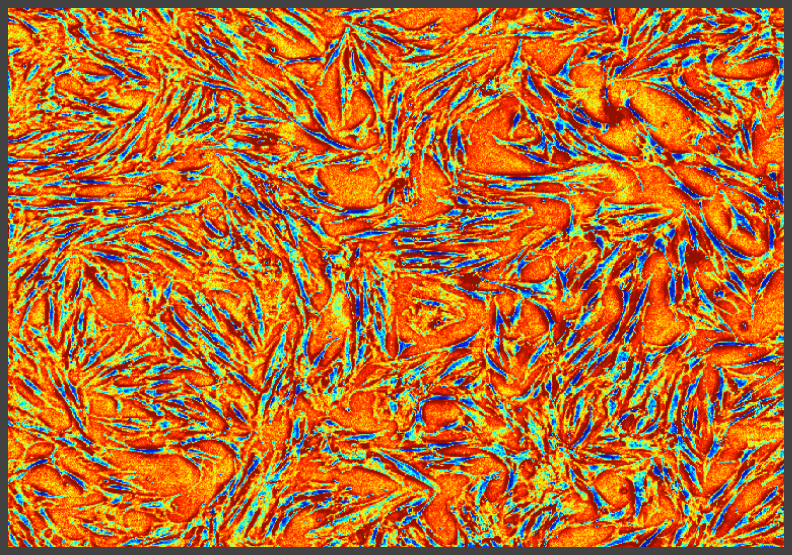
Active Mask Framework for Segmentation of Fluorescence Microscope Images



Gowri Srinivasa

bimagicLab
Center for Bioimage Informatics
Department of Biomedical Engineering
Carnegie Mellon University

Active Mask Framework for Segmentation of Fluorescence Microscope Images

Gowri Srinivasa

Advisor: Prof. Jelena Kovačević

Center for Bioimage Informatics
Department of Biomedical Engineering
Carnegie Mellon University Pittsburgh, PA 15213

Thesis Document

Submitted in partial fulfillment of the requirements towards the Ph.D. degree awarded by the Department of Biomedical Engineering, Carnegie Inst. of Tech., Carnegie Mellon University.

Thesis Committee Members

Prof. Jelena Kovačević (Advisor)
Departments of Biomedical Engineering and Electrical and Computer Engineering
Carnegie Mellon University

Prof. Matthew C. Fickus
Department of Mathematics and Statistics
Air Force Institute of Technology

Prof. Adam D. Linstedt
Department of Biological Sciences
Carnegie Mellon University

Prof. José M. F. Moura
Department of Electrical Engineering
Carnegie Mellon University

Prof. Robert F. Murphy
Departments of Biological Sciences, Biomedical Engineering, Computational Biology and Machine Learning
Carnegie Mellon University

मालासुधाकुम्भविबोधमुद्राविद्याविराजत्करवारिजाताम् ।
अपारकारुण्यसुधाम्बुराशिं श्रीशारदाम्बां प्रणतोऽस्मि नित्यम् ॥

श्रीसद्गुरुचरणारविन्दार्पणमस्तु

I always bow to Śrī Śāradāmbā,
the limitless ocean of the nectar of compassion,
who bears a rosary, a vessel of nectar,
the symbol of knowledge and a book in Her lotus hands.

Dedicated to the Lotus Feet of the revered Sadguru.

Abstract

This thesis presents a new active mask (AM) framework and an algorithm for segmentation of digital images, particularly those of punctate patterns from fluorescence microscopy.

Fluorescence microscopy has greatly facilitated the task of understanding complex systems at cellular and molecular levels in recent years. Segmentation, an important yet difficult problem, is often the first processing step following acquisition. Our team previously demonstrated that a stochastic active contour based algorithm together with the concept of topology preservation (TPSTACS), successfully segments single cells from multicell images.

In this work, we begin by presenting improvements to TPSTACS and highlighting some of the benefits of combining TPSTACS with a multiresolution approach for segmenting fluorescence microscope cell images. We also demonstrate the flexibility of the active-contour framework by developing algorithms for segmentation on different modalities, including DIC microscopy, MRI and fMRI. As a further improvement, we combine the active-contour framework with a multiscale transformation perspective to form the *multiscale active contour (MSAC) transform.* The need to overcome some of the limitations inherent to these active-contour-based frameworks while retaining their flexibility inspired the new AM framework.

The AM segmentation framework is suited for digital images, particularly for fluorescence microscope images. It is based on a local majority voting-based scheme, and can incorporate different forms of the voting function as well as several different functions to skew the voting to obtain a meaningful segmentation result. This framework has multiresolution and multiscale techniques built into it and can be instantiated to segment data of any dimension. We demonstrate the efficacy of the AM through an algorithm for segmenting punctate patterns of cells in fluorescence microscope images. While the theory opens up interesting vistas for research and development, the results demonstrate AM's utility in practice.

Acknowledgements

... I have stood on the shoulders of giants.

Isaac Newton

This thesis is seeing the light of the day due only to many a *giant*, thanking each of whom, in the detail so rightly deserved, would be a book in itself. It is with a pen restricted to but a few names and pages that I begin the most important chapter of the thesis.

It is with immense gratitude that I thank Prof. Jelena Kovačević for giving me the opportunity to pursue my graduate studies under her supervision. Her insightful guidance and encouraging support has marked every step of a graduate education that has been an enjoyable learning experience both professionally and personally. It is no exaggeration to say that not once have I walked in despondent to Jelena's office and not come out feeling upbeat and lighter. Whether it be such moral support or guidance on some aspect of our work, whether it be a critical review of a paper or design of a poster, suffice it to say, she has always been there for me and indeed wielded her "magic wand". Jelena has provided me with such opportunities that have extended beyond this project and highly enriched my graduate experience. Needless to say, Jelena has played a critical role throughout this project and it is to her inspiring leadership that this work owes its timely completion.

My sincere thanks to the committee members for acceding to be a part of my thesis committee. Their invaluable guidance has helped shape this work. A heartfelt thanks to Prof. Matthew Fickus—my *math advisor* this past year—-without whose help most of the work would not have been possible. Matt's creative genius and penetrating insight as well as good-natured tolerance of my inadequacies have made working with him great fun, a tremendous learning experience and the most fruitful research collaboration for me. My thanks to Mrs. Fickus as well for putting me up at their place, her sweet compliments on my culinary experiments and for proof-reading many of our papers.

Prof. Adam Linstedt provided the images that are the bedrock on which this work has been built. I am grateful to him for answering barrages of questions and for his encouraging

feedback during the intermediate stages. My thanks to Prof. José Moura for STACS, the starting point and inspiration for this work, and for his great sense of humor that made meetings fun. His valuable inputs sharpened our focus because of which this work has crystallized. A warm thanks to Prof. Robert Murphy for being a special part of my graduate experience, right from the qualifier through the thesis defense. It is Bob's generosity that got us started off with fluorescence microscope cell images and his considered inputs that have pushed the work, each time, to a higher level.

My sincere thanks to our other collaborators: Prof. A. F. Laine *et. al.* from Columbia for providing us with the cardiac MRI data and to Prof. B. A. Wandell and Dr. Michal Ben-Shachar from Stanford University for providing us with the brain fMRI data and for their active involvement especially during the teething stages of the project. My thanks to colleagues—Doru Balcan, Yusong Guo, Kang Li and Justin Newberg—for sparing their time even when they were hard-pressed for it as well as their valuable collaboration.

Thanks to Lionel Coulot and Heather Kirschner for their initial effort in adapting STACS to segment fluorescence microscope images and opening up vistas for further research (which kept me in business!). A very huge thanks to Shantanu Agarwal, JiaShu Chen, Eric Chu, Philip Cuadra, Siddharth Garg, Manuel Gonzalez, Sarah Hsieh, Ryan Kellogg, Gunhee Kim, Naoki Kimura, Vivek Oak and Sarah Reid for all their work that finds a place in this thesis as well as to all the other students I have had the pleasure of working with or serving as a TA. I have learnt much and carry with me indelible memories of these interactions.

I am grateful to my teammates from bimagicLab with whom it has been a privilege to work. I feel very lucky to have had the company of Amina Chebira and have profited immensely from her amazing perspective. I cannot imagine these four years without wise Amina and thank her profusely for making this journey so much more enjoyable. My thanks to Charles Jackson for painstakingly proof-reading many a document including this thesis, for his software tips and above all, for the regular and amazing reminder that it is actually possible to maintain a disciplined schedule even in grad school. My thanks to Inci Özgüneş and Ramu Bhagavatula for being sweet neighbors and to Aliaksei Sandryhaila and Xindian

Long and all the others for their inputs during our group meetings.

I have benefited from the counsel of many seniors and thank Estelle Glory, Pablo Hennings, Tad Merryman, Elvira Osuna, Shobha Venkataraman and Ting Zhao for making a giant leap seem but a small step. My thanks to my CBI labmates, CBI faculty, especially Profs. Markus Püschel, Gustavo Rohde and Stefan Zappe, and to the staff of CBI and BME for all their help. Many thanks to Sanna Gaspard for her camaraderie and help and to Neeti Gore, Smriti Gupta, Theta Maxino and other colleagues from ECE, BME and other programs for their fellowship. I thank my buddies through school and college (and their families), Dr. Ashwini Kamath, and many others who have played a role *backstage*.

My heartfelt thanks to Ms. Meenakshi Lakshmanan, Smt. V. Malathy Dwarakanathan and Ms. Bhavana Rao Potluri for the huge difference they have made and to the others from "the group" for their large hearts and all the fun. A special thanks to those whose exemplary attitude as well as words and actions have helped me get motivated.

Thanks to my extended family for just being there, especially my aunt and uncle in Lansing for all their help. A heartfelt thanks to all my teachers through school, preuniversity, undergraduate and graduate years for the impact they have had on my life and learning and to other elders for the influence their experiences have had on me and for their blessings. A warm thanks to my brother Jayanth for, among other things, sweetly accommodating my schedules and letting me enjoy my space and time through the years of being a roommate. Many thanks to my grandmothers who have been epitomes of love and pillars of strength and to my late grandfathers for their encouragement of my activities, academic and extra-curricular.

Most of all, my thanks to my parents for all the sacrifices they have made and the troubles they have put up with (and continue to!). Their undiminished support through times thick and thin has been my rock. These years they have ungrudgingly accommodated my not calling them as often as they would have liked. I will always cherish their understanding, their relentless trust and caring and their letting me choose my own path. I ever owe them a debt of gratitude.

Contents

Symbol	Notation
f	Original image
d	Dimension of f
S_m	mth segment of f
C_m	A closed curve which forms the boundary of S_m
χ_m	Characteristic function of S_m
ϕ	Level-set function in which $C = \bigoplus C_m$ is embedded
J_i	An energy functional associated with C
Υ	A function that detects features of interest in f
g	Lowpass filter
F_r	Region-based force
M_{in}	Statistics inside the current contour
M_{out}	Statistics outside the current contour
$\hat{M}_{\text{in}}$	Model statistics of the objects of interest
$\hat{M}_{\text{out}}$	Model statistics of the background
F_c	Curvature-based force
f_d	DNA image
f_p	Total protein image
η	Iteration number
w	Rectangular window (local neighborhood) around a pixel
M	Initial number of masks
K	Number of decomposition levels
K_0	Desired level of refinement
R	Region-based distributing function
α	Weight of R
β	Harshness of the threshold applied to compute R
γ	Average value of pixels at the foreground-background border
V	Voting-based distributing function
T	Sign function

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the focus in biological science has shifted to understanding complex systems at the cellular and molecular levels, a task greatly facilitated by fluorescence microscopy. Its success is due in part to the advent of a range of new fluorescent probes, including the nontoxic green fluorescent protein (GFP), used to tag proteins or molecules of interest. While fluorescence microscopes permit the collection of large, high-dimensional datasets, their manual processing is inefficient, not reproducible, time-consuming and error-prone, prompting the movement towards automated, efficient and robust processing to allow for high-throughput applications.

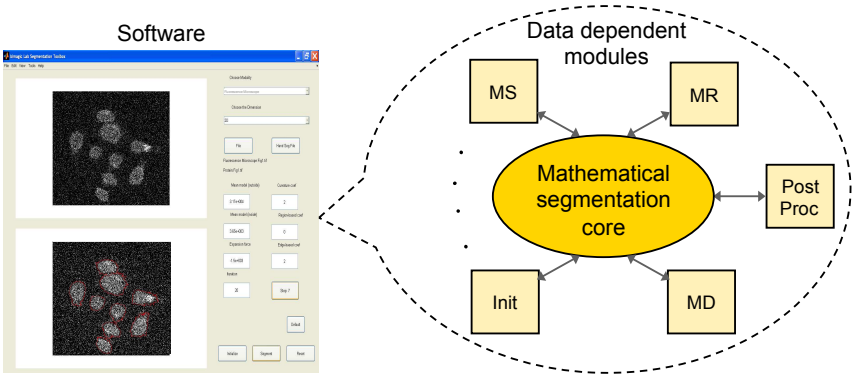


Figure 1.1: A diagram of the proposed research.

Segmentation, a fundamental yet very difficult problem in image processing, is often the first processing step following acquisition. Its aim is to separate objects of interest from other objects as well as from the background, and its output is either a collection of simple closed curves or a collection of masking functions. While it is always desirable for imaging tasks in biology to be as automated as possible, this is especially critical for segmentation, because it takes human experts anywhere from hours to days to segment by hand. The current segmentation algorithm used in fluorescence microscopy, the watershed algorithm—a region-growing method—is not designed to produce tight boundaries around the objects of interest. Meanwhile, state-of-the-art segmentation algorithms have only recently begun to be applied to this problem.

In this work, we present improvements to a stochastic active contour based algorithm designed to segment fluorescence microscope cell images. We demonstrate the flexibility of the frameowork by adapting it to segment images of various imaging modalities. Inspired by the success of the active-contour based framework and to overcome some of its limitations, we develop a mathematical framework suited to segment digital images. This, like the active-contour framework, can incorporate data-specific modules and thus has the potential to be adapted to other image types.

In summary, (see Fig. 1.1):

We present a new active mask framework particularly suited to the digital domain, and an instantiation of the framework to segment punctate patterns of fluorescence microscope cell images.

A mathematical core and data-dependent modules. As shown in Fig. 1.1, we endeavor to build a mathematical tool, that has at its core a flexible segmentation framework on top of which data-dependent modules may be added to instantiate the framework for a particular segmentation task. This idea is, to a large extent, inspired by the flexible framework of active-contour methods. The design reflects that the challenges of segmentation are different depending on the specimen, imaging modality, experimental conditions and application, but the underlying task is common to them all. So this framework is both

general, in that it is not limited to a certain application and yet, specific, in that it is not a "universal segmenter" but one that can be adapted to a specific application through the choice of data-specific modules and functions that drive the segmentation to be used with the core.

