded to show the 3D cell correspondences throughout the 6-image stack. Longer cycles are first extracted from the eigenvector.

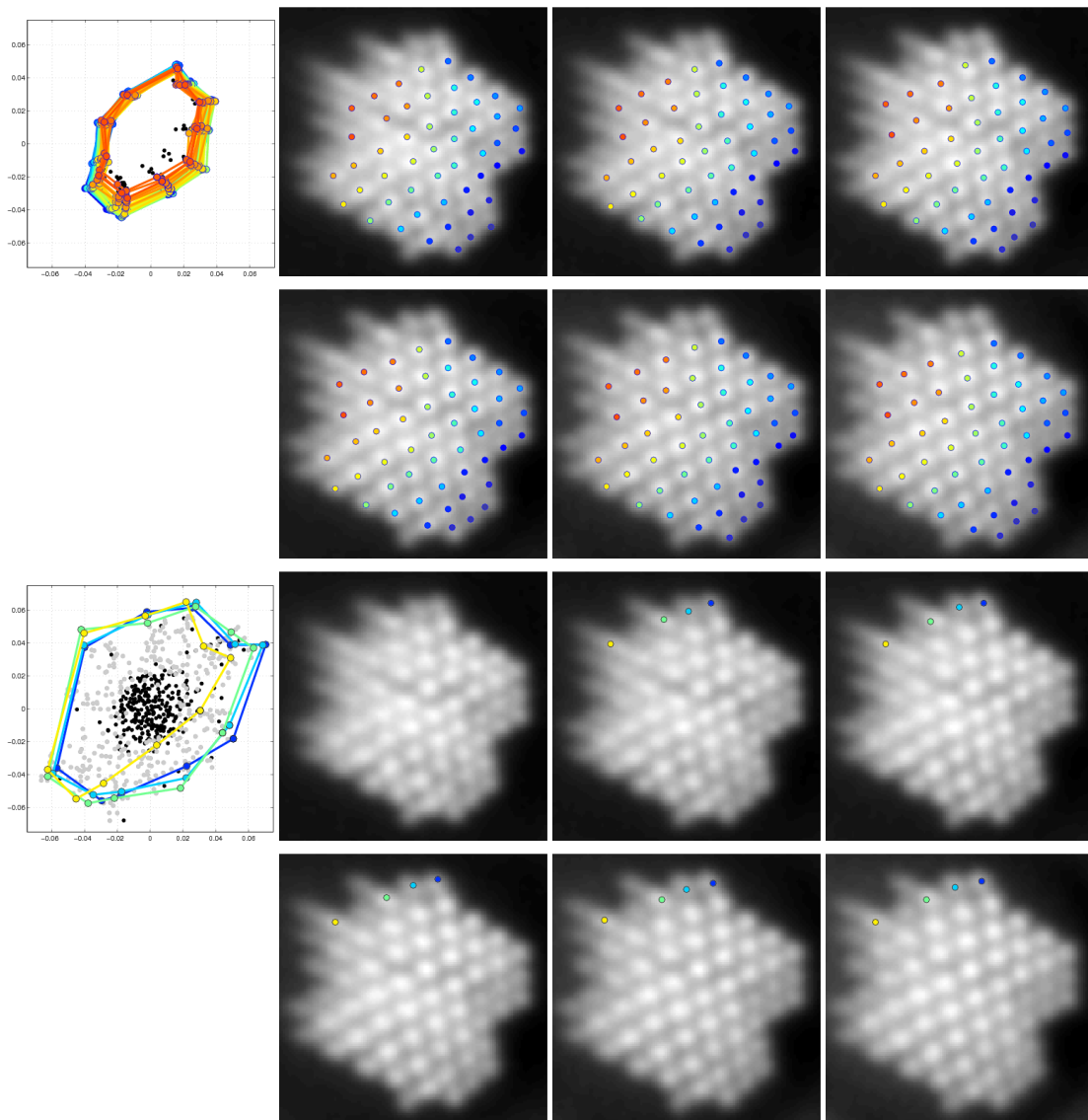


Figure 7.6: Extracting 3D tubes shifting across image stacks. Sample cycles extracted from two eigenvectors: above, cycles of lengths 6 and below, of length 5. Each cycle in the embedding space is color-coded to show the 3D cell correspondences throughout the 6-image stack. Longer cycles are first extracted from the eigenvector.

Chapter 8

From Dots to Real-Scene Textures

Dots are truly everywhere. As illustrated in Fig. 8.1, corn, cobblestones and roof tiles can all be catalogued as dots. Increasing scales, windows can be seen as dots in a building, and the building themselves become dots from a satellite image. Our dot extraction approach, relied on the assumption that dots appeared as many repeated elements and spatially homogeneous throughout the image. In real-images these dots are denoted by *textels* or *textons* and their repetition results in a textural surface.

Textural surfaces have traditionally been studied as stochastic patterns resulting from these basic textons elements and special focus has been given to understanding their underlying statistics [Jul62], by defining Markov texture models [GG90], or using multi-scale decomposition approaches such as wavelet representations [ZWM98, HB95, PS00]. By



Figure 8.1: Dots in real-scene images. A mosaic floor made of cobblestones, an irregular lemon array, shaded cobble stone street, corn on the cob and a rooftop with regular pattern.

viewing these textures as 2D stochastic entities, the individual dots and their (3D) geometry became of secondary importance.

The focus shifted back to ‘geometrically meaningful’ textons in shape from texture models, approximating texture fields from varying views of single textons [For01, LF04, MR97] and recovering the underlying lattice for near-regular or regular lattice structure [HLEL06, PBCL09]. In many cases, assumptions on shape, viewpoint or scales of the textons have to be set a priori. Detecting the individual textons was done in the work of [LM96], which, for the first time, highlighted the importance of feature positional relationship within each texton and the point-to-point correspondences between features of different textons. The number of texton elements in the image though had to remain small.

In this chapter, we show the potential of our finding dots approach to automatically detect textons in real-scene images without prior knowledge of specific shape or viewpoint. While our method is meant for microscopic images and fails in the presence of extreme gradient changes that can occur in real-scenes, our results show that our approach can successfully extract textons in many instances. The precise location and contour delineation of the textons found can be of potential benefit for many shape from texture or graphics applications. Hence, we conclude this chapter by presenting one such example of a new method for synthesizing 3D textural scenes.