In particular we focus on the task of segmenting punctate[1] patterns of fluorescence microscope images and seek specificities of this data to inspire the design of functions that drive the segmentation. However, throughout this work, we demonstrate the advantage of flexibility by calling upon other applications to which this framework may be similarly extended.

Mathematical framework. Given a fluorescence microscope dataset, we need to efficiently extract features (such as local densities) in terms of the functions that aid in their segmentation. The punctate patterns of fluorescence microscope images motivates the use of multiscale transforms, known to extract intricate textures. Texture features, may facilitate the segmentation of these images [4]. Moreover, we would like to distinguish the foreground cells not only from their background but also from each other, without the use of external constraints. Above all, we would like to depart from the traditional formulation of active contour-based methods that are typically defined in the continuous domain. The framework we propose is suited especially for the segmentation of digital images, and in particular, the algorithm (choice of functions) we present is suited to segment punctate patterns of fluorescence microscope images. This is a large class of images in fluorescence microscopy.

Data-driven algorithmic development. Based on the nature of input, such as the number of channels and the dimensionality as well as the specific application, the processing demands different functionalities. While we have a powerful framework at the core, we need different modules to cater to the specific needs of the data. These modules are in the form of functions that drive the segmentation. The design of these functions is based on the characteristics of the objects of interest that we seek to segment. We present some of the

[1]Punctate refers to dotted patterns or patterns with very small holes. Such patterns form a large class of fluorescence microscope cell images, such as those of subcellular location patterns of proteins in cells.

data-specific modules and application-specific post-processing that could be included with the segmentation core through the needs of a few different applications that we consider.

Software. As tool developers, we look to the biological problem for insight to designing an algorithm and in turn, provide a tool that solves a computational task that is either intractable to be solved manually, and/or provides an effective alternative to manual processing so that resources can be better engaged. Consequently, in this process, biologists' feedback on the tool's performance is an important aspect to ensure the tool's utility. While the nature of the biological images at hand provide the inspiration and basis for the algorithm's design, it is the biologist's feedback that drives the algorithm's further development. Thus, to facilitate testing and further development, we provide the software to both biologists and algorithm developers.

1.1 Major Contributions

1. **Segmentation framework.** We present the perspective of a mathematical core that is general and can be adapted to different applications. This is possible through the choice of forces (or functions) and other data-specific modules that can be used with the core as different instantiations of the framework to drive the segmentation. We demonstrate this idea through adapting the stochastic active-contour scheme to segment images from various applications.

2. **Multiscale active-contour transform.** We combine the active contour framework with the multiscale transform perspective to improve the efficacy and efficiency of a level-set based formulation. This is particularly suited to applications in which traditional level-set based methods perform well.

3. **Active mask framework.** We present an iterative local-majority-voting based and local-averaging based framework suited to the segmentation of digital images and demonstrate it with the example of functions designed specifically to segment punctate patterns of fluorescence microscopy.

1.2 Thesis Outline

This thesis is organized as follows. In Chapter 2, we introduce the notion of segmentation and place it in the larger context of bioimaging applications. We discuss at length the need for automating segmentation and motivate the problem by introducing two of the applications in fluorescence microscopy for which our collaborators, Murphy et. al. and Linstedt et. al., both from CMU, require automated segmentation. To emphasize the broader impact of this work, we also highlight a few applications that use other modalities and require automated segmentation. The focus of these sections is to introduce the need for automated segmentation in each of these applications as well as the specific challenges posed by the data.

In Chapter 3, we review some of the segmentation methods in the literature and highlight some of the advantages and limitations of these methods [2]. In particular, we introduce the active-contour framework and discuss in detail an instance of this method, the stochastic active contour scheme (STACS), developed by Moura et. al. at CMU, as it forms the basis and inspiration of the methods we develop.

In Chapter 4, we describe how STACS was adapted to segment fluorescence microscope cell images. Up until Section 4.4, we present the previous work of bimagicLab (Dr. Kovačević's group) [21]. Thereafter, we present our improvements to this method in the context of a particular application in fluorescence microscopy.

In Chapter 5, we present how we adapt STACS to different imaging modalities. This flexibility of the framework, together with the specificity offered by data-specific modules forms the impetus for the present work. However, the traditional formulation assumes a continuous domain. We highlight some of the drawbacks of the method and transition towards a method more suited to digital images.

In Chapter 6 we present a multiscale active mask (MSAC) transform framework that combines the flexible active-contour framework with the powerful multiscale transforma-

[2] We note that the citations we provide are not exhaustive but only suggestive. Since the segmentation literature is vast, while we have tried faithfully to cite the related work at each relevant juncture throughout this thesis, it is possible we may have omitted some important and relevant method/reference or the other inadvertently.

tions. We demonstrate enormous speed ups of the algorithm as a result of this change in perspective. This also forms the first step towards a framework suited especially for the segmentation of digital images.

In Chapter 7, we present the new active-mask (AM) framework, the major contribution of this thesis. We motivate the need for moving away from the traditional contour view of segmentation towards a mask-based perspective. We discuss the design considerations and details of two distributing functions (analogous to forces in the active-contour framework) in the context of fluorescence microscope image segmentation. We build in topology preservation through the appropriate design of these functions and also include multiresolution and multiscale techniques as a part of the framework. We end this chapter by exploring some of the issues in proving mathematical convergence of the procedure.

In Chapter 8, we discuss the choice of parameters and evaluate the segmentation on a particular dataset. AM supersedes the active-contour-based algorithms for fluorescence microscope cell segmentation developed by bimagicLab. We compare the performance of AM segmentation with seeded watershed (SW)—the algorithm widely considered as most accurate in the fluorescence microscopy community—against the reference of manual segmentation. Results demonstrate that AM is very competitive with SW by both qualitative and quantitative measures of performance, and is thus a viable alternative to hand segmentation as well as SW and other active-contour based techniques. In line with our effort to extend AM to other modalities, we present preliminary results on applying the method to segment DIC stem cell images used for tracking. We conclude the chapter with a few pointers to one of the next steps in further developing the AM framework.

Chapter 2

Background and Motivation

In this chapter, we review some of the biomedical applications that rely on imaging, and motivate the need for automated processing of these images. In particular, we discuss the important role that fluorescence microscopy has assumed in biological research. We introduce the concept of segmentation and highlight the role it plays in two of the applications in fluorescence microscopy that we focus on as well as three other biomedical applications using different imaging modalities.

2.1 Biological Motivation: Automating Knowledge Extraction in Fluorescence Microscopy

In recent years, the focus in biological sciences has shifted from understanding single parts of larger systems (a vertical approach) to understanding complex systems at the cellular and molecular levels (a horizontal approach). Thus, there has been a revolution of "omics" projects, such as genomics, and now proteomics. Understanding complexity of biological systems is a task that requires acquisition, analysis and sharing of huge databases, and in particular, high-dimensional image databases.

The Advent of Fluorescence Microscopy. This task has been greatly facilitated by fluorescence microscopy, its success due in part to the advent of a range of new fluorescent probes used to tag proteins or molecules of interest, including the nontoxic, green fluorescent

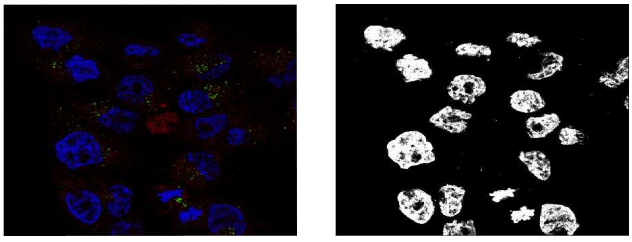
protein (GFP) [22, 23]. These sophisticated probes are capable of marking proteins or molecules of interest. There are several ways these probes can be directed to a target. For example, in *immunofluorescence*, antibodies (either primary or secondary) are used to transport fluorescent probes to the target protein (not suitable for live cell imaging). With *gene tagging* the blueprint for the GFP—the DNA sequence coding of the GFP (or one of its cousins)—is attached to the gene of interest. Traditional techniques such as staining [1] can also be used either in addition to the fluorescent probe or independently.

Fluorescence microscopes (see [24] for details) are then used to collect both 2D slices (for example, see Fig. 2.1(a)) and 3D volumes (z-stacks). Acquiring these images at multiple time instants results in 3D movies (2D time series) or 4D datasets (3D z-stacks). These microscopes also allow for imaging of multiple structures through multiple fluorescence channels. Biologists can use these techniques to collect images not only of multiple structures at multiple time points but also images across a span of resolutions and modalities (for example, electron microscopy and light microscopy), leading to enormous quantities of image data.

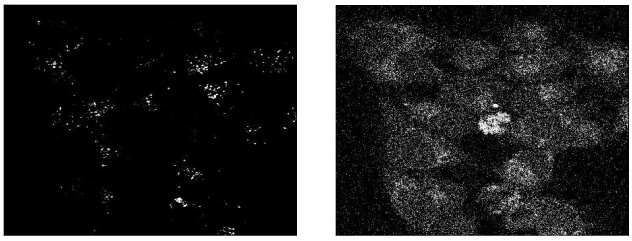
The Need for Automated Processing of Fluorescence Microscope Images. Visual processing of such a large number of multidimensional images is inefficient, not reproducible, time-consuming and error-prone, prompting the movement towards automated, efficient and robust processing of fluorescence microscope images. Moreover, some information hidden in the images may not be easily discerned by the human eye. Thus, we strive towards automated processing, not only for speed and efficiency, but to generate new knowledge through use of sophisticated algorithms as well. While such tools are widely present in clinical (medical) imaging, their use is not as widespread in imaging of biological systems at cellular and molecular levels. This is a huge challenge and requires integration of teams from such diverse fields as mathematics, signal processing, machine learning and biology.

A Must: Automated Segmentation. Segmentation is often the first step after

[1] Commonly used in vivo stains include DAPI and Hoescht for staining DNA and Rhodamine for staining Actin



(a) Multichannel image (b) Nuclear protein



(c) Specific protein (d) Total protein

Figure 2.1: A representative slice of a z-stack of HeLa cells imaged using a laser-scanning-confocal microscope [1]. (a) Three fluorescence channels (in pseudo color) superimposed. (b) Nuclear channel (total DNA stained with DAPI). (c) Golgi protein channel (protein in trans-Golgi network tagged using GFP-UCE). (d) Total-protein channel (labeled with LRSC). (Images courtesy of Dr. R. F. Murphy [2].)

acquisition. It is a fundamental yet very difficult problem in image processing, especially for microscope images that contain more than one cell. The goal of segmentation is to separate objects of interest both from other objects and from the background, and is used, for example, in high-content screening to identify cellular structures. The output of the analysis phase can be used as feedback to drive the acquisition, segmentation and any other intermediary processing phases to facilitate and enhance the analysis. Fig. 2.2 shows a block diagram of such a system.

As shown in the example in Fig. 2.2, often the output of a segmentation algorithm is a collection of simple closed curves, with one curve enveloping each object of interest. Alternatively, the result of segmentation may be regarded as a binary image, called a *mask*, taking the value 1 inside the objects of interest and 0 otherwise. While it is desirable for all tasks dealing with biological or biomedical images to be as automated as possible, this requirement is absolutely crucial for segmentation. For example, it takes a skilled expert one hour to segment an image (a z-stack of about 10 slices with 8-15 cells in each slice—one such slice is shown as in Fig. 2.1(d)) and almost three days for just the left hemisphere of

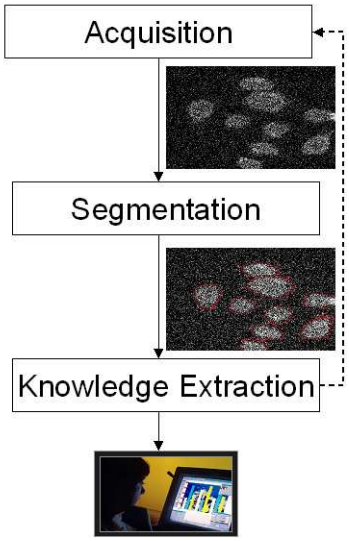


Figure 2.2: A diagram of the stages in a typical bio(medical) imaging system.

a brain MRI (left half of a z-stack of images such as the one in Fig. 2.7(b)).

While there has been an increasing effort to automate analysis of biological images, tools to meet the various challenges posed by specific applications in this area are still in their infancy [25]. When the size of the data is tractable, most often the analysis is performed visually; on the other hand, in large-scale studies, either semi-automated or automated techniques are used, trading off accuracy versus efficiency. For example, one of the most popular approaches to segmentation of cells in fluorescence microscope images is Voronoi-based and another algorithm, considered to be one of the most accurate for this task, is the seeded watershed. These approaches are beset with limitations that do not produce an accurate segmentation of the cells (see Sections 3.6.1 and 3.6.2).

There is a rich body of literature on segmentation algorithms developed for various applications in other fields. However, the nature of fluorescence microscope images is such that it does not allow for algorithms developed for applications such as multimedia to be used directly. This issue is discussed in some detail in Section 3.6.

In the following sections, we start by introducing two important biological problems for which our collaborators try to arrive at answers by using fluorescence microscope im-

ages. These problems require automated segmentation both to enable analysis, as hand segmentation of a sufficient number of images might not be feasible, as well as to ensure the reproducibility and accuracy of results. Although our focus is on segmentation of fluorescence microscope images, we consider a wider range of modalities, including problems using differential interference contrast microscopy (DIC), magnetic resonance imaging (MRI) and functional MRI (fMRI), to show the broader impact that automated segmentation would have.

2.1.1 Determining Protein Subcellular Location

Understanding the behavior and role of all proteins is a major current focus in the biological sciences, as it will help researchers map out complex systems, such as the nervous system, and will be critical in understanding how cells respond to injury, disease, stress, and aging. Crucial aspects of understanding a protein include understanding its structure, activity, function and its subcellular location. Such an understanding has several ramifications, including disease diagnosis and better drug design. Today's method of choice to determine protein subcellular location is fluorescence microscopy.

Murphy pioneered automated interpretation and analysis of protein subcellular location images, resulting in systems that can classify protein location patterns (see Fig. 2.3) with well-characterized reliability and better sensitivity than human observers [3, 4, 26, 27]. This work was followed by [28, 29].