8.1 Automatic Parameter Estimation

While in microscopic applications dots of different scales represent dots of different nature and parameters can be set accordingly to find one type versus the other, in real-scene images the same object can appear as dots of different sizes due to geometrical perspective or variations in surface geometries, and constant parameters are no longer sufficient. We exploit the patch segmentation approach of Chap. 4 to adaptively estimate the radii parameters within each image patch, thus properly segmenting all textons independently of their size or viewing angle throughout the image.

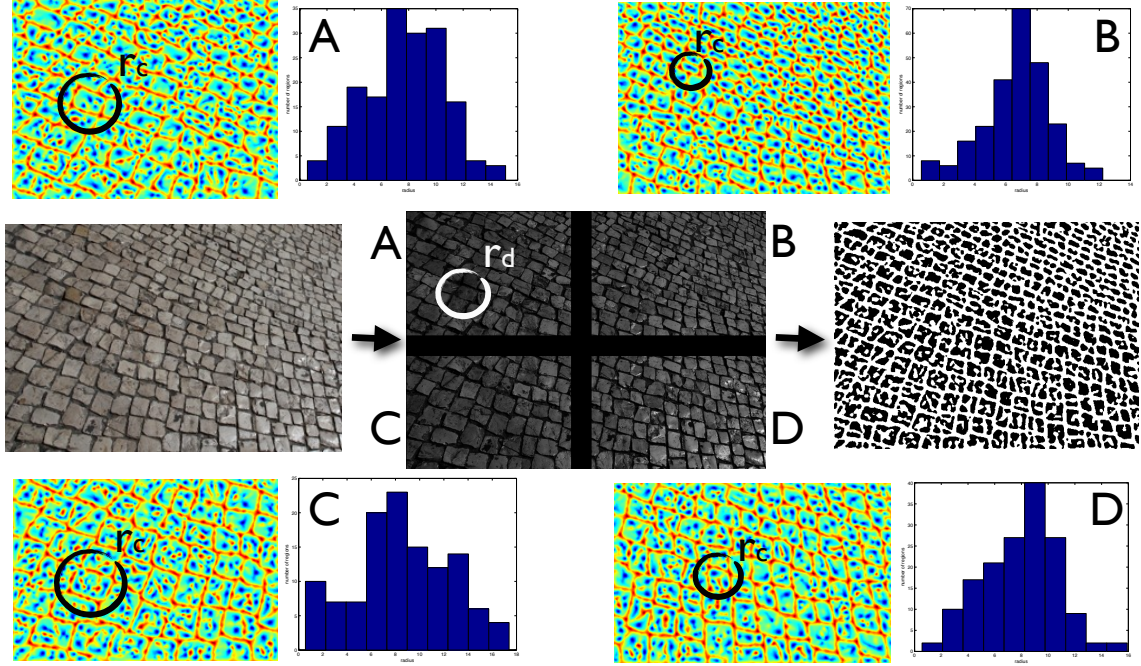


Figure 8.2: Adaptive radii estimation. The large (600×900 pixels) input image is divided into four patches for processing. The vector field p is computed by setting the *direction radius* to r_d . For each patch we compute the degree by adding up the angles differences from each pixel to its surrounding neighbors. Negative values (blue) represent sinks (textures) while positive ones (red) are background. By thresholding these new images, we obtain many disconnected components, whose radii distributions shown in the histograms are used to estimate the average core radius for each patch. Radii parameters are set accordingly, such as the *convexity radius* r_c displayed on each patch.

Recall from Chap. 5 that four radii are involved in the local grouping cues setup:

r_d : radius of the neighborhood to compute the peak direction vector p . Beyond a certain radius, r_d gives a constant p , so it can be chosen independently of the dot scale.

r_c : radius of the neighborhood used to compute the feature vector F , it encodes the local convexity of the neighborhood and increasing it allows to look for larger shapes.

r_A : radius used to compute short-range attraction cues. Increasing r_A brings nearby pixels together, so it should be at least comparable to the dot size.

r_R : radius that determines the extent of long-range repulsion cues.

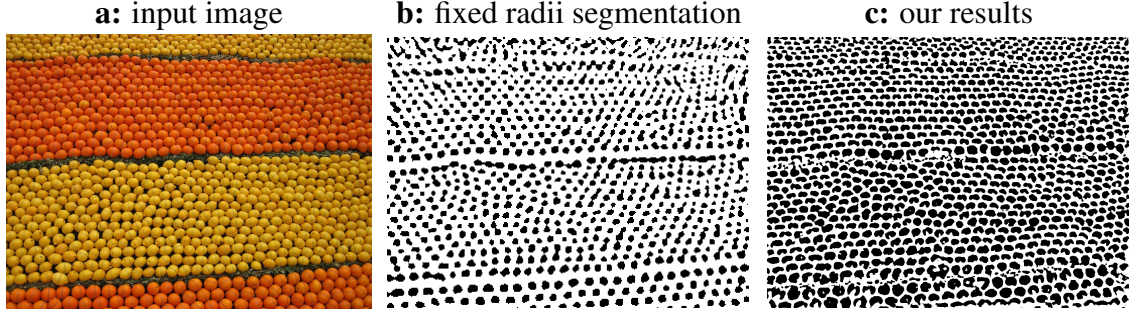


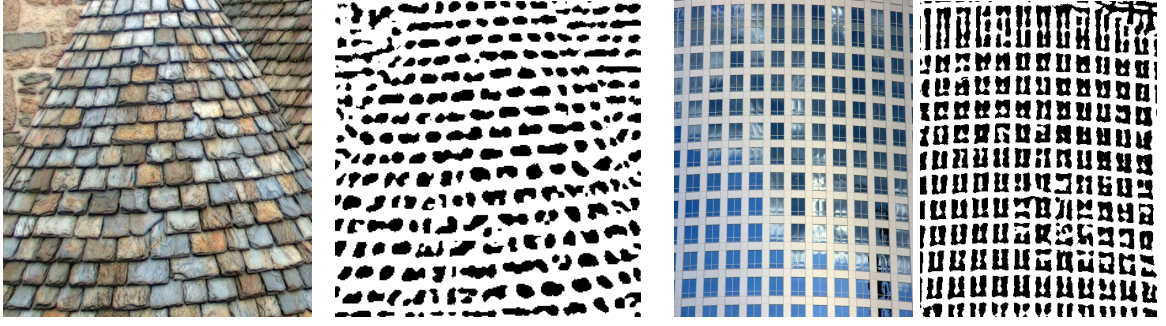
Figure 8.3: Automatic segmentation of **a)** textons in image (e.g. lemons and mandarines) of different scales. We compare our **c)** adaptive radii results to a **b)** fixed parameter segmentation. While fixed parameters yield cleaner results for the larger textons, they incorrectly clusters together most of the smaller ones. In addition, the fixed parameters fail in capturing the appropriate texton shape and results in a uniform dot appearance.