Apart from improving the efficacy of image analysis, automation is inevitable for this application for various reasons: (*1*) The work on classification of protein patterns is just an initial step towards building generative models that would contribute to furthering our understanding of known proteins and to discovering new proteins/subcellular structures. (*2*) With data for over a million proteins and an estimate of more than two million proteins in just the human body [30], with proteins such as Actin known to have more than one role in the cell based on its localization and finally, with the subcellular localization of a protein likely being different in various diseases or stages of a single disease, location proteomics is a problem of enormous combinatorial proportions.

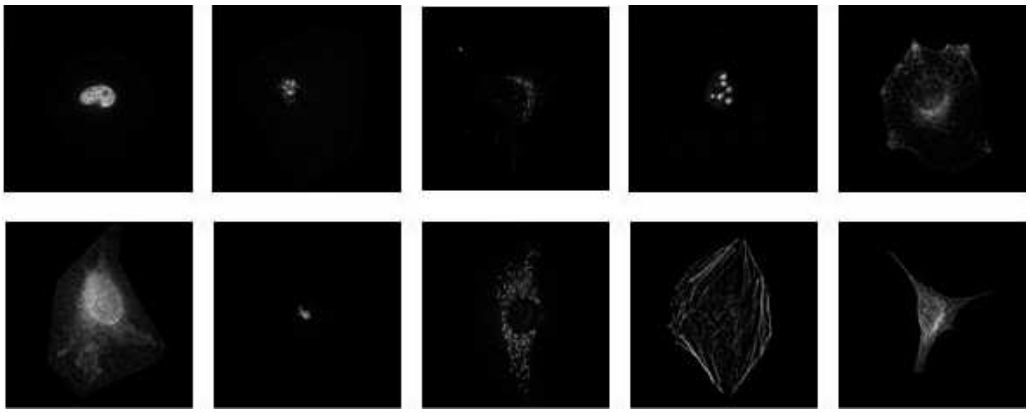


Figure 2.3: Images of subcellular localization of ten proteins [3]. Top row(L-r): DNA, Giantin, Lysosomal, Nucleolar and Tfr. Second row(L-r): ER, Gpp, Mitochondrial, Actin and Tubulin. The highest reported classification accuracy for these patterns is 95.4% [4]. (Images courtesy of Dr. R. F. Murphy [2].)

The Need for Automated Segmentation. In [3], the dataset contained parallel images for a specific protein, total protein and total DNA. Of the four data sets collected, on only one could automated segmentation be performed, as it contained a third fluorescent probe marking the total cell protein (in addition to the total DNA and the specific protein). Fig. 2.1 shows the three channels for an example image from this dataset. Although it is the specific protein channel (in this, case GFP) that is required for further analysis, the extent of the cell can be determined only from the total protein channel, as proteins in the cytoskeleton (cell periphery) also show up in this image. The nuclear channel has an unambiguous, bright pattern corresponding to each cell, which provides a good starting point. The segmentation outcome on the total protein channel can then be used to isolate each cell's specific protein localization pattern for further analysis.

Of the few studies which have used automated segmentation of fluorescence microscope images, the watershed algorithm is considered the most accurate method. Consequently, segmentation was performed using the seeded watershed algorithm on the total-protein channel, using the nuclei as seeds [31]. However, the method suffers from some limitations, especially in the context of cell segmentation applications such as this one, as we will see in Fig. 4.3. Watershed suffers from both splits (one cell/object is split into more than one) as well as merges (more than one cell/object are merged into a single one) when the initial seeds

are not accurately picked. Furthermore, by design the algorithm does not produce tight contours, causing the segmented cells to include a significant portion of the background. As a result of these drawbacks, a large number of cells are cannot be used (nearly 62%, see Table 4.1 as well as [13])[2] as such for further processing, wasting time and resources.

2.1.2 The Influence of Golgi-Protein Expression on the Size of the Golgi Apparatus

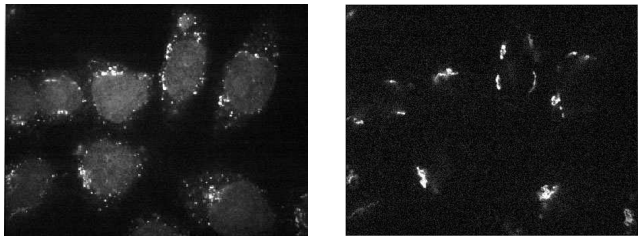
Eukaryotic cells are fundamentally distinguished from prokaryotes by the presence of compartments which optimize reactions by creating specialized environments, increasing surface area and dramatically increasing regulatory potential. Vesicle budding and fusion achieve protein exchange between biosynthetic secretory compartments and, in turn, these membrane trafficking reactions establish and maintain the secretory compartments. Defects in membrane trafficking are responsible for many human diseases [32] and our understanding of the molecular basis of these defects is paving the way to future effective therapeutics [33].

The Golgi body is one such membrane-bound organelle in eukaryotic cells whose cytoplasmic surface is a site for a number of important signaling pathways [34]. The Golgi body also mediates the processing and sorting of proteins and lipids in the final stages of their biosynthesis. The Golgi apparatus is subcompartmentalized and make it possible for the membrane and protein components to move through the organelle. The subcompartmentalized membranes are also responsible for the amazing capacity of the organelle to undergo rapid cycles of disassembly/reassembly in response to stress, cell division and even differentiation.

Linstedt and his group seek to understand how the Golgi body's underlying structures are established and maintained, how they are regulated in stress, and the purpose of each structural feature. That is, they seek a structure/function analysis from the underlying components of the organelle.

The Need for Automated Segmentation. The current hypothesis being tested is that the affinity of interaction between vesicle coat complexes and SNARE molecules (and

[2]We get 62% of cells are not usable based on the computation: 100 - round(max(Recall,Precision)).



(a) Cell image (b) Golgi image

Figure 2.4: (a) A HeLa cell image (COPII). (b) A HeLa Golgi image (giantin). (Images courtesy of Dr. A. D. Linstedt [5].)

other Golgi proteins) establishes and controls the size of Golgi compartments [35]. These tests depend on a quantitative ratiometric assay comparing Golgi size to cell size. The assay is fluorescence microscopy-based; immunofluorescence (secondary) is used to stain a component of one of the Golgi-body coat proteins as well as mark the Golgi body by labeling one of its proteins. The cells are imaged using a spinning-disk microscope. An accurate segmentation of the cell (as well as of the Golgi body) images is a critical step in determining cell (and Golgi-body) volume. Fig. 2.4 shows both the cell and corresponding Golgi channels for a representative slice from a z-stack of HeLa cell images used in this study.

To date, the experimental tests of this hypothesis have resulted in the discovery that direct interactions between the cytoplasmic domains of Golgi proteins existing in the Endoplasmic Reticulum (another membrane bound organelle in eukaryotic cells) and the COPII (a specific coat protein complex; this is a type vesicle that transports proteins from the Endoplasmic Reticulum to the Golgi Body) component Sar1p regulate COPII assembly, providing a variable exit rate mechanism that influences Golgi size [36].

Segmentation was carried out without automation and involved, for each cell, hand segmenting the cell boundaries in each of as many as 20 optical slices. For this reason, the analysis was limited to cells with flat morphology (fewer optical slices) and to small numbers of cells. To extend this finding, the Linstedt group will assay the role of Golgi protein/Sar1p interactions in Golgi size changes that accompany differentiation/dedifferentiation of the secretory pathway. Because a developmental time course will be analyzed and because the

cell types involved are not flat, the proposed experiments necessitate analysis of many more cells and much larger numbers of slices per cell. The automated segmentation algorithms to be developed here will allow a rigorous, objective and efficient means of Golgi/cell volume determination.

2.2 Importance of Automated Segmentation on Other Modalities

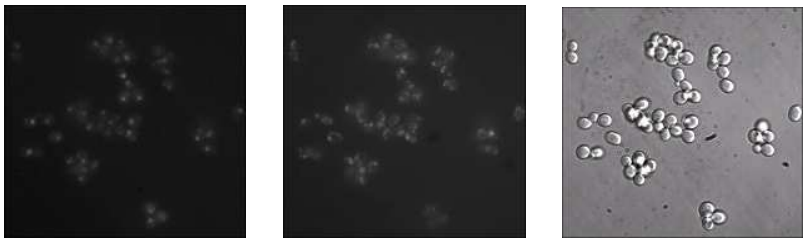
While we focus on developing the mathematical and algorithmic tools needed for segmentation of fluorescence microscope images, due to their flexibility we believe these tools will allow the development of algorithms for automated segmentation on other modalities as well. As representative examples, we discuss some of the projects we are involved in, including differential contrast microscopy (DIC), magnetic resonance imaging (MRI) of the heart and functional MRI (fMRI) of the brain, all applications for which our collaborators are looking for segmentation solutions.

2.2.1 DIC Images of of the Yeast

The yeast organism has become one of the most often used eukaryotic microorganisms for biological studies, leading to the "awesome power of yeast genetics" [37]. The sequencing of its genome was a tremendous help in the sequencing of the human genome. Moreover, yeast may be genetically manipulated with relative ease.

One of the standard ways of imaging yeast is using differential interference contrast (DIC) microscopy. It provides an excellent way for showing contrast in transparent specimens. DIC microscopy has many uses. One of the most important ones is that it represents an optical view of the entire cell and some of its parts, whereas any fluorescence label will exhibit a certain localization pattern (which may be restricted and could also vary over time). DIC microscopy does not require UV illumination (which may damage cells) and it works well on both live and fixed samples. Furthermore, it requires no special reagents (antibodies, cDNAs, dyes etc.).

The Need for Automated Segmentation. Segmentation is the first step in quantitative and automated studies of yeast [6, 38]. For instance, just as in the protein localization



(a) DAPI channel (b) GFP channel (c) DIC channel

Figure 2.5: A sample image showing three available channels from the yeast localization database [6]. (a) Total DNA marked by DAPI. (b) Specific protein tagged with GFP. (c) Budding yeast cells as seen under DIC. (Images courtesy of the Yeast GFP fusion localization database [7].)

study of HeLa cells, a GFP channel of randomly tagged specific proteins in the yeast is available. Due to the typically large number of these cells in each image (see Fig. 5.1(c)), hand segmentation is highly impractical and time consuming.

2.2.2 MRI Images of the Heart

MRI has emerged as a powerful technique to noninvasively visualize biological structures such as bone and soft tissues. It uses a powerful magnetic field and electromagnetic (radio) waves that are not ionizing (unlike x-rays) for the imaging. Hence, MRI is routinely used to evaluate structures of the heart, valves and major blood vessels. MRI images can be used to diagnose various cardiovascular diseases such as defects in the valves, myocardium (heart muscle) and blockages in the coronary artery as well as to monitor the progress of a patient who has been treated for a cardiovascular ailment.

Cardiac MRI is being used by Laine and his team [8] to understand heart wall motion and quantify the left-ventricular heart volume. The motivation for quantifying the left-ventricular volume is to be able to realize more accurate ejection fraction measures. Such measures facilitate the efficacy of interpretation of the images in relation to the cardiac output. As for understanding wall motion itself, the first step is to study normal hearts with the motivation of being able to understand anomalies in a diseased condition, and detect abnormal wall motion due to ischemia (which may eventually lead to an infarction or heart attack) and perhaps diagnose the condition in its early stages.

The Need for Automated Segmentation. The first step for either quantifying the

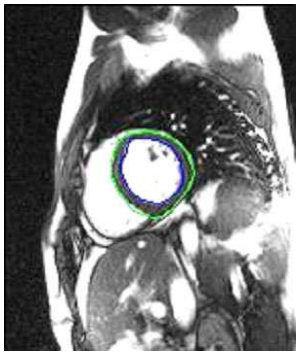


Figure 2.6: A 2D image from a human chest MRI showing a hand segmentation of the endocardium (in blue) and epicardium (in green) [8]. (Images courtesy of Dr. A. F. Laine [9].)

left-ventricular volume or understanding wall motion is to image a time series of the heart motion using MRI, and to isolate the inner heart wall (endocardium) and outer heart wall (epicardium) from the resulting images of a chest MRI (see Fig. 2.6. At least 25 frames at a given depth in the coronal direction at 125 time points is used to capture a single heartbeat for a study of this kind. Because the number of images in a time series of the chest MRI are large, it is necessary to automate the segmentation. An automated segmentation will also facilitate the subsequent visualization of heart wall motion. While there have been a few algorithms proposed for the segmentation of cardiac MRI images, a challenge is to develop a technique that will work on peripheral volumes, as well as on images from high-speed MRI that have poorer signal quality and that are beset with noise.

2.2.3 fMRI Images of the Brain

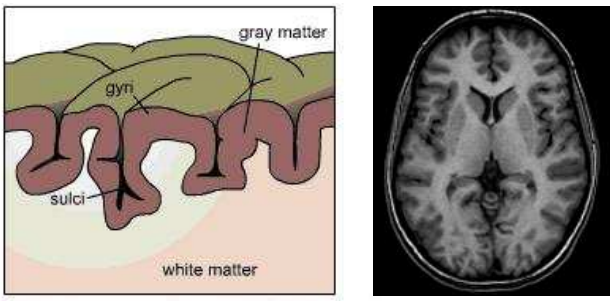
With the development of functional MRI techniques, it is now possible to directly visualize neuronal activity in the brain by using the amount of oxygen in the surrounding blood flow as a proxy for neuronal activity. This can be used to study the development, structure and activity of the human brain. In particular, the work of Wandell and his team focuses on using functional neuro-imaging to study visual pathways [39].

The Need for Automated Segmentation. Anatomical MRI images of the whole human brain are collected routinely at a resolution of about 1mm^3 in functional and anatomical imaging studies. Segmentation of these MRI brain images into gray and white matter,

or, tissue types, is crucial for multiple purposes, including: (a) measuring the location and size of visual field maps in human occipital cortex by overlaying fMRI signals on surface representations of the gray matter [40]; (b) studying the organization of retinotopic, motion and object sensitive cortex through 3D visualization of an inflated surface along the gray-white boundary of the brain [41], (c) mapping gray-matter growth and loss patterns in normal development and in neurological conditions, by computing deformation fields on the entire gray matter mask or on specific structures (for example, lateral ventricles, corpus callosum) defined by tissue type (respectively: CSF, white matter) [42], and many others. Thus, high quality tissue segmentation is a powerful tool that enables multiple lines of investigation in neuroscience [43].