Intuitively, r_c has to entirely contain the dot to set a proper ‘template’ size for the feature vector F computation, while r_A and r_R have to be large enough to capture shape information and surrounding information respectively. r_c , r_A and r_R can be therefore defined in terms of a core radius r_0 that represents the size of the average dot, computed with the heuristics depicted in Fig. 8.4. Starting with degree of the convexity feature F , we threshold the negative values to obtain disconnected components and estimate the radius of each component. We take the radius associated with peak of the radii distribution as core radius estimate, restricting the radius to be at least 4 pixels. The rest of the parameters can then be set accordingly: $r_c = \max(4, 1.5r_0)$, $r_a = \max(4, r_0)$ and $r_R = \max(20, 3r_0)$. Sample results with constant and our new adaptive parameters are compared Fig. 8.4.

8.2 Experiments on the CMU NRT database

We implement our algorithm in MATLAB and apply to a variety of textons in real-scenes structures (Fig. 8.4 and Fig. 8.5). Distribution and sizes can vary significantly even within a single image. We test our method on the Near-regular textures from the CMU NRT database (<http://vivid.cse.psu.edu/texturedb/gallery/>). Images are further divided into 2×2 patches with a patch overlap size of 20 pixels.

a: intensity and color variation within textons



b: occlusions and intensity variation across textons

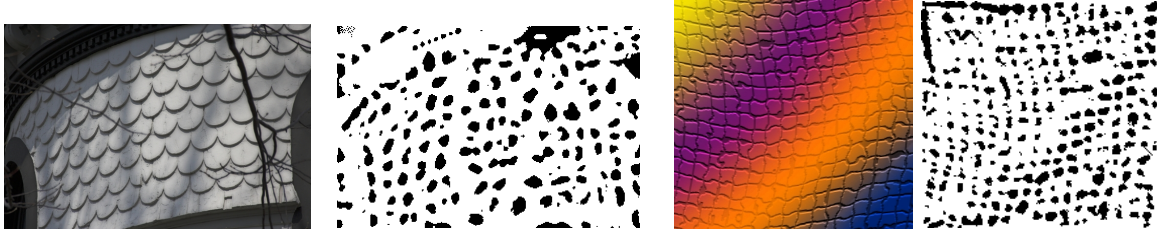


Figure 8.4: Intensity variations within and across textons. **a)** A common background allows textons to be extracted simultaneously regardless of color variations within the tiles. **b)** In the presence of larger intensity variations across the image (shadows, occluding objects, background gradient variation), local convexity cues are not sufficient to properly extract all the dots and additional directional information should be used.

We measure the goodness of segmentation by scoring our results with respect to the ground-truth center locations provided with the dataset. We define for each image I :

$$\text{score}(I) = \frac{\text{No. true positives}}{\text{No. segments} + \text{No. missed}} \quad (8.1)$$

and, given the dot variety, we display each score on the respective images in Fig. 8.5. As we show in Fig. 8.4, our approach is able to deal with varying color and gradients within each textural element. The importance relies on having a common background that can be distinguished by a lighter (or darker) intensity. This assumption fails in the presence of larger image gradients such shadows or occluding objects or varying background gradients (Fig. 8.4b), which could be obviated by adding a directional component to the local weights, as shown in Sec. 5.3.2. Overall though, our method is able to automatically segment the dots in many real-scene images, independently of scale and viewing conditions.