In particular, to generate meaningful visualizations of the data obtained from fMRI studies for understanding visual pathways, it is important to first obtain accurate topological maps of the area of the brain in which the neuronal activity is being measured—the gray matter in the cerebral cortex. One way to create these topological maps is to segment 3-D MRI images of the brain into three regions: gray matter, white matter and cerebral spinal fluid. Gray matter forms the outer layer of the brain (called the cortex), completely encasing the inner white matter. The gray matter is highly folded to allow a higher surface area to volume ratio, and its topology resembles "two crumpled sheets having no holes or self intersections" [44]. Further, a large portion of the gray matter resides in the deep fissures, or, sulci, that surround the ridges on the surface of the brain. On the other hand, white matter forms the bulk of the deep parts of the brain, and is a large monolithic structure with no holes (see Fig. 2.7(a)). Finally, the cerebral spinal fluid (CSF) occupies the region between the cerebral cortex and the skull and appears in brain MRI images as a black region that surrounds the gray matter.

Once the 3-D topology of gray matter region has been identified by segmentation, an equivalent 2-D visualization can be created by flattening out the 3-D shape. The flattened 2-D representation makes it easier to visualize the deep furrows (sulcii) that exist in the gray matter. As noted earlier, accurately hand segmenting half a hemisphere of the brain



(a) Brain structure (b) An axial slice

Figure 2.7: (a) A schematic cross section of the human brain showing the gray matter and white matter [10]. (b) A 2D image from an axial section showing the left and right hemispheres of the brain. (Images courtesy of Dr. B. A. Wandell and Dr. M. B.-S. Chechik [11].)

along one of the axes such as the axial section (a section of which is shown in Fig 2.7(b)), takes an expert about 3 days. For all these reasons, automating the segmentation process is of immense value.

2.3 Application and Algorithm—A Symbiosis

In all of the applications discussed above, we see that computation, and in particular segmentation, plays a crucial role in interpreting the images produced in the acquisition phase. The acquisition phase in Fig. 2.2 subsumes the preparation of the data for the imaging process as well as the process of imaging itself (and any intermediate steps involved in digitizing the image). The segmentation and knowledge extraction phases are part of the image processing phase.

In studies such as those described above, no step of this system can function in isolation. The quality of the segmentation directly impacts the quality of the outcome of the knowledge extraction phase. The outcome of the knowledge extraction phase (class labels in case of pattern recognition, cell volume in case of a volumetric assay, cell number (or motion) in case of a proliferation study, etc.) is examined by the biologists and this in turn drives the further steps of their research, which may influence the acquisition phase. In turn, the considerations of the acquisition phase (modality, dimensionality, specific application etc.) drive the design of the image processing phase. In addition to this, the accuracy of the results

obtained in control experiments, for example, enable the biologists to provide feedback to the algorithm developers regarding the performance of the image processing phase. This feedback serves as the impetus for further development of the algorithm. It is, therefore, an iterative process as shown by the feedback loop in Fig. 2.2, with each phase influencing the other. Consequently, the hallmark of a successful study is these areas is the synergetic combination of knowledge from various domains. We take into account the importance of the role of the feedback from our collaborators in making the tools we design useful for them. Thus, we endeavor to make the segmentation algorithms we design accessible to our end users and available for testing by other algorithm developers to facilitate the process of testing and further development.

Chapter 3

Overview of Segmentation Methods

As highlighted in the previous chapter, segmentation is a fundamental task in image processing. It is the process by which an image is divided into its parts and background [45]. Since we discuss the segmentation of real-valued discrete images $f \in \mathbb{R}^d$, where d is the dimension, of size $D_1 \times D_2$ for $d = 2$ (and similarly for higher dimensional datasets), segmentation may be defined as the mapping $\mathcal{S} : f \mapsto S$, where $S = \bigoplus S_m$ and each $S_m, m \in \{1, 2, ..., M\}$ is a bounded open subset of $\mathbb{R}^d$ and represents a segment of the image. Corresponding to each S_m there is a closed curve, C_m, which is the boundary of S_m. The curves are such that any pixel in an image f belongs to one and only one segment S_m, where C_m encloses an object in the foreground and one of the segments is arbitrarily selected to represent the background. Thus, the segments S_m span image f.

The set S_m may be represented as a black and white mask or characteristic function χ of the same size as the input image; the white regions correspond to the objects of interest and the black to the background. Another representation of S_m may be a multi-hued mask ψ_m with a different color m assigned to each object (members of a distinct curve C_m) and black to the background.

In case of segmentation of biological images, $\mathcal{S}$ is a good mapping if the resulting S_m

are biologically meaningful segments of the input image. The quality of an automated segmentation scheme $\mathcal{S}$ is usually measured in terms of its similarity to the result of manual segmentation, what is commonly referred to as *ground truth*.

Efforts to automate segmentation have been ongoing for more than three decades [46,47]. Automated segmentation algorithms began with simple thresholding operations and have since come a long way [48,49]. As the different imaging modalities and the different applications for which the segmentation is required present highly specific challenges, particularly in biomedicine, there is no "universal segmenter"; a solution developed for one problem often cannot be applied to another without compromising segmentation accuracy [50]. This specificity has resulted in a vast body of literature. Segmentation algorithms can be classified broadly into edge-based, region-based and hybrid methods. In the sections below, we briefly review these methods.

3.1 Edge-Based Segmentation Methods

An edge is typically the boundary between the object and its background [51]. Edge-based segmentation methods may be further categorized as follows:

Gradient-based edge detection. The edge strength is the magnitude of the gradient and the edge direction the angle of the gradient of the image. The gradient is commonly estimated by operators such as the Sobel and Prewitt approximations to the first derivative [52]. Thresholding is usually the next step in obtaining the edges from the gradient of the image. The edges detected after thresholding are usually in the form of ridges. Thus, in a model involving an edge map, the edge point is usually considered to be the point whose strength is locally maximum along the direction of the gradient that reduces the ridges. Occasionally, edge-preserving filters may be used to enhance the edge detection [53].

Laplacian-based edge detection. In this method, an initial smoothing operator is first applied to the image to filter out the noise. However, this dilutes the edges. In order to enhance them, the second derivative of the image is taken. This has been empirically found to be a superior method to the gradient-based edge detection [54].

Canny edge detection. This is one of the most widely used edge-detection techniques [55]. It involves first smoothing the image and then computing the magnitude and angle of the gradient. This is followed by nonmaximum suppression that involves computing the zero-crossings of the second directional derivative in the direction of the gradient. Finally, a two-way threshold is applied to detect the edge points. All pixels greater than or equal to a threshold value T_1 are presumed to be edge-pixels. Likewise, any pixel connected to an edge-pixel with a value greater than or equal to threshold T_2, are also presumed to be edge-pixels. While the initial smoothing helps eliminate spurious edges, the two-way threshold helps detect weak edges in the image that may have otherwise been neglected.

Edge-based methods are usually quick to compute. The limitations of edge-based segmentation are that it is sensitive to noise, involves the selection of an edge-threshold which has to be empirically determined to work well and that the edge may not fully enclose the object.

3.2 Region-Based Segmentation Methods

Region-based methods segment an image based on some properties such as the mean, variance or textures of the object(s) of interest.

Pixel-level properties. A region-based method may distinguish the object of interest based on pixel similarities and pixel differences [56]. The advantage of this method is that gaps produced by missing edge pixels are not an issue. The disadvantage is that decisions about region membership are often more difficult than applying an edge detector.

The region-growing method first picks a seed pixel p and computes a similarity measure $\xi(p, q)$ where q is another pixel in the image. It adds pixel q to pixel p's region if and only if $\xi(p, q) > T$, where T is some threshold, fixed *a priori*. The design of the similarity measure $\xi(\cdot)$ may be based either on the average intensity over a neighborhood of size w around each pixel, or on the gradient or geometric properties. Adding a pixel to a region may be done in several ways. Comparing the original pixel directly to the seed pixel is easy but makes the region produced very sensitive to the choice of the seed pixel. Comparing to a neighbor

in the region is another option. This is not sensitive to the choice of the initial seed but the transitive closure of similarity might cause a significant drift.

Neighborhood properties. An alternative to pixel-level comparisons is comparing some property of a pixel or properties of a small neighborhood of a pixel to some model region statistics [56]. A pixel's properties (or the properties of a small neighborhood of the pixel) could be compared with the aggregated statistics of the current region. This is also referred to as centroid region growing [57].

A prominent example of the region-based methods is the *watershed* algorithm, discussed in detail in Section 3.6.2. With watershed segmentation, gaps produced by missing edge pixels are not an issue; however, decisions about region membership are often more difficult than applying an edge detector (for a critical comparison of several flavors of the watershed algorithm, see [58]). Other examples of region-based segmentation methods include statistical clustering, region merging and Markov-random-field-based methods [59,60]. Although region-based methods produce connected regions, decisions about region memberships are often ambiguous and it is usually nontrivial to set the threshold values to delineate one object from another or to delete the object from the background.

3.3 Active Contour Segmentation Methods

Active contours are a flexible and adaptive class of algorithms that evolved from the need to find a method that looks for any shape in the image that is smooth and forms a closed contour around it. Even though active-contour segmentation can be viewed as a hybrid between the edge-based and region-based forces, its flexible framework allows the incorporation of other techniques such as multiscale methods and graph partitioning, which may be beneficial for a specific application.

In general, active-contour segmentation is formulated as an energy minimization problem that involves evolving a curve towards the boundaries of objects of interest. As the curve *evolves* and connects points at the same energy level, it is called an active contour. The energy minimization problem is mapped to a PDE that describes the evolution of the

contour. The evolution is expressed in terms of forces so that the contour is comparable to an elastic string that moves according to two kind of forces: internal and external [61–65]. External forces are those derived from the image to segment (for example, based on edge detection). Internal forces are determined from the intrinsic geometric properties of the contour, such as its curvature [66,67]. The contour can be tracked using either a Lagrangian or Eularian solution to the PDE, which gives rise to either a parametric (in which case the curve is parameterized) or geometric (in which case the curve is evolved using the level-set embedding) implementation of the active contour.

3.3.1 Parametric Active Contours

Parametric active contours are also called parametric deformable models. They can be understood with the famous example and first design of this class of algorithms, "snakes" [61]. In this method, a parameterized curve ("snake"), $C(s) : [0, 1] \mapsto \mathbb{R}^2$, is initialized close to the object boundary. An energy functional J is associated with the curve: $J_{\text{snake}} = J_i + J_e$. The internal energy J_i of the curve depends on the intrinsic properties of the curve and could be the sum of its elastic energy (which discourages stretching) and bending energy (sum of the squared curvature of the contour):

$$J_i = \alpha \int_0^1 |C'(s)|^2 \mathrm{d}s + \beta \int_0^1 |C''(s)|^2 \mathrm{d}s,$$

where α and β are positive parameters. The external energy J_e, of the curve is defined as

$$J_e = -\lambda \int_0^1 |\nabla f(C(s))|^2 \mathrm{d}s,$$

so that it takes on smaller values at the features of interest such as boundaries. The quantity $|\nabla f|$ in J_e can be replaced by any function Υ of f that is an edge detector of the objects of interest in f.

The curve that minimizes J_{snake} is given by the Euler-Lagrange equation,

$$\alpha C''(s) + \beta C''''(s) - \nabla \Upsilon(C(s)) = 0. \tag{3.1}$$

Thus, the variational problem is mapped to a PDE. This is solved numerically by introducing an artificial time parameter t that parameterizes C along with s and rewriting (3.1) as

$$\frac{\partial C}{\partial t} = \alpha C''(s) + \beta C''''(s) - \nabla \Upsilon(C(s)). \tag{3.2}$$

This can be interpreted as the action of two forces, internal and external (corresponding to the respective energy functionals), based on which the snake starts deforming and moving towards the desired object boundary. Gradient descent can be used to iteratively solve for the zero of (3.2).

While snakes have the advantage of being tuned to be highly accurate, they have the drawback of self-intersections. Further, they are not amenable for simultaneously segmenting multiple regions of interest as they require the splitting and merging of multiple parameterized curves, necessitating reparameterization in case of topological changes. Finally, snakes are not good for 3D segmentation and are very difficult to generalize (to segment higher-dimensional data).

To overcome these drawbacks, we may geometrically embed the contour as the level set of a higher-dimensional function [68]. The boundary of an object of interest is then given by that level set [69].

3.3.2 Geometric Active Contours

A geometric active contour is also formulated as a variational problem that involves energy minimization of an energy functional composed of internal and external energy terms, much like the parametric active contour described above. The hallmark of a geometric active contour is a higher-dimensional embedding of the contour; the *level set* forms this higher-dimensional embedding of the contour of interest. Given an image $f(x)$, the level-set function $\phi : \mathbb{R}^d \to \mathbb{R}$, and a contour $C = \{x \in \mathbb{R}^d \big| \phi(x) = 0\}$, $\phi(x)$ is positive inside C, zero on C and negative outside C [70]. To evolve the contour C, we evolve the level-set function itself. Akin to parametric active contours described above, evolution of the level-set function is cast as an energy minimization problem that is solved by introducing an artificial

time parameter t and computing the solution to the Euler-Lagrange equation:

$$\frac{\partial}{\partial t}\phi(x,t) \;=\; \sum_i \lambda_i F_i(x,t)|\nabla\phi(x,t)|, \tag{3.3}$$

where F_i is a driving force and λ_i its weight.

The level-set formulation presupposes an initial curve, which is then evolved based on (3.3). In a signed-distance interpretation of the level-set function, the forces are computed only at points on the current contour whereas the level-set function is defined over the entire image domain. Thus, the level-set function is updated at each iteration based on the so called velocity-extension function to maintain the integrity of the level-set function [68].

One of the advantages of using the level-set embedding is that C is not parameterized, and in fact, one does not need to keep track of the number and shape(s) of the objects being tracked as the contour is obtained at any time as the zero level set of the level-set function. As an example, a geometric active-contour model based on the mean curvature motion is given by [71],

$$\frac{\partial}{\partial t}\phi \;=\; \Upsilon(|\nabla f|)|\nabla\phi|\left(div\frac{\nabla\phi}{|\nabla\phi|}\right)+\nu,$$

where

$$\Upsilon(|\nabla f|) = \frac{1}{1+|g_\sigma(x,y)*f(x,y)|^p},$$

$p=2$ is an edge-detector of f with Gaussian $g_\sigma(x,y) = 1/\sqrt{\sigma}\left(\exp\left(-|x^2+y^2|/4\sigma\right)\right)$, used to suppress noise and $\nu > 0$ is some constant. In this case, an initial curve embedded in ϕ moves in the normal direction with speed $\Upsilon(\nabla f)(curv(\phi)(x,y)+mean(\phi))$ and stops at the desired boundary where ϕ vanishes and $\nu > 0$ is a constraint on the area inside the curve and increases the propagation speed [65]. Another well-known example is the level-set formulation of the geodesic active-contour model [66].