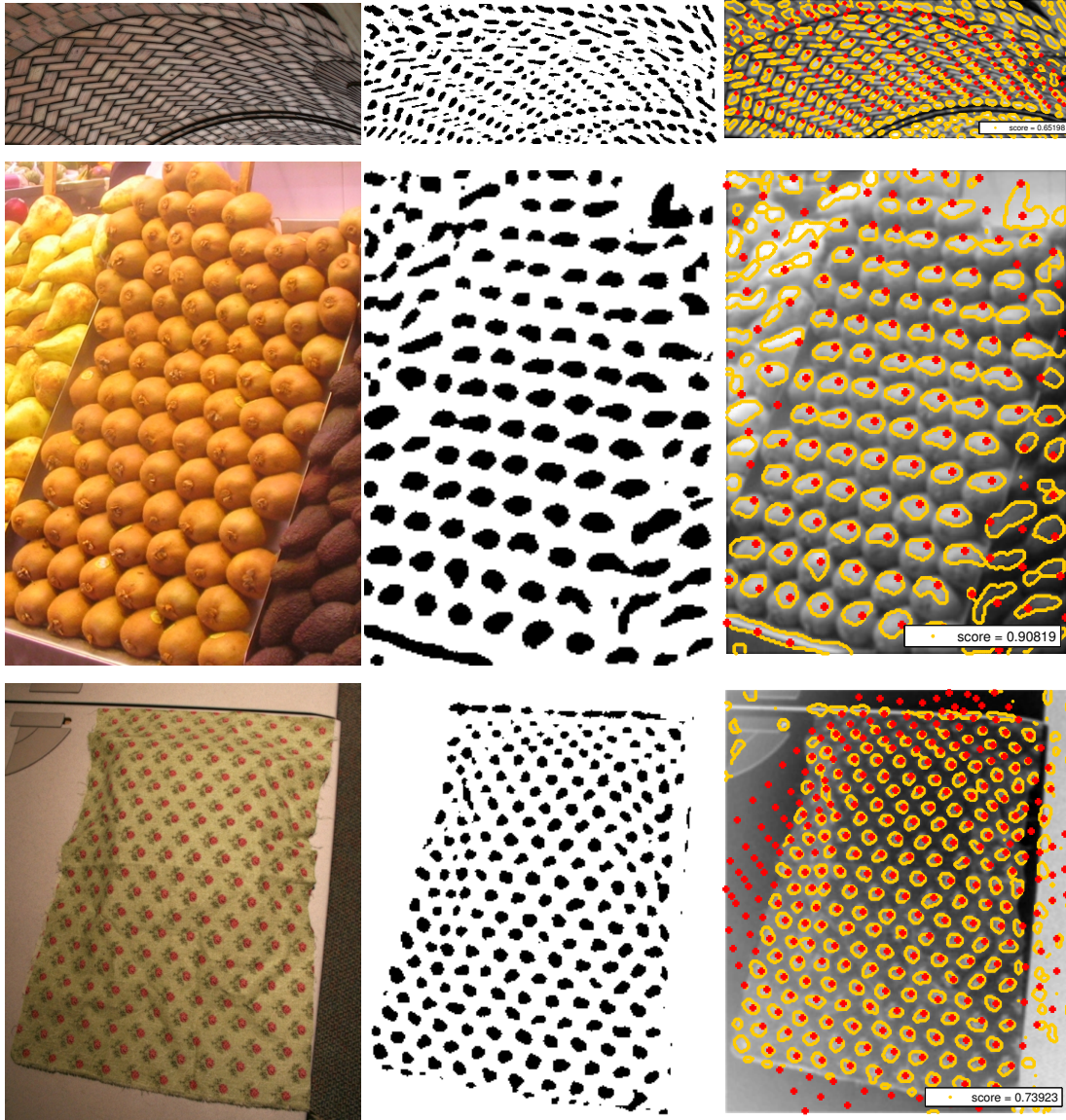


Figure 8.5: Automatic texton segmentation on the CMU NRT dataset on different texton lattices with complex geometries. The parameters adapt to each image allowing to find the textons present independently of their shape or scale.

8.3 3D Scene Rendering From Textured Images

The concept of texture mapping is one of the most powerful perceptual insights in computer graphics. Simply put, the idea is that one can create a very realistic representation of even the most complex scene using just a few surfaces, as long as these surfaces are “painted” in the right way. Planar surfaces, in particular, have a very nice property: any two views of the same plane are related via a simple homography transformation. This means, for example, that a planar building facade on a photograph could be easily warped to look like it’s being seen from any other viewpoint. This concept has been extensively used by image-based modeling and rendering systems to cheaply and easily create very realistic 3D models from just a few photographs (e.g. [DTM96]) or even a single image (e.g. [HAA97]). In the game industry, planar texture mapping from photographs is widely used for adding realistic road surfaces and building facades to urban environments.

Planar texture mapping makes two somewhat related assumptions: 1) that the surface to be texture-mapped is ideally planar (without even the smallest bumps), and 2) that it’s lambertian (i.e. viewpoint-independent). Unfortunately, many real-world surfaces – grass fields, asphalt pavements, pebble roads, sandy beaches – violate these assumptions. As a result, one often encounters characteristic stretching and flattening artifacts, especially at oblique angles. Although these artifacts are usually small, perceptually they are significant enough to substantially reduce the perceived realism, e.g. of a virtual environment, to be of major concern, particularly to game designers.

Several solutions to this problem have been proposed. In traditional computer graphics, people have used bump mapping [Bli78] and displacement mapping to introduce surface roughness. However, while attempts have been made to capture bump maps from real photographs, using photometric stereo [RTG97] or normal stereo [DTM96], the results have not been particularly encouraging. Others have taken the path of utilizing a dataset of textures taken under varying lighting and viewing conditions as a way to learn how a texture might look from a different viewpoint/lighting. For example, the now classic work of Leung and Malik [LM01] proposed to utilize the CuReT database [DNvGK97] to

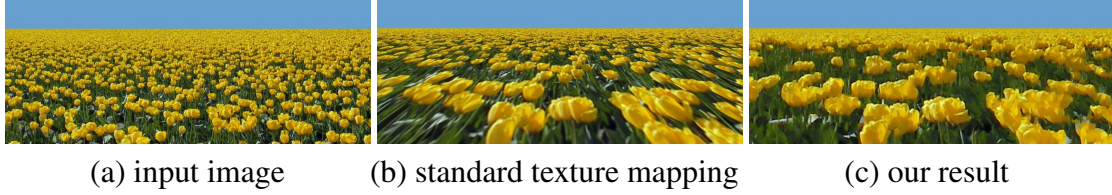


Figure 8.6: From the input image (a), moving the camera downwards, standard texture mapping produces a warped result (b), but our algorithm is able to preserve the correct three dimensional structure of the texture (c).

compute 3D textons across the stack of different views of the same texture. Each 3D texton represents how a particular type of geometric surface patch on the texture behaves under different conditions. When novel material is presented, the approach tries to first identify the corresponding 3D textons and then “rotate” them to generate a novel view/lighting of that material. Other approaches that follow in this general direction include [LYS01, VZ02, CD04, VT04]. While successful for some types of textures taken in a laboratory setting, all these approaches share the same weakness – they rely on the texture database to have very similar textures to the ones being processed. As a consequence, the methods have not been shown to work well, say, arbitrary photographs of outdoor scenes, such as the one on Fig. 8.6. At the same time, texture synthesis methods [EL99, WL00, HJO⁺01, EF01] have been shown to be successful at synthesizing prospectively-varying textures [KSE⁺03, DZP08] and adding surface detail [FJP02, MBG03], but have not, to our knowledge, been used to create novel views of texture.

We present a very simple but effective approach for synthesizing novel views of a given textured surface from a single photograph. The main idea is to exploit the fact that analyzed surface is, in fact, a texture. i.e. it is composed of a repeated stochastic structure. But because the surface is being viewed at an angle, different texture elements are being seen at different viewing angles. This allows us to use a single image as a statistical sample that should include the necessary data to synthesize novel views of that surface (since the outdoor illumination remains constant).

This same insight has been exploited before, in particular for sharing the parameters

of a reflectance model spatially across the surface [REB06, WZT⁺08]. What makes our method different is that it's non-parametric, and much more simple. Briefly, we start by assuming that the given surface is planar, and apply a homography to warp it to a novel view. We then apply non-parametric texture transfer [EF01] to inject the appropriate statistics while maintaining the correct structural appearance. Moving the camera in 3D space, we can predict how a patch would look like under a different viewing angle from a) its appearance under the original viewing angle (structural similarity) and b) the appearance under the new viewing angle of a different patch of the same texture (statistical similarity). The rest of the chapter will present the details of the algorithm and show some results.

8.4 View-dependent Texture Transfer

Let the original camera location (x_w, y_w, z_w) and tilt angle θ correspond to the input image I and consider a new camera location (x'_w, y'_w, z'_w) with new tilt angle θ' corresponding to a new image I' . Points on the image plane given by (x_i, y_i, f) , where f is the focal length of the camera, will be abbreviated with (x_i, y_i) . Points on the ground plane will be given by $(x_w, 0, z_w)$.

For a point in the new image (x'_i, y'_i) , we compute the viewing angle ϕ of its corresponding ground surface point $(x'_w, 0, z'_w)$ and find, by similar triangles, the point $(x_w, 0, z_w)$ on the ground surface that has the same viewing angle with respect to the original image I . For simplicity of the explanation and without loss of generality, consider camera movements along the $x_w = 0$ plane. The image points in I with the correct viewing statistics can again be found by similar triangles by setting $y_i = f \tan \phi$ and solving

$$x_i = \frac{\sqrt{f^2 + y_i^2} x_w}{\sqrt{(z_w - z_c)^2 + y_c^2}}. \quad (8.2)$$

where f is the focal length of the camera and $(x_c, 0, z_c)$ is the original camera position. In addition to exploiting viewing angles information, we also require that the transfer apply the correct scaling of the texture elements. The rescaling factor s is the ratio between the projected distances in the two images between two arbitrary known 3D points in the scene.



Figure 8.7: 3D texture synthesis results. For each input image (left column), we simulate camera movements by showing frames at different moments in time. The first row corresponds to the standard texture mapping, while the second displays our results. The camera path is chosen to be down and a bit forward. Our synthesis shows the correct 3D structure of the textures. The flowers towards the back become denser as we move downwards (yellow tulips), while it recedes and more green patches become visible as we move upwards (pink tulips). Both sets of flowers are scaled properly, giving the correct perception of the distance to the surface surface.

Now that the geometry of our problem is defined, creating a new view is a relatively straightforward application the texture transfer [EF01]. In Step 1, we assume that the surface is totally planar and create the new view by simply applying the appropriate homography transformation to the input image. In Step 2, we “fix up” the distorted warped image by transferring the texture from the original image onto it. We use the standard texture transfer algorithm, except for each patch location (x'_i, y'_i) in the new image, we only allow it to sample from the neighborhood of the corresponding angle location (x_i, y_i) in the original image (see previous section). The neighborhood is defined by a 2D Gaussian with σ related to the patch size (we used .6 in our experiments). Moreover, the neighborhood is resized by the scale factor s before matching to ensure that the correct texture scale is preserved.



Figure 8.8: 3D texture synthesis results. For each input image (left column), we simulate camera movements by showing frames at different moments in time. The first row corresponds to the standard texture mapping, while the second displays our results. The camera path is chosen to be down and a bit forward. Our synthesis shows the correct 3D structure of the textures. The other two cases, eliminate distortions otherwise present in the homography. Both the grass and the hay maintain a vertical overall structure as we move downwards and the overall appearance remains three dimensional.

The above algorithm works well for small camera movements, “repainting” a less-than-perfect homography warp to have the right texture statistics. However, for large camera movements, the homography becomes completely wrong, and the transfer algorithm is not able to “fix it”. We propose a simple but effective iterative solution: a large camera movement is divided into intermediate steps. For each step, we run the above algorithm, and use its result as the input image for the next iteration. This way, each time the algorithm needs to do only a little bit of fixing up, better preserving the correct structure of the input texture.



Figure 8.9: Synthesis results. We display the homography and our synthesis at incremental camera movements. The input images are displayed in the left column. While the aim of this paper is to synthesize textures composed or repeated elements, it is interesting to see that it can be applied in more general cases. The roof tops illustrate its potential. Although it fails in some parts of the image due to not enough samples of the same texture and elements of varying height within the input image, our synthesis is able to maintain the correct 3D appearance as the camera is moved downwards and improves significantly when compared to the distorted appearance given by the homography.

8.5 Experiments and Discussion

For each input image, we simulate incremental camera movements and display frames at different moments in time. The results are shown in Fig. 8.7 and Fig. 8.8. The synthesized results maintain the correct 3D appearance of the textures. The flower examples show an interesting phenomenon that is lost in the standard texture mapping. The flowers towards the back of the field become denser as the camera moves downwards (in the yellow tulips case), while the line of flower density change recedes and the camera moves upward and more green patches become visible (pink tulip scene). Both sets of flowers are scaled



Figure 8.10: Flowers automatically extracted with our segmentation approach which could be used as input for the synthesis application.

properly, giving the correct perception of the surface, showing a clear and significant improvement over the standard mapping. The other cases, eliminate distortions otherwise present in the homography. Both the grass and the hay maintain a vertical overall structure as we move downwards and the overall appearance remains three dimensional. In the case of images with the presence of textures, such as the flower one, our dot segmentation results could be used (Fig. 8.10) as input to provide better synthesis. While the aim of this paper is to synthesize textures composed or repeated elements, it is interesting to see that it can be applied in more general cases. In the penguin image in Fig. 8.9, the 3D structure is more visible towards the sides, where our texture transfer is able to keep the penguins in vertical position. The roof tops illustrate its potential for future work. Although it fails in some parts of the image due to not enough samples of the same texture and elements of varying height within the input image, our synthesis is able to maintain the correct 3D appearance as the camera is moved downwards and improves significantly when compared to the distorted appearance given by the homography.