While the examples mentioned above rely on edge information in the image to stop the active contour at the boundaries of the objects of interest, edges are often too noisy or weak leading a contour to overshoot the desired boundary. Thus, edge-based information may be supplemented (or in the absence of edge information in an image, replaced) with

region-based information such as statistics based on pixel intensities. A classic example of a region-based geometric active contour is the Chan-Vese model [65].

The key to active contours' or level-set based techniques' flexibility lies in the fact that F_i's may be selected based on the data or application under consideration and suitably balanced.

A recent example of this class of algorithms that combines different forces to segment images with low contrast, and outperforms other similar methods on a specific dataset is the stochastic active contour scheme that we describe in the following section.

3.4 Stochastic Active Contour Scheme

Stochastic Active Contour Scheme (STACS) belongs to the class of geometric active contour algorithms and it was originally developed to segment MRI images of transplanted mice hearts in the thigh muscle of mice [67]. The particular application posed the challenge of low contrast, as the heart (a muscle) was attached to the thigh (another muscle), which would cause most algorithms (even active-contour based) to not stop at around the desired location. Besides, the heart chambers were also required to be segmented, which made a perfect case for using a level-set based approach to be able to adapt to topological changes.

As a key concept, STACS maps the segmentation problem into an energy functional minimization problem [65, 67, 70],

$$\text{minimize: } J(\mathbf{C}) = \lambda_1 J_1(\mathbf{C}) + \lambda_2 J_2(\mathbf{C}) + \lambda_3 J_3(C) + J_4(\mathbf{C}), \tag{3.4}$$

where $\mathbf{C} = C(x, y)$ is the contour, $J_1(\mathbf{C})$ incorporates the model matching requirement and is called the region-based term; $J_2(\mathbf{C})$ is an edge-based term; $J_3(\mathbf{C})$ incorporates the prior knowledge on the shape of the contour; $J_4(\mathbf{C})$ is the contour smoothing term; and λ_1, λ_2, λ_3 and λ_4 are parameters that control the relative strength of J_1, J_2, J_3 and J_4, respectively.

The terms on the right hand side of (3.4) are detailed hereafter.

The model-matching term, J_1. This term is used to force the contour to partition the image into two regions with distinct (and homogeneous) statistics. One of the regions forms

the object(s) of interest, with the statistical model $\mathcal{M}_\infty$ and the other, the background, with the statistical model $\mathcal{M}_\in$. The contour is moved around by attempting to minimize the function

$$J_0(\mathbf{C}) = p(\mathbf{u}|\mathbf{C}, \mathcal{M}_\infty, \mathcal{M}_\in),$$

where $p(\mathbf{u}|\mathbf{C}, \mathcal{M}_\infty, \mathcal{M}_\in)$ is the joint probability density function of the image intensities $\mathbf{u}$ given the contour $\mathbf{C}$ and models $\mathcal{M}_\infty$ and $\mathcal{M}_\in$.

Let p_1 and p_2 be the probability density functions (pdf's) of the models $\mathcal{M}_\infty$ and $\mathcal{M}_\infty$ respectively. Since the intensities of the pixels outside and inside the contour $\mathbf{C}$ are statistically independent, the equation above can be written as $J_0(\mathbf{C}) = p_1(\mathbf{u_1}|\mathbf{C})p_2(\mathbf{u_1}|\mathbf{C})$. Taking the negative log of the above gives

$$J_1(\mathbf{C}) = -\ln(p_1(\mathbf{u_1}|\mathbf{C})) - \ln(p_2(\mathbf{u_2}|\mathbf{C})),$$

and maximizing $J_0(\mathbf{C})$ is equivalent to minimizing $J_1(\mathbf{C})$.

Suppose that the intensities of all pixels u_{ij} within each region are statistically independent. Then, when contour $\mathbf{C}$ is embedded as the zero level of the level set function ϕ, the equation for $J_1(\phi)$ becomes

$$J_1(\phi) = \int_\Omega -\ln[p_1(u(x,y))]\mathcal{H}_\epsilon(\phi(x,y)) - \ln[p_2(u(x,y))][1-\mathcal{H}_\epsilon(\phi(x,y))]\mathrm{d}x\mathrm{d}y,$$

where $\mathcal{H}_\epsilon(\phi(x,y))$, is the regularized Heaviside function representing the pixels within the contour, $1-\mathcal{H}_\epsilon(\phi(x,y))$ is the function representing the pixels outside the contour and the integral is taken over the entire domain Ω of the image. The original formulation of STACS for MRI images of transplanted hearts in mice assumes the object and background to be Gaussian [72].

The edge-based term, J_2. This term forces the contour to search for the prominent

edges in the image $u(x, y)$ by minimizing the edge map $\Upsilon(x, y)$ along the contour $\mathbf{C}$,

$$J_2(\mathbf{C}) = \int_{\mathbf{C}} \Upsilon(x, y) \mathrm{d}s,$$

where ds represents the infinitesimal Euclidean arc length of the contour $\mathbf{C}$, and $\Upsilon(x, y)$ is an edge map derived from the original image $u(x, y)$. The simplest way to obtain the edge map is to take the gradient magnitude of the image $u(x, y)$, i.e., $\Upsilon(x, y) = -|\nabla g_\sigma * u(x, y)|^2$, where g_σ is the 2D Gaussian kernel with variance σ^2, ∇ is the gradient operator and $*$ is the 2D convolution operator.

When $\mathbf{C}$ is embedded in the level set ϕ, we get

$$J_2(\phi) = \int_{\mathbf{C}} \Upsilon(x, y) |\mathcal{H}_\epsilon(\phi(x, y))| \mathrm{d}x\mathrm{d}y = \int_{\mathbf{C}} \Upsilon(x, y) \delta(\phi(x, y)) |\nabla \phi(x, y)| \mathrm{d}x\mathrm{d}y.$$

Shape prior, J_3. For the specific problem addressed in [72], this term incorporates an ellipse shape into the model. In general, this term could incorporate any (parameterizable) shape that we desire the contour to approximate. Let $\mathbf{C}_H(\theta)$ be the parametric form of the desired shape with parameters of the shape in vector θ. Forcing the shape of the evolving contour $\mathbf{C}$ to resemble the contour of shape $\mathbf{C}_H(\theta)$ is done by minimizing the squared distance of the pixels on the contour $\mathbf{C}$ to the contour of the desired shape $\mathbf{C}_H(\theta)$. In other words, we minimize $J_3(\mathbf{C}) = \int_{\mathbf{C}} D^2 ds$, where $D(x, y)$ is the appropriate distance metric.

Embedding $\mathbf{C}$ as the zero of the level set function $\phi(x, y)$, we have

$$J_3(\phi) = \int_\Omega D^2(x, y) \delta_\epsilon(\phi(x, y)) |\nabla \phi(x, y)| \mathrm{d}x\mathrm{d}y.$$

Contour smoothing, J_4. This term is used to ensure the final contour is smooth and not too noisy. As minimizing the total length of $\mathbf{C}$ is a way to minimize the jaggedness of the contour, we minimize the total Euclidean arc length of the contour $\mathbf{C}$. In other words, we could minimize

$$J_4(\mathbf{C}) = \int_{\mathbf{C}} \mathrm{d}s.$$

In terms of the level set function $\phi(x, y)$, this would be

$$J_4(\phi) = \int_\Omega |\mathcal{H}_\epsilon(\phi(x,y))| \mathrm{d}x\mathrm{d}y = \int_\Omega \delta_\epsilon(\phi(x,y))|\nabla\phi(x,y))| \mathrm{d}x\mathrm{d}y,$$

where $\delta_\epsilon(\phi)$ is the regularized delta function that selectively masks out only the pixels in the contour $\mathbf{C}$.

The minimum of the functional is found at a zero of its first variation $\delta J(C) = 0$, from which a PDE is extracted of the form $F(C) = 0$, termed the Euler-Lagrange equation. This equation is usually solved by introducing an artificial parameter t into $C(x, y, t)$, and solving

$$\frac{\partial C}{\partial t} = F(C). \tag{3.5}$$

In steady state, $\partial C/\partial t = 0$, leading to the solution of the Euler-Lagrange equation by introducing an explicit parameter, t, as described in (3.3).

For STACS, a specific form of the equation is obtained with $F_4 = F_c, F_3 = F_s$ are the internal, curvature and shape forces, and $F_1 = F_r, F_2 = F_g$ are the external, region-based and edge-based forces, and $\lambda_{1,2,3,4} = \lambda_{r,g,s,c}$, respectively.

3.5 Other Methods

There have been several different approaches to segmentation depending on different perspectives taken to the problem. For example, one could think of segmentation as the problem of dividing a given image into a (predefined) number of classes. This opens up a host of methods such as neural networks, traditionally designed for classification (or pattern recognition) to be used for segmentation. In this section, we discuss a few of the many different methods used for segmentation.

3.5.1 Graph-Theoretic and Related Methods

Recently, particularly in the vision and machine learning communities, there has been a significant amount of research on graph-theoretic approaches such as Bayesian networks, normalized cuts, spectral graph partitioning and graphical models for segmentation [73–76].

Bayesian Networks. This method consists of assigning a class label from a predefined set of labels to each pixel in the image and subsequently combining the decisions in a local neighborhood of a pixel to form meaningful segments of the image. The decision of region memberships is based on some prior knowledge acquired from a set of training images. As a training set is necessary for the decision phase, Bayesian networks are often categorized as a supervised segmentation method. Bayesian networks are also called belief networks because of the use of some belief (prior knowledge) in making a decision. Belief networks can use any other method, most notably Markov-random-field, maximum likelihood or tree-structured-based methods, to build the prior segmentation models for the decision phase. The decision phase may also be referred to as belief propagation.

In some applications where sufficient and representative training images are available, neural networks have been used for segmentation.

Normalized cut. This method casts the problem of segmentation as that of graph partitioning. Unlike belief propagation, which focuses on local features and their degree of similarity, normalized cut is a global criterion for segmenting an image. The normalized cut criterion measures a global dissimilarity between groups and a global similarity within groups to partition the image [74]. Unlike a min-cut (or max-cut) graph partitioning that might result in optimal cuts of isolated pixels and large portions of the image, the normalized cut criterion can be used to penalize such an unbalanced cut to achieve a more meaningful segmentation.

Spectral-graph partitioning. A spectral graph is a graph representation, $G = \langle V, E \rangle$, of an image, where V is a set of nodes, which could be pixels (or a group of pixels) or features (such as edge features) extracted from an image and E is a set of edges between these nodes that represents the relationship, such as the proximity of edges, between members of V. The problem of segmentation is thus recast as one of partitioning the spectral graph so that a partition consists of members of V with the strength of the relationship defined by E being strongest between members of the same partition and weakest between members of different partitions. The design goal is to ultimately achieve a segmentation where each

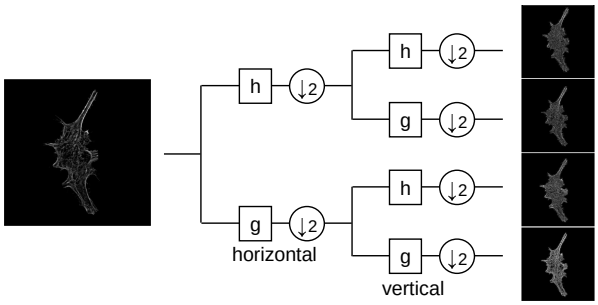


Figure 3.1: Two-channel analysis filter bank for 2D signals. The filter h is a highpass filter and g is a lowpass filter.

partition corresponds to a meaningful segment of the image. Different methods, such as edge detection, may be used to transform an image to its spectral-graph representation and different methods, such as normalized cuts, may be used to partition. This technique has been combined with machine learning approaches such as the Hebbian rule to facilitate the partitioning process.

3.5.2 Multiresolution and Multiscale Methods

Multiresolution (MR) transformations are known to offer localization, adaptivity and speed. For example, our team showed that using MR for classification of fluorescence microscope data significantly improves the classification accuracy, demonstrating the presence of hidden information in multiresolution subspaces [4]. The theory of MR is well-established and widely applied to many image processing problems with great success [77–79].

The fundamental unit for the most common form of this class of transformations—the discrete wavelet transform—is a two-channel analysis filter bank. Each channel of the two-channel filter bank consists of a filtering operation (either lowpass or highpass depending on the channel) followed by a downsampling by two. The lowpass filter g and its even translates typically form an orthonormal set. Further, for an orthonormal transform, g is orthogonal to the orthonormal set formed by the highpass filter h and its even translates [79]. When the analysis filter bank is applied to an image, it operates first on the rows of the image then on the columns. This results in four complementary labeled images (see Figure 3.1).

There are various benefits to using MR in segmentation. For instance, we can use MR to obtain a very fast initial segmentation on a very coarse approximation of the image and successively refine the result to the desired extent [80]. MR can also be used to characterize edges in image and to detect those edges that are meaningful to its segmentation [81]. Redundant versions of MR have proved to be more useful than their nonredundant versions for classification applications and have also been applied to segmentation tasks. However, nonredundant MR adds significantly to the computational load.

MR, rather than being used independently for segmentation, can be combined with most other segmentation methods to form a powerful combination that improves either the efficacy and/or efficiency of the segmentation method. For example, anisotropic diffusion and wavelet-based edge-detection have been shown to be superior to a fixed resolution/scale edge detection [81, 82]. Moreover, at any given resolution, we may apply the segmentation method of our choice at multiple scales to obtain information not provided by applying the same method at a fixed scale. Some methods using multiscale (MS) approaches have demonstrated the advantages of applying a method at multiple scales over their fixed scale counterparts [83,84]. MS and MR methods have both been shown to be highly advantageous to segmentation and have been combined with most of the methods described in the sections above [85–87]. In particular, we will be combining MR and MS with active contours and the active-mask framework that we propose in this work. Ideas along these lines have appeared in [84, 88–96].

3.6 Segmentation Methods for Fluorescence Microscopy

While the literature abounds in segmentation methods for digital images in various applications, the nature of fluorescence microscope images is very different from digital images seen in applications such as multimedia. For instance, fluorescence microscope images may lack edges (see Fig. 2.1(c)-(d)) that are central to defining an object of interest in a traditional application. Thus, many of the algorithms developed by image processing, computer vision and machine learning communities for generic applications cannot be directly used to analyze biological images. Moreover, fluorescence microscope images may have "features"

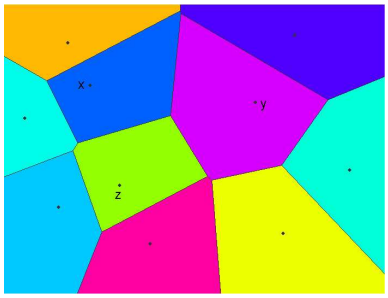


Figure 3.2: The Voronoi tessellation of a plane showing Voronoi polygons for points x, y and z.

that are topical to the application and, when tapped, aid in their segmentation. Thus, it becomes necessary to design segmentation methods tailored for this class of images.