Chapter 9

Conclusions

Extracting dots is a challenging image segmentation problem: faint boundaries and low contrast between regions, large intensity variations within the regions and conjoined clusters of dots make some dots hard to tease apart even upon close inspection. This thesis presents a constrained spectral graph partitioning framework to deal with the fine granularity of these small structures together with the ongoing challenge of dealing with the complexity associated with increasing image sizes. Our work makes several contributions:

- We successfully extract salient regions by exploring local image structure to capture 2D geometrical information in the local pairwise cues. We 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. As discussed for thin and long structures, an extension of our work is to handle more complicated geometries other than convexity.
- We introduce grouping cues of both attractive and repulsive natures. While repulsion from feature dissimilarity seems to encode the same attraction cue of feature similarity, as it operates at a larger spatial range, it plays an active and complementary role to local attraction. Popout would not be possible without repulsion.

- We decouple the challenge of fine segmentation granularity and image complexity in constrained patch segmentations. We include stitching constraints that naturally integrate within the spectral-graph framework of finding dots. The image is divided into many subpatches than can be independently run subject to stitching constraints between them, allowing constraints to propagate through even in the interior of the patches. This obviates segmentation complexity associated with larger images and emphasizes that for many medical images the ‘global’ information contained in smaller image patches is sufficient to properly segment the salient structures.
- While our method is designed to pop out all the dots in a microscopic image based entirely on the convexity flow field feature, we show the potential of our dot model to segment textons in real-scenes. We briefly present an algorithm for 3D synthesis, but the potential of using dot segmentation as input has yet to be studied. Similarly, many other shape from texture applications could benefit from an automatic texton detection.
- We explore shape extraction for larger objects within a contour grouping framework, by exploiting region information to establish the local cues in the 1D contour grouping approach in trying to extract large objects with faint boundaries. Intensities still play a role, but most importantly local cues encode region membership which allows even farther apart boundary points to correctly group together and form long salient contours.

This thesis represents a first step in studying how local geometry representation can be exploited in the local pairwise cues setup of spectral-graph approaches. We show how to automatically segment many small and thin structures in an image, regardless of its size and demonstrate the potential of repulsion cues for region extraction. Our ‘finding dots’ approach is robust to local intensity fluctuations and extracts regions of interest simultaneously, without user initialization, in one foreground with many disconnected components.

Bibliography

- [AMFM09] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, 2009.
- [BC04] Danny Barash and Dorin Comaniciu. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image Vision Comput.*, 22(1):73–81, 2004.
- [BGA⁺10] Ajay N. Basavanthally, Shridar Ganesan, Shannon Agner, James P. Monaco, Michael D. Feldman, John E. Tomaszewski, Gyan Bhanot, and Anant Madabhushi. Computerized image-based detection and grading of lymphocytic infiltration in her2+ breast cancer histopathology. *IEEE Transactions Biomedical Engineering*, 57(3):642–53, 2010.
- [BL79] S. Beucher and C. Lantuejoul. In *Proc. Worskshop on Image Processing, CCETT/IRISA*, pages 2.1–2.12., 1979.
- [Bli78] J. F. Blinn. Simulation of wrinkled surfaces. In *SIGGRAPH 78*, pages 286–292, 1978.
- [BPV⁺11] M.I. Bertoni, D.M. Powell, M.L. Vogl, S. Castellanos, A. Fecych, and T. Buonassisi. Stress-enhanced dislocation density reduction in multicrystalline silicon. *Physica Status Solidi - Rapid Research Letters*, 5(1):28 – 30, 2011.

- [BS08] Elena Bernardis and Jianbo Shi. Shape extraction through region-contour stitching. In *Int. Symposium Visual Computing*, volume 1, pages 393–405, 2008.
- [BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(11):1222–1239, 2001.
- [BY09] Elena Bernardis and Stella X. Yu. Robust segmentation by cutting across a stack of gamma transformed images. In *Energy Minimization Methods Comp. Vision and Patt. Recog.*, pages 249–260, Berlin, Heidelberg, 2009. Springer-Verlag.
- [BY10a] Elena Bernardis and Stella X. Yu. Finding dots: Segmentation as popping out regions from boundaries. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 199–206, 2010.
- [BY10b] Elena Bernardis and Stella X. Yu. Segmentation subject to stitching constraints: Finding many small structures in a large image. In *Medical Image Computing and Computer-Assisted Intervention*, volume 1, pages 119–126, 2010.
- [BY10c] Elena Bernardis and Stella X. Yu. Structural correspondence as a contour grouping problem. In *Proc. of IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, 2010.
- [BY11] Elena Bernardis and Stella X. Yu. Pop out many small structures from a very large microscopic image. *Medical Image Analysis*, 2011. To appear.
- [BZ03] Adrian Barbu and Song-Chun Zhu. Graph partition by swendsen-wang cuts. In *International Conference on Computer Vision*, 2003.

- [CA79] G.B Coleman and H.C. Andrews. Image segmentation by clustering. In *Proc. IEEE*, pages 773–785, 1979.
- [CBNC09] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Trans. Pattern Anal. Machine Intell.*, 31:1362–1374, 2009.
- [CBS05] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 1124–1131, Washington, DC, USA, 2005. IEEE Computer Society.
- [CD04] Oana Cula and Kristin Dana. 3d texture recognition using bidirectional feature histograms. *Int. Journal of Computer Vision*, 59(1):33–60, August 2004.
- [CGNT09] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watersheds: a new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *International Conference on Computer Vision*, 2009.
- [CSB08] Antonio Criminisi, Toby Sharp, and Andrew Blake. Geos: Geodesic image segmentation. In *European Conference on Computer Vision*, pages 99–112. Springer-Verlag, 2008.
- [CVR10] Jierong Cheng, Merlin Veronika, and Jagath C. Rajapakse. Identifying cells in histopathological images. In *Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, ICPR’10, pages 244–252, Berlin, Heidelberg, 2010. Springer-Verlag.

- [DAK⁺08] Olivier Duchenne, Jean-Yves Audibert, Renaud Keriven, Jean Ponce, and Florent Sgonne. Segmentation by transduction. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.* IEEE Computer Society, 2008.
- [DNvGK97] K.J. Dana, S.K. Nayar, B. van Ginneken, and J.J. Koenderink. Reflectance and texture of real-world surfaces. In *Proc CVPR*, pages 151–157, 1997.
- [DTCW01] Huseyin Tek Dorin, Hseyin Tek, Dorin Comaniciu, and James P. Williams. Vessel detection by mean shift based ray propagation. In *Proc. of IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 228–235, 2001.
- [DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96*, August 1996.
- [DZP08] Weiming Dong, Ning Zhou, and Jean-Claude Paul. Perspective-aware texture analysis and synthesis. *Vis. Comput.*, 24(7):515–523, 2008.
- [EF01] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH 2001*, 2001.
- [EL99] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.
- [FB03] Bernd Fischer and Joachim M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 25:513–518, April 2003.
- [FBX⁺09] Hussain Fatakdaawala, Ajay Basavanhally, Jun Xu, Gyan Bhanot, Shridar Ganesan, Michael Feldman, John Tomaszewski, and Anant Madabhushi. Expectation maximization driven geodesic active contour with

- overlap resolution (emagacor): Application to lymphocyte segmentation on breast cancer histopathology. In *Proceedings of the 2009 Ninth IEEE International Conference on Bioinformatics and Bioengineering*, BIBE '09, pages 69–76, Washington, DC, USA, 2009. IEEE Computer Society.
- [FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–81, 2004.
- [FH05] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61:55–79, January 2005.
- [FJP02] William T. Freeman, Thouis R. Jones, and Egon C Pasztor. Example-based super-resolution. *IEEE Comput. Graph. Appl.*, 22(2):56–65, 2002.
- [For01] David A. Forsyth. Shape from texture and integrability, 2001.
- [Gal00] Jean Gallier. *Geometric methods and applications: for computer science and engineering*. Springer-Verlag, London, UK, 2000.
- [GG90] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. pages 452–72, 1990.
- [GGP10] Felix Graf, Marcin Grzegorzec, and Dietrich Paulus. Counting lymphocytes in histopathology images using connected components. In *Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, ICPR'10, pages 263–269, Berlin, Heidelberg, 2010. Springer-Verlag.
- [GM96] Gideon Guy and Gérard Medioni. Inferring global perceptual contours from local features. *Int. J. Comput. Vision*, 20:113–133, October 1996.

- [Gra06a] Leo Grady. Fast, quality, segmentation of large volumes - isoperimetric distance trees. In *ECCV (3)*, pages 449–62, 2006.
- [Gra06b] Leo Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 28(11):1768–1783, 2006.
- [GSBB03] Meirav Galun, Eitan Sharon, Ronen Basri, and Achi Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *International Conference on Computer Vision*, 2003.
- [GZW07] Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu. Primal sketch: Integrating structure and texture. *Comput. Vis. Image Underst.*, 106:5–19, April 2007.
- [HAA97] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH '97*, pages 225–232, 1997.
- [HB95] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proc. of Int. Conf. Image Processing*, pages 229–238, 1995.
- [HBSB08] K. Hartman, M. Bertoni, J. Serdy, and T. Buonassisi. Dislocation density reduction in multicrystalline silicon solar cell material by high temperature annealing. *Applied Physics Letters*, 93, 2008.
- [HJO⁺01] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin. Image analogies. In *Proceedings of SIGGRAPH 2001*, 2001.
- [HLEL06] James Hays, Marius Leordeanu, Alexei A. Efros, and Yanxi Liu. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision*, pages 522–535. Springer, 2006.

- [HPB97] T. Hofmann, J. Puzicha, and J. M. Buhmann. An optimization approach to unsupervised hierarchical texture segmentation. In *Int. Conf. on Image Processing*, 1997.
- [HS85] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. In *Computer Vision, Graphics and Image Processing*, volume 29(1), pages 100–132, 1985.
- [Jac96] David W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Trans. Pattern Anal. Machine Intell.*, 18:23–37, January 1996.
- [Jul62] Bela Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.
- [JWR04] Andrei C. Jalba, Michael H. F. Wilkinson, and Jos B. T. M. Roerdink. Cpm: A deformable model for shape recovery and segmentation based on charged particles. *IEEE Trans. Pattern Anal. Machine Intell.*, 26(10):2004, 2004.
- [KGS11] Ryan Kennedy, Jean Gallier, and Jianbo Shi. Contour cut: identifying salient contours in images by solving a hermitian eigenvalue problem. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 2902–2909, 2011.
- [KSE⁺03] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, July 2003.
- [KSG10] Manohar Kuse, Tanuj Sharma, and Sudhir Gupta. A classification scheme for lymphocyte segmentation in h&e stained histology images. In *Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, ICPR’10, pages 235–243, Berlin, Heidelberg, 2010. Springer-Verlag.

- [LF04] Anthony Lobay and David A. Forsyth. Recovering shape and irradiance maps from rich dense texon fields. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, volume 1, pages 400–406, 2004.
- [LM96] Thomas Leung and Jitendra Malik. Detecting, localizing and grouping repeated scene elements from an image. In *European Conference on Computer Vision*, pages 546–555. Springer-Verlag, 1996.
- [LM98] Thomas Leung and Jitendra Malik. Contour continuity in region based image segmentation. In *European Conference on Computer Vision*, volume 1, pages 544–59, 1998.
- [LM01] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001.
- [LXGF05] Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D. Fox. Level set evolution without re-initialization: A new variational formulation. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 430–436. IEEE, 2005.
- [LYS01] Xinguo Liu, Yizhou Yu, and Heung-Yeung Shum. Synthesizing bidirectional texture functions for real-world surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 97–106, New York, NY, USA, 2001. ACM.
- [MAFM08] Michael Maire, Pablo Arbelaez, Charless Fowlkes, and Jitendra Malik. Using contours to detect and localize junctions in natural images. *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, 2008.
- [MBG03] Ryan M. Mert, Kavita Bala, and Donald P. Greenberg. Detail synthesis for image-based texturing. In *I3D '03: Proceedings of the 2003 symposium*

on *Interactive 3D graphics*, pages 171–175, New York, NY, USA, 2003. ACM.