3.6.1 Voronoi-Based Methods

One of the most widely used region-based cell segmentation methods in fluorescence microscopy is Voronoi based [97, 98].

Voronoi diagrams are based on some measure $\delta(\cdot)$ between two points in a plane. Let x and y be two points in the Euclidean plane and let $\delta(x, y)$ be the Euclidean distance between them. The bisector of the line joining x and y forms the locus of points equidistant from x and y and divides the plane into two halves, one of them containing points closer to x than y and the other containing points closer to y than x. Suppose we add a third point z to the plane and similarly construct the polygon joining their vertices. The intersection of the bisectors of these three bisectors is a point equidistant to x, y and z. The intersection of the half planes formed by the bisector of the lines joining x to y and z respectively, forms the Voronoi polygon for x (likewise for the other points, y and z). The generalized Voronoi diagram for a set of (three or more) initial points in a chosen space is a set of complete polygons under a chosen δ for all (three or more) initial points in a chosen space. The collection of Voronoi diagrams together with incomplete polygons in the convex hull of the initial points forms the Voronoi tessellation (see Fig. 3.2) of the plane [99].

Voronoi diagrams applied to cell segmentation. The Voronoi-based segmentation is the process of obtaining the Voronoi tessellation of the image and depends on an initial

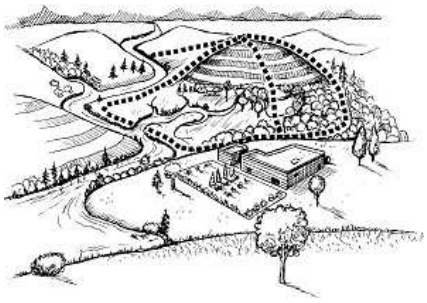


Figure 3.3: A sketch showing watersheds in a landscape [12].

set of points. In the case of cell images, these initial points are either provided manually, or automatically and based on, for example, a parallel channel depicting the nuclei of the cells. A small annulus around the initial points may be used instead of just the points themselves. With this initialization, Voronoi diagrams are drawn based on $\delta(\cdot)$, which could be pixel values, texture features or any other measure derived from the image. A polygon in the tessellation corresponds to a region. Voronoi-based segmentation is particularly useful in coarsely determining the locations of cells and in studying the spatial relationships between cells.

Disadvantages of Voronoi-based algorithms. Vornoi-based algorithms are usually not robust to noise. More importantly, they divide the entire image into polygonal regions. Thus, they do not produce tight contours around the cells and therefore, are not suitable for applications that depend on an accurate segmentation of the cells.

3.6.2 Seeded Watershed—A Region-Based Method

An algorithm (and its many variants) considered the most accurate for the segmentation of fluorescence microscope cell images is a region-based segmentation algorithm known as the *Seeded Watershed (SW)*. Because this is the algorithm we compare ours against, we will outline the design considerations of this method.

In the watershed algorithm [100], the intensity of the image is interpreted as elevation in a landscape. The algorithm splits the image into regions similar to the drainage regions of this landscape, so that points are assigned to the same region if they drain to the same point

(see Fig. 3.3). To create the watersheds, a gradient magnitude image is built in which water will start to rise from minima representing areas of low gradient (such as areas inside the cell or in the background regions), and the watershed borders will be built at the maxima of the gradient magnitude. Thus, ideally, the borders will be at the edges of cells, assuming those edges are well defined. For fluorescence microscope images, the problem is that these edges are not well defined and so the contour (watershed border) keeps evolving following gradual changes in the gradient, resulting in segmented regions much larger than the true cell.

Yet another, more recent design for the watershed, is based on binning regions in the image based on their gray scale values or building watersheds based on a distance map of a thresholded (binarized) version of the image to be segmented [18]. This method results in a large number of spurious splits and requires a postprocessing method to merge some of these regions.

In the SW algorithm [101, 102], instead of letting water rise from every minimum in the image, water rises only from places marked as seeds (such as the cell's DNA image, if a parallel DNA channel is available). This design ensures we have as many regions as the number of initial seeds.

Problems with watershed-based methods. The problems with watershed and its variants are twofold. Firstly, many segmented regions are discarded due to true cells being segmented into more than one region (splits), and more than one cell being merged into the same region (merges, see Fig. 4.3(c)). Any postprocessing to deal with merges would typically be manual. Secondly, by design, the SW algorithm splits the image into regions and does not define a contour around the region of interest. The segmented regions are typically much larger than the true cell (same figure), leading to background noise outside the cell being included with the cell.

Variations on a theme. To solve the problem of separating single cells from a multicell image taken by a fluorescence microscope, various modifications of the watershed algorithm have been extensively used. A survey of these algorithms is found in [103]. For instance, a

shape-based watershed segmentation has been found useful for segmenting cell nuclei [104]. Since merging of contours is a common problem with the watershed algorithms, rules may be defined so that neighboring objects may be merged or not. An example is to use the mean value of pixels along the border to decide if the border must exist [105].

3.6.3 Other Methods

As we noted in Section 2.1.1, the task of segmenting the nucleus of a cell in a fluorescence microscope image is simple compared to that of segmenting the cell itself (see Fig. 2.1(b) and (d)). However, it is more challenging to segment nuclei in tissue images as the cells may be more tightly packed and the nuclei less distinct. In one of the studies, the authors have applied a partial differential equation based geometric modeling that uses the level set embedding to preprocess and postprocess the image to solve the problem of segmenting nuclei from tissue images in three dimensions [106]. To isolate the individual cells from a tissue image is even more challenging and an attempt has been made to solve this problem by applying the more recent techniques of gradient curvature flows that also use a level set embedding [107]. These methods, designed for a specific application, have been attended with encouraging success.

Automating the entire process of segmentation may not be feasible when a high segmentation accuracy is desired. Thus, in off-line processes (small-scale studies), it may be viable to allow some user intervention to ensure better segmentation. A recent study based on dynamic programming accepted two inputs from the user—a point on the cell boundary and another inside the cell—to initiate the algorithm. Thereafter the segmentation was automatic. The optimum border around each cell was defined as the border with an average intensity per unit length greater that any other possible border around that cell, and was calculated using the gray-weighted distance transform. They reported perfect segmentation using this approach [108]. The disadvantages of these and other such methods are that *(1)* they cannot be applied to high-throughput applications due to the need for human intervention and *(2)* they are heuristic-based. Selecting the right heuristic involves significant effort and if the heuristic is not well chosen the segmentation outcome is beset with faults.

Apart from fluorescence microscope images, there have been algorithms based on computer vision applied to other imaging modalities within bioimaging. An example is the use of principal component analysis for the analysis of cell shape in light micrographs [109] and active contour methods for DIC or phase contrast microscope images [110]. Despite the reported success of these methods, not much has been done to extend them to fluorescence microscope images. To a large extent, watershed continues to be held as the most accurate, and the Vornoi-based and SW algorithms that are widely used. Occasionally, new methods are developed for a particular application in fluorescence microscopy, but are not seen to be used by others. A major reason for this stunt in progress is attributed to the nonavailability of the new methods or their nonflexibility to be tuned for applications other than those for which they were originally designed.

There have also been some commercial efforts towards developing fluorescence microscope image analysis toolboxes that contain generic spot detection, thresholding-based or edge-based segmentation algorithms along with some preprocessing and postprocessing tools (for example, BioApplication from Cellomics, ImageJ and others [111]). However, most of the available generic tools fail to perform satisfactorily when applied to a specific problem and more notably, are not within easy reach of end users or developers. A major effort towards making some of the cell analysis algorithms available as software that can be used by biologists is the Cell Profiler [112], an open-source software that is freely available, modular and compatible with most imaging formats. It allows the user to analyze thousands of cells without tedious user interaction. Following this lead and inspired by the idea of reproducible research, we hope that more algorithms will be made available for use as they are published.

As active contour-based methods are very flexible, besides combining them with MR and MS, there have been efforts to combine concepts from Bayesian networks and graph partitioning as well [113–115]. In addition to their flexibility to incorporate different approaches within the same framework, the ability of active contours to be tuned to be highly accurate has made them popular in biomedical image analysis, particularly for segmentation

and tracking [116, 117]. While they have been among the state-of-the-art tools in the medical imaging community for nearly a decade now, active contours have only recently made inroads into segmentation of biological images with promising success [13, 106, 110, 118]. In the following section we will see how STACS can be adapted to segment fluorescence microscope cell images.

Chapter 4

STACS for Fluorescence Microscopy

In this section we discuss how STACS was adapted to fluorescence microscope cell image segmentation in the context of the application of studying subcellular location patterns. Sections 4.2-4.3 are previous work conducted by our lab and reported in [13]. As I have been involved in extending the algorithm extensively (Section 4.4 onwards) and as the new framework we present in this thesis has been inspired by the design ideas and results of adapting STACS to fluorescence microscopy as well as extending STACS to other modalities, we have devoted a chapter to each of these two topics and, in particular, discuss the previous work at length in this chapter.

4.1 Dataset

Three patterns (total DNA, specific protein and total protein) in HeLa cells expressing GFP-UCE were imaged using confocal immunofluorescence microscopy (see Fig. 2.1). Serial sections in the z-axis through entire cells were taken with a step size of 0.1628μm and a pixel size of 0.0977μm in the x and y dimensions (1024×1024 pixels per section). There are 82 images in this set from 8 3D volumes [1]. Only the total-DNA and total-protein channels were used for segmentation.

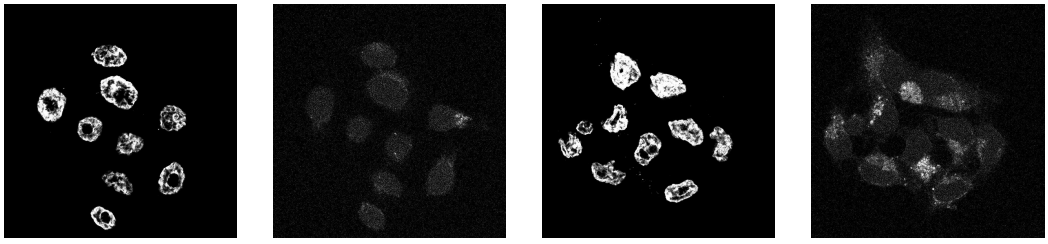
4.1.1 Reference for Evaluation—Hand-Segmented Images

For each image in the dataset, the DNA channel was used to guide a manual segmentation of the total-protein channel in each case. We use this hand segmentation (HS) as our reference for evaluating the performance of the algorithm. Because the total-protein images are noisy and devoid of edges, fluorescent pixels inundate the image with no clear separation between two cells in the foreground or between the foreground and background. Thus, HS is often imprecise, with the same image having different segmentations. A couple of extreme examples of HS contours from two different people are shown in Fig. 4.2.

The examples demonstrate the large variation between the HS examples and establish that the performance measure numbers are within a large margin of error (due to the large variation in the reference). Thus, although the algorithms' results are compared to HS contours, it might be that the algorithm actually performs correctly while disagreeing with HS.

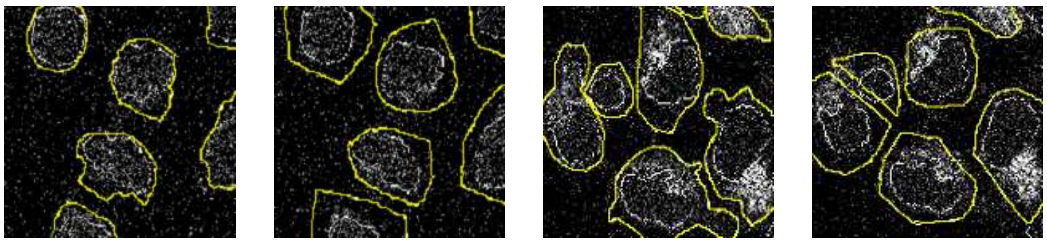
4.2 Modified STACS

The crux of the STACS algorithm is the choice of forces used for segmentation. As discussed in the previous chapter, the original STACS uses four forces: region-based, edge, shape prior and contour smoothness to segment cardiac MRI images. However, fluorescence microscope cell images do not exhibit any edges (see Fig. 4.1(b) and Fig. 4.1(d)), obviating the need for an edge-based force. Moreover, as each cell has a different shape, there is no benefit from



(a) DNA channel (b) Total protein (c) DNA channel (d) Total protein

Figure 4.1: Example multicell images for an easy case (first two images) and a difficult one (last two images). The images (a) and (c) depict the DNA channel and (b) and (d) depict the total protein channel. The algorithms are run on the total- protein image using the DNA channel as the starting contour for each case. (Images courtesy of Dr. R. F. Murphy [2].)



(a) Easy 1 (b) Easy 2 (c) Difficult 1 (d) Difficult 2

Figure 4.2: HS of on an easy case (first two images) and a difficult one (last two images) by two different people. These images have been cropped to highlight the differences. The interior contours in white represent the boundaries of total DNA of the cells obtained from an edge detection on the corresponding DNA channels whereas the contours in yellow represent HS of the total-protein channels. (Original images courtesy of Dr. R. F. Murphy [2].)

constraining the shape of the contour. Thus, the modified STACS used only two forces, an external, region-based force F_r and an internal force F_c which depended on the curvature of the contour. The evolution of the level-set function was given by

$$\frac{\partial \phi(x,y,t)}{\partial t} = \lambda_r F_r(x,y)|\nabla\phi(x,y,t)| + \lambda_c F_c(x,y,t)|\nabla\phi(x,y,t)|, \tag{4.1}$$

where λ_r and λ_c were scalar weights for the region-based force and the curvature force, respectively [13]. The authors used the information in the DNA channel to obtain the number of cells in the image, as well as an initial contour. The information in the total-protein channel was used to drive the segmentation algorithm.

4.2.1 Initialization of the Level-Set Function

Equation (4.1) assumes that an initial level set function is available. This was obtained by performing an edge-based segmentation on the DNA channel to develop initial contours. Subsequently, the level-set function was initialized using the Euclidean distance transform: Given a binary mask as input, for each pixel in the plane, the authors assigned the shortest Euclidean distance between the pixel and the nearest point on a contour as its value. Finally, they invert the sign of the distance for pixels outside of the contour. Thus, the value of any point (x, y) in the level-set function represented its distance from the closest point to it on the current contour and the sign of the value indicated whether (x, y) was inside or outside

the current contour.

4.2.2 Evolution of the Level-Set Function

The evolution of the level-set function is given by (4.1) and is driven by the following forces.

External force, F_r. The region-based force is based on the assumption that the pixels in the objects and the background are drawn from two different statistical models; $\hat{M}_{\text{in}}$ and $\hat{M}_{\text{out}}$. Thus, a pixel lying inside the contour C should be described by the model $\hat{M}_{\text{in}}$, while a pixel outside the contour should be described by $\hat{M}_{\text{out}}$ [67]. More specifically, the external region-based force relies on the idea that the density of proteins should be higher inside the cell than outside. As a simple statistical measure, the mean number of white pixels in the image (previously converted to a binary image) are used. For each pixel (x, y) on the contour, the mean number of white pixels is estimated in a small rectangular neighborhood W_d around the pixel given by the span of $(x - W_d, y)$ and $(x, y + W_d)$, inside and outside the contour as M_{in} and M_{out} respectively. The deviation between the model and these values let one define a force to drive the segmentation. Thus, the region based force at each pixel (x, y) on the current contour is given by,

$$F_r = M_{\text{in}} - \hat{M}_{\text{in}} + M_{\text{out}} - \hat{M}_{\text{out}}. \tag{4.2}$$

The strength of F_r in driving the evolution is determined by its weight, λ_r.

Internal force, F_c. The internal force is a smoothing force based on the curvature F_c of the contour. Given the level set function, this quantity can be computed at each iteration [70] as:

$$F_c \;=\; \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{|\nabla\phi|^3},$$

where $\phi_x, \phi_y, \phi_{xx}, \phi_{yy}, \phi_{xy}$ are the appropriate partial derivatives with respect to x and y and $|\nabla\phi| = \sqrt{\phi_x^2 + \phi_x^2}$. This force imposes that the final contour not be jagged but be smooth. The strength of the smoothness is determined by its weight, λ_c.

Annealing schedule. In (4.1), the coefficients λ_r and λ_c evolve in time: this is referred to as annealing, or, cooling schedule in [67]. (This is different from the "annealing" used in traditional optimization literature.) The region-based coefficient λ_r is kept relatively large initially and its influence is reduced with iteration number, whereas the curvature coefficient λ_c is kept constant, so its influence is greater towards the end.

4.2.3 Numerical Implementation

The evolution described in (4.1) is realized through the numerical implementation,

$$\phi_{i,j}^{\eta+1} = \phi_{i,j}^{\eta} + \Delta t \left[\lambda_r \left(\max(F_{r_{i,j}}^{\eta}, 0)\Delta^+ + \min(F_{r_{i,j}}^{\eta}, 0)\Delta^- \right) - \lambda_c F_{c_{i,j}}^{\eta} \sqrt{(D_{i,j}^{0x})^2 + (D_{i,j}^{0y})^2} \right], \tag{4.3}$$

where η denotes the discrete time index with time step Δt and subscripts i, j denote the discrete values of the corresponding continuous domain coordinates (x, y). $\phi_{i,j}^{\eta}$ is the grid value of the point (i, j) at ηth iteration and is an approximation to the continuous domain $\phi(x, y)$ at that iteration. Likewise, $F_{r_{i,j}}^{\eta}$ and $F_{c_{i,j}}^{\eta}$ are approximations to F_r and F_c in equations (4.2) and (4.2.2).

The spatial derivatives necessary of ϕ are computed using the forward, backward and central approximations of ϕ_x and ϕ_y respectively,

$$\begin{aligned} D_{i,j}^{+x} &= \phi_{i+1,j}^{\eta} - \phi_{i,j}^{\eta}, & D_{i,j}^{+y} &= \phi_{i,j+1}^{n} - \phi_{i,j}^{\eta}, \\ D_{i,j}^{-x} &= \phi_{i,j}^{\eta} - \phi_{i-1,j}^{\eta}, & D_{i,j}^{-y} &= \phi_{i,j}^{n} - \phi_{i,j-1}^{\eta}, \\ D_{i,j}^{0x} &= \frac{\phi_{i+1,j}^{\eta} - \phi_{i-1,j}^{\eta}}{2} \text{ and} & D_{i,j}^{0y} &= \frac{\phi_{i,j+1}^{\eta} - \phi_{i,j-1}^{\eta}}{2}. \end{aligned}$$

The symbols,

$$\begin{aligned} \Delta^+ &= \sqrt{\max(D_{i,j}^{-x}, 0)^2 + \min(D_{i,j}^{+x}, 0)^2 + \max(D_{i,j}^{-y}, 0)^2 + \min(D_{i,j}^{+y}, 0)^2}, \\ \Delta^- &= \sqrt{\max(D_{i,j}^{+x}, 0)^2 + \min(D_{i,j}^{-x}, 0)^2 + \max(D_{i,j}^{+y}, 0)^2 + \min(D_{i,j}^{-y}, 0)^2} \end{aligned}$$

represent the forward and backward approximations of $|\nabla\phi|$. Finally, the numerical approximation of the curvature term in (4.2.2) is implemented using central differences, $\phi_x = D_{i,j}^{0x}$,

$\phi_y = D^{0y}_{i,j}$ and

$$\begin{aligned}
\phi_{xx} &= \phi^{\eta}_{i-1,j} - 2\phi^{\eta}_{i,j} + \phi^{\eta}_{i+1,j}, \\
\phi_{yy} &= \phi^{\eta}_{i,j-1} - 2\phi^{\eta}_{i,j} + \phi^{\eta}_{i,j+1} \text{ and} \\
\phi_{xy} &= \frac{\phi^{\eta}_{i-1,j-1} - \phi^{\eta}_{i+1,j} - \phi^{\eta}_{i-1,j+1} + \phi^{\eta}_{i+1,j+1}}{4}.
\end{aligned}$$

At every iteration η, forces are computed only at the points (x, y) on the current contour $\mathcal{C}$. However, ϕ is defined over the entire image domain. To keep the ϕ consistent across iterations, it becomes necessary to update the rest of the values in ϕ in accordance with the values computed at the points on the current contour. This process of level-set update is called the *velocity extension* function. As ϕ is interpreted as a signed distance function, moving any point (x, y) on the contour by $(\Delta x, \Delta y)$ in some direction, it would move all the points connected to (x, y) by the same amount. The movement of points connected to the contour are most influenced by the movement of the point they are closest to. Hence, for every point $(\tilde{x}, \tilde{x})$ in ϕ, we compute its distance from every point (x, y) on the current contour. We pick that point (x, y) on the contour which is closest to $(\tilde{x}, \tilde{y})$ and update ϕ at $(\tilde{x}, \tilde{y})$ by the same value it is being updated by at (x, y). Trivially, for each point on the contour, it is closest to itself.

4.2.4 Results

The results of this algorithm are given in Fig. 4.3, third column, for an easier case (top) and a harder case (bottom) [13]. As these are intermediate results, no objective measures were computed. From the figures we see that MSTACS produces continuous, smooth contours that appear to match well the hand segmented contours (first column). However, when cells are close together or linked by an area of noise, merges occur (for both easy and difficult cases), that is, two individual contours at the initialization phase merge during the segmentation process. This is because, as explained in Section 3.3.2, level-set formulation of active contours handle changes in topology gracefully, that is, contours merge and split

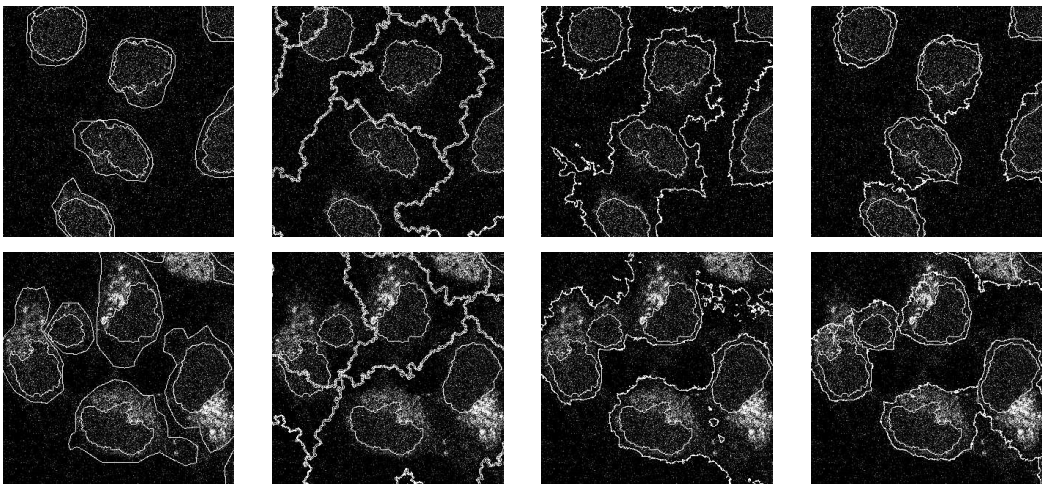


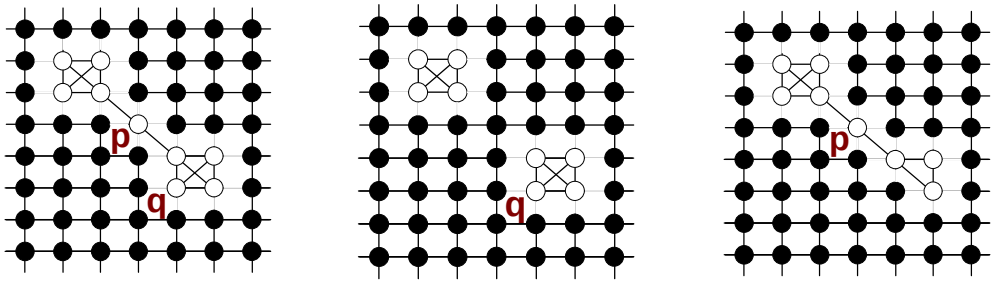
Figure 4.3: Segmentation results on total-protein images of HeLa cells [13]. Sample images for an easy case (top row) and a difficult one (bottom row). All images are of total protein with the initial contour inside the final segmented contour. Cropped regions are shown for detail. First column: Hand-segmented images used as ground truth. Second column: Results of the modified seeded watershed [3]. Note how in the difficult case (bottom image), there are both splits and merges. Third column: Results of the modified STACS algorithm. Note how in both cases, the contours merge, creating artificial cells. Fourth column: Results after 35 iterations of our algorithm with topology preservation added (TPSTACS). The merging of contours is no longer a problem. (Original images courtesy of Dr. R. F. Murphy [2].)

adapting to the image topology. Since we know the number of cells (given by the number of nuclei in the DNA channel), we want to impose the constant number of contours on our algorithm (termed "topology preservation").

4.3 Topology Preserving STACS

In the context of fluorescence microscope cell image segmentation, topology is used to mean the number of contours being represented by level-set function. Thus, topology preservation refers to conserving the number of contours present at the end of the initialization phase throughout the evolution. To do this, the authors followed the approach in [119], where the notion of topology preserving level-set method is introduced.

Under given assumptions, a change in topology can occur only at points where the sign of the level set function is changing. Moreover, at these points, a change in topology will occur if the point is so-called nonsimple [120]. Fig. 4.4 shows an example of a simple and a nonsimple point and how a change in the sign of the nonsimple point results in a change



(a) Discrete lattice (b) Topology change (c) No change

Figure 4.4: Simple and nonsimple points in digital topology: (a) A discrete lattice with 4-connected black components that are the background and an 8-connected white component that is the foreground. p and q are foreground points. (b) Deletion of p causes a change in the number of white components and hence topology. Thus p is a nonsimple point. (c) Deletion of q, a nonisolated border point, does not change the topology and so q is a simple point.

in the topology. The simple-point criterion is based on the idea of digital topology and topological number [119]. The approach is to monitor the change of sign of the level-set function and apply the simple-point criterion.

From Fig. 4.4(c), it is clear that for any point (i, j) in a discrete lattice, one only needs to test in a 3×3 neighborhood around (i, j) to determine if the point is simple or nonsimple. Considering a discrete lattice with (4,8) connectivity (that is, 4-connected background and 8-connected foreground), the simple-point criterion states that a point is simple if its topological numbers, T^4 and T^8, are both equal to one. Otherwise it is nonsimple.

The topological numbers T^4 and T^8 give the number of connected components in the 3×3 neighborhood of a point on the current contour in its background and foreground mask, respectively. The background and foreground masks are extracted from ϕ as the characteristic functions of all points less than zero and greater than or equal to zero, respectively. Then, to compute T^4 for a point (i, j), the 3×3 binary neighborhood around (i, j) is considered from the background mask. If, after deleting (i, j) (assuming it was present) from this 3×3 block, the number of 4-connected components is 1, then $\mathrm{T}^4 = 1$. Likewise, to compute T^8 for a point (i, j), the 3×3 binary neighborhood around (i, j) is considered from the foreground mask. If, after deleting (i, j) (assuming it was present) from

this 3×3 block, the number of 8-connected components is 1, then $\mathrm{T}^8 = 1$.

If point (i, j) is not simple, the level-set function is not allowed to change its sign at that point. Note that this test is performed at the end of each iteration on all points that have changed their sign from the previous iteration of our Modified STACS algorithm. This test does not require much overhead. The pseudocode for TPSTACS is presented in Algorithm 1.

Algorithm 1: **[TPSTACS]** Input: I_d, a DNA image, I_p, a total protein image, dt, time step, scalar η_{t}, number of iterations, scalar λ_c, weight of the curvature force F_c, S_r, scheme of evolution for the coefficient of the region-based force F_r. Output: ϕ, the final level set function.

TPSTACS(I_d, I_p, dt, η_{t}, λ_c, S_r)
  initialize level set function $\phi(0)$ based on edge detection in I_d and Euclidean distance transform
  compute all the values of λ_r based on S_r
  convert I_p to a binary image by thresholding
  for $\eta = 1$ to η_{t} **do**
    for all points on the contour **do**
      compute F_r, F_{curv}, RHS of (4.1)
    end for
    extend the values to the whole domain, get F_ϕ
    $\phi_{\mathrm{tmp}} = \phi(\eta) + dtF_\phi$
    for all points where the sign of $\phi(\eta)$ is changing **do**
      if nonsimple point **then**
        prevent change of sign of ϕ: $\phi_{\mathrm{tmp}} = \epsilon \cdot \mathrm{sign}(\phi(\eta))$
      end if
    end for
    $\phi(\eta + 1) = \phi_{\mathrm{tmp}}$
  end for
  return ϕ

4.3.1 Measures of Performance

To assess the performance of the algorithm, the authors of the previous work used these common methods: area overlap, area similarity and recall/precision. They compared SW and TPSTACS to the hand segmented images (HS), as well as DNA images. We delve deeper into these measures for a deeper insight to the results and also because we use some of these measures to quantify the performance of the algorithms we present this in this

thesis.