- [MBLS01] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 2001.
- [Mey94] Fernand Meyer. Topographic distance and watershed lines. *Signal Process.*, 38(1):113–125, 1994.
- [Mey05] Fernand Meyer. Morphological segmentation revisited. *Space, Structure and Randomness, Springer*, pages 315–347, 2005.
- [MFM04] David Martin, Charless Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Machine Intell.*, 26(5):530–549, 2004.
- [MFTM01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, 2001.
- [MR97] Jitendra Malik and Ruth Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces, 1997.
- [MS89] David Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Math.*, pages 577–684, 1989.
- [MS97] R. Malladi and J. A. Sethian. Level set methods for curvature flow, image enhancement, and shape recovery in medical images. In *Proc. of Conf. on Visualization and Mathematics*, pages 329–45. Springer-Verlag, 1997.

- [MWTX03] Shyjan Mahamud, Lance R. Williams, Karvel K. Thornber, and Kanglin Xu. Segmentation of multiple salient closed contours from real images. *IEEE Trans. Pattern Anal. Machine Intell.*, 25:2003, 2003.
- [NWB03] Hieu Tat Nguyen, Marcel Worring, and Rein Van Den Boomgaard. Watersnakes: Energy-driven watershed segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 25:330–342, 2003.
- [NWvdBW03] Hieu T. Nguyen, Marcel Worring, Rein van den Boomgaard, and Hieu T. Nguyen Marcel Worring. Watersnakes: Energy-driven watershed segmentation, 2003.
- [OS88] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.
- [PBCL09] Minwoo Park, Kyle Brocklehurst, Robert T. Collins, and Yanxi Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1804–1816, 2009.
- [PRT10] Costas Panagiotakis, Emmanuel Ramasso, and Georgios Tziritas. Lymphocyte segmentation using the transferable belief model. In *Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, ICPR’10, pages 253–262, Berlin, Heidelberg, 2010. Springer-Verlag.
- [PS00] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, 2000.
- [REB06] Ravi Ramamoorthi, Sebastian Enrique, and Peter N. Belhumeur. Reflectance sharing: Predicting appearance from a sparse set of images of a

- known shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1287–1302, 2006. Member-Todd Zickler.
- [RTG97] Holly E. Rushmeier, Gabriel Taubin, and André Guéziec. Appying shape from lighting variation to bump map capture. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 35–44, London, UK, 1997. Springer-Verlag.
- [SG07] Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *International Conference on Computer Vision*, pages 1–8, 2007.
- [SKRV99] Joes Staal, Stiliyan Kalitzin, Bart Ter Haar Romeny, and Max Viergever. Detection of critical structures in scale space. In *Scale-Space Theories in Computer Vision*, pages 105–116, 1999.
- [SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):888–905, 2000.
- [THW⁺07] X.C. Tai, E. Hodneland, J. Weickert, N.V. Bukoreshtliev, A. Lundervold, and H.H. Gerdes. Level set methods for watershed image segmentation. In *SSVM07*, pages 178–90, 2007.
- [TLM08] Shin-Ya Takemura, Zhiyuan Lu, and Ian A. Meinertzhagen. Synaptic circuits of the drosophila optic lobe: The input terminals to the medulla. *The Journal of Comparative Neurology*, 509(5):493–513, 2008.
- [TZ06] Zhuowen Tu and Song-Chun Zhu. Parsing images into regions, curves, and curve groups. *International Journal of Computer Vision*, 69:223–249, August 2006.

- [US88] Shimon Ullman and Amnon Sha’ashua. Structural saliency: The detection of globally salient structures using a locally connected network. Technical report, 1988.
- [VF10] A. Vedaldi and B. Fulkerson. Vlfeat - an open and portable library of computer vision algorithms. In *Proc. of the 18th annual ACM international conference on Multimedia*, October 2010.
- [VKC07] M A Vollrath, K Y Kwan, and D P Corey. The micromachinery of mechanotransduction in hair cells. *Annual Review of Neuroscience*, 30:339–65, 2007.
- [VT04] M. Alex O. Vasilescu and Demetri Terzopoulos. Tensortextures: multilinear image-based rendering. In *SIGGRAPH ’04: ACM SIGGRAPH 2004 Papers*, pages 336–342, New York, NY, USA, 2004. ACM.
- [VZ02] Manik Varma and Andrew Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *ECCV ’02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 255–271, London, UK, 2002. Springer-Verlag.
- [WKS05] Song Wang, Toshiro Kubota, Jeffrey Mark Siskind, and Jun Wang. Salient closed boundary extraction with ratio contour. *IEEE Trans. Pattern Anal. Machine Intell.*, 27:546–561, April 2005.
- [WL93] Z Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. In *PAMI*, volume 15, pages 1101–1113, 1993.
- [WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 00*, pages 479–488, 2000.

- [WSC09] 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 *International Conference on Computer Vision*, Kyoto, Japan, 2009.
- [WZT⁺08] Jiaping Wang, Shuang Zhao, Xin Tong, John Snyder, and Baining Guo. Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Trans. Graph.*, 27(3):1–9, 2008.
- [XP98] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. on Image Processing*, 1998.
- [XS07] Bai Xue and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *International Conference on Computer Vision*, pages 1– 8, 2007.
- [YS01] Stella X. Yu and Jianbo Shi. Understanding popout through repulsion. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, 2001.
- [YS03] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *International Conference on Computer Vision*, 11-17 Oct 2003.
- [YS04] Stella X. Yu and Jianbo Shi. Segmentation given partial grouping constraints. *IEEE Trans. Pattern Anal. Machine Intell.*, 26(2):173–83, 2004.
- [Yu04] Stella X. Yu. Segmentation using multiscale cues. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, Washington DC, June 27 - July 2 2004.
- [Yu05] Stella X. Yu. Segmentation induced by scale invariance. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.*, 2005.
- [ZSS07] Qihui Zhu, Gang Song, and Jianbo Shi. Untangling cycles for contour grouping. In *International Conference on Computer Vision*, 2007.

- [ZSXZ10] Bo Zhang, Yongli Su, Yongfeng Xu, and Shuling Zhang. An adaptive geodesic active contour model. In *International Conference on Natural Computation*, pages 2267–2270, 2010.
- [ZWM98] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame) – towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.
- [ZY96] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multi-band image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 18:884–900, 1996.