Area Overlap. This detects how much of each cell overlaps with HS (or DNA). HS_i, $i = 1, \ldots, C$, is an array with values 1 where the hand segmented masks are nonzero and 0 elsewhere. Similarly, SW_i $(i = 1, \ldots, C_s)$, and TP_i $(i = 1, \ldots, C_t)$, denote masks output by either SW or TPSTACS. Note that there are C true cells, and SW and TPSTACS output C_s, C_t contours (masks), which could be different from C. We also denote by $n(A)$ the number of nonzero values in the mask A. Then, area overlap is defined as

$$AO_{i,j} \;=\; \frac{n(HS_i \wedge SW_j)}{n(HS_i)},$$

where $\wedge$ denotes the logical "and" operator (pixel-wise). The above equation is valid for SW, and for TPSTACS we substitute TP for SW.

Algorithm 2 gives the procedure used to compute the area overlap and is later useful for recall/precision. For each true cell mask (HS or DNA) in each image, we compute the area overlap with all SW as well as TP masks. This gives rise to matrices A_{SW} and A_{TP} (again for HS and DNA). These matrices are used later to compute recall and precision with respect to a series of thresholds. Then, for each SW, TP mask, the largest overlap per true cell mask (per row) is found, and this is added to the overall area overlap AO_{SW} and AO_{TP}. These are then normalized by the number of true cells in each image and the total number of images.

Note that this measure will count the splits although the resulting contour might be discarded. For example, if a cell is split in half between two contours, the AO will count 50% once though the segmentation result is not usable. Similarly, if two cells are merged into one contour, one of them will be counted as 100% once, though again, the segmentation result is not usable. This measure will be lenient towards algorithms producing loose contours, such as SW.

Area Similarity. Another commonly used measure of performance is *Area Similarity (AS)* [121], which, for each hand segmented mask, compares the area of that mask with the area of any SW, TP mask which overlaps with it and normalizes it by the total area of

Algorithm 2: [**Area Overlap**] Input: Set of true cell masks HS_i, $i = 1, \ldots, C$, as well as SW and TPSTACS masks, SW_i, $i = 1, \ldots, S$, and TP_i, $i = 1, \ldots, P$. Output: Two arrays. A_{SW} of size $S \times C$ and A_{TP} of size $P \times C$, containing area overlaps for all cell masks and all output masks for both algorithms. Also, two area overlaps for SW and TP are output: AO_{SW} and AO_{TP}.

AreaOverlap(HS, SW, TP)
  for all images, $n = 1$ to N **do**
    for all true cell masks, $i = 1$ to C **do**
      for all SW masks, $j = 1$ to S, TP masks, $k = 1$ to P **do**
        compute $AO_{i,j}$ and store in $(A_{SW})_{n,i,j}$
        compute $AO_{i,k}$ and store in $(A_{TP})_{n,i,k}$
        keep track of S and P
      end for
    end for
    Initialize $AO_{SW} = 0$, $AO_{TP} = 0$
    for all SW masks, $j = 1$ to S, TP masks, $k = 1$ to P **do**
      find the largest overlap, $\max_i (A_{SW})_{n,i,j}$, add to AO_{SW}
      find the largest overlap, $\max_i (A_{TP})_{n,i,k}$, add to AO_{TP}
      keep track of S and P
    end for
    normalize AO_{SW} and AO_{TP} by the number of true cells C
  end for
  normalize all measures across all images
  return $A_{SW}, A_{TP}, AO_{SW}, AO_{TP}$

both masks (unlike area overlap). Thus

$$AS_{i,j} = \frac{2n(HS_i \wedge SW_j)}{n(HS_i) + n(SW_j)},$$

for seeded watershed, and similarly for TPSTACS. As for area overlap, the number is normalized by the number of true cells present, as well as the total number of images. This measure penalizes an algorithm if its contour is not tight, even though it might contain the entire hand segmented contour. For this application, tightness of the contour is a desirable property as a nontight contour will introduce a significant amount of background noise existing outside the cell. According to [121], $AS \geq 70\%$ indicates excellent agreement of the segmented region with HS.

Recall and Precision. The *recall/precision* method derives two quantities, as per-

centages. Consider the output of the algorithm to be a set of contours (masks). A true positive, $T^{(+)}$, is a mask designated by the algorithm as a cell and that is one, whereas a false positive, $F^{(+)}$, is a mask designated as a cell but which is not. Likewise, a false negative, $F^{(-)}$ is defined. Unlike standard definitions of recall and precision, we do not have true negatives, $T^{(-)}$. The two quotients, *Recall* (R) and *Precision* (P) can then be computed:

$$R = \frac{T^{(+)}}{T^{(+)} + F^{(-)}}, \qquad P = \frac{T^{(+)}}{T^{(+)} + F^{(+)}}.$$

Note that $T^{(+)} + F^{(-)} = C$, that is, the number of true positives and false negatives is equal to the number of true cells. Similarly, $T^{(+)} + F^{(+)} = C_s$, for SW and $T^{(+)} + F^{(+)} = C_t$, for TPSTACS, that is, the number of true positives and false positives is equal to the total number of contours produced by the algorithms. Since there is no true negative, it is possible to simultaneously obtain high R as well as P (see Fig. 4.5).

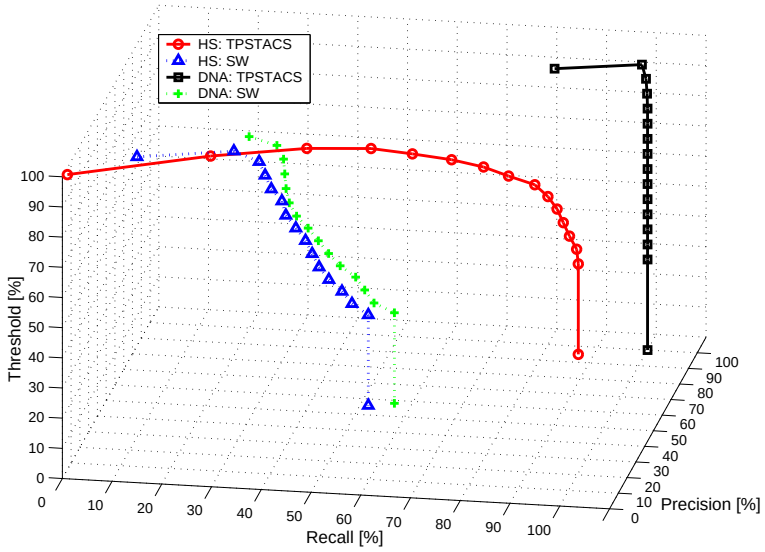


Figure 4.5: Recall and precision for SW and TPSTACS, computed against the hand segmented images (HS) as well as the DNA ones (DNA) with values of threshold from 100% decreasing by 5% to 35% [13]. Note that, for orientation, a final, artificial point at T=0% has been added as a projection of the last point on the curve. The curve for TPSTACS against DNA reduces essentially to one point as all the DNA contours are enclosed within the final TPSTACS contours.

Algorithm 3: [**Recall/Precision**] Input: Two arrays, A_{SW} of size $C_s \times C$ and A_{TP} of size $C_t \times C$, containing area overlaps for all cell masks and all output masks for both algorithms. Output: Four arrays, R_{SW}, P_{SW}, R_{TP}, P_{TP}, with recall/precision values for both SW and TP, for various values of threshold for area overlap.

RecallPrecision(A_{SW}, A_{TP})
  initialize lower threshold $T_l = 0.1$
  for all images **do**
    for threshold $T = 1$ to 0.55 **do**
      initialize $A_{SWtmp} = A_{SW}$, $A_{TPtmp} = A_{TP}$
      for all values v of A_{SWtmp}, A_{TPtmp} **do**
        if $v < T_l$ **then**
          zero out v in the matrix
        else if $T_l \leq v < T$ **then**
          $v = -1$
        end if
      end for
      for SW and TPSTACS **do**
        compute $T^{(+)}$ as the number of rows with one strictly positive entry only
        compute $F^{(-)} = C - T^{(+)}$
        compute $F^{(+)} = C_s - T^{(+)}$ {for SW}
        compute $F^{(+)} = C_t - T^{(+)}$ {for TPSTACS}
        compute R, P and store in R_{SW}, P_{SW}, R_{TP}, P_{TP}
      end for
    end for
  end for
  normalize all measures across all images
  return R_{SW}, P_{SW}, R_{TP}, P_{TP}

4.3.2 Results

The authors of [13] tested TPSTACS on the above set of images with the cooling parameters set to $\lambda_r = 50$ at the start and $\lambda_r = 10$ at the end of the run, and $\lambda_c = 1$. This means that the region-based term dominates in the beginning, growing the contours to roughly divide the pixels based on their statistical models, while the curvature term remains constant and grows in importance towards the end, smoothing the contour.

Results based on these methods are given in Table 4.1 as well as Fig. 4.3. Both algorithms were tested against HS as well as DNA. All measures have been averaged over all cells and all images. In terms of recall and precision, which roughly measure the percentage of usable contours after segmentation, TPSTACS outperforms SW by a fair margin. This is because

		SW [%]	TPSTACS [%]
Area Similarity (AS)		30.82	80.51
Area Overlap (AO)	HS	62.15	82.14
	DNA	62.29	99.80
Recall (R)	HS (T=70%)	37.88	71.13
	DNA (T=95%)	36.75	99.06
Precision (P)	HS (T=70%)	39.99	76.82
	DNA (T=95%)	36.28	99.06

Table 4.1: Segmentation results for the seeded watershed algorithm (SW) and our topology-preserving STACS (TPSTACS) [13]. Both algorithms were tested against both hand segmented images (HS) as well as the DNA ones (DNA). Note that R and P are given for the value of threshold $T = 70\%$ for HS and $T = 95\%$ for DNA. Other values of R and P are given in Fig. 4.5.

there is extensive splitting and merging in SW, whereas there are few in TPSTACS due to our topology preservation constraint.

Note that the measures of recall and precision are somewhat coarse as they do not take into account the fit of the final contour to the hand segmented one. Thus, area similarity is used to compare the areas of hand segmented cells versus those segmented both by SW as well as TPSTACS. The measure yielded 80.51% for TPSTACS versus 30.82% for SW; this was expected as TPSTACS produces tight contours as opposed to oversegmented SW ones (see Fig. 4.3, second and fourth columns). Area overlap yielded 82.14% for TPSTACS versus 62.15% for SW (against HS). Looking at the area overlap against the DNA (as well as recall and precision against the DNA), we see that essentially, almost 100% of our segmented cells are usable. Therefore, both objective (recall/precision, area similarity and area overlap) and subjective (visual inspection) measures of quality establish that TPSTACS outperforms SW by a fair margin.

4.4 Discussion—Transitioning from the Past to the Present

In summary, by using the combination of STACS with topology preservation, the authors have built a powerful algorithm for segmentation of fluorescence microscopy images [13]. We note that the quantitative results obtained for TPSTACS establishes the accuracy of the initialization procedure. Further, although the parallel DNA channel was used to initialize the seeds even for SW, as they were not used in exactly the same way, the results for the various measures are different. That is, the poor performance of SW due to splits and

merges is due in large measure to faulty initialization. Even with perfect initialization SW is not designed to produce tight contours around the cells. TPSTACS has the upper hand in this respect. Up to this point, we have presented the previous work on top which we began to build to ultimately lead to the new framework.

In the following sections we discuss improvements to TPSTACS to enhance the accuracy of segmentation and/or decrease the runtime of the algorithm. As AO does not give us an accurate measure of how well automated segmentation matches with HS, we no longer report this measure for the modifications proposed to TPSTACS. Moreover, as the initialization procedure using the parallel DNA channel is nearly 100% accurate, we cease to measure the recall and precision measures, which give us a coarse measure of the accuracy of cell detection. Rather, we report only AS which is a stringent measure that penalizes an algorithm that does not produce tight contours and describes how similar the final result is to the desired result—reference algorithm/HS. We also measure runtime and any other quantity that is relevant to the aspect of the algorithm that we endeavor to improve.

4.5 Local Area Topology Preserving STACS

While TPSTACS is found to be an effective segmentation method, its large run time (approximately four hours for an image of size 1024×1024) does not allow it to be used for high-throughput applications. A profile of the code reveals that the bottleneck is the velocity extension function, which accounts for nearly 80% of the computational load. This extension function, as explained above, involves updating the entire level-set function with the force values computed at only the points on the contour. For this, we compute the distance of every point in the level-set function to every point on the current contour (the zero level set). This step of extending the forces to update the level-set function is a time-consuming and computation-intensive process. We note that the approach described here is indeed the brute force approach to the problem. However, regardless of the implementation, the process of updating the level set function is an involved one.

As the problem of updating the level-set function is inherent to the often used signed

distance interpretation of the level-set function, there have been several modifications suggested to expedite the update (for example, see [122–124]). Among the many methods, the narrow band algorithm is particularly useful in applications that need topology preservation [125]. However, our experiments revealed that because of the irregular shape of the cells, it took more computation to construct a narrow band around the cells. Further, the high density of cells in a given image, caused the narrow bands to overlap very early in the segmentation process. This required an overhead of resolving the conflicts caused by overlaps and further contributed to the runtime.

Instead of using a narrow band around the current contour, we decided instead to use a small rectangular neighborhood W_l around the current contour. This region would be quick to construct and any overlaps would be resolved by merging the neighborhoods—that is, considering a single neighborhood around the two (or more contours) whose individual local areas overlapped. We called this the *local area TPSTACS algorithm* (LATPSTACS). The steps of the algorithm are presented Algorithm 4.

We note that if a number of iterations is large and W_l small, then we might see abruptness in the contour. To overcome the incoherence in the level-set function, we will have to reinitialize the level-set function. This reinitialization, after sufficiently many iterations, is inherent to optimization schemes that restrict the region of level-set update (including the narrow-band algorithm).

4.5.1 Results

For ten representative images of size 512×512 pixels, we ran LATPSTACS for $\eta_{\mathrm{t}} = 10$ iterations on each image with $W_l = 20$. This choice ensured LATPSTACS would not quickly degenerate to the original TPSTACS problem as local areas, being small, would take a number of iterations before overlapping. We averaged the time taken to compute the velocity extension function across these 100 iterations and report the result in Table 4.2.

While we saved time during each iteration, it turned out that $W_l = 20$ was too small a number, as the contour often hit the bounding-box of the local area and was constrained its expansion. This necessitated running more iterations of the algorithm to achieve the same

Algorithm 4: **[LATPSTACS]** Input: f_d, a DNA image, f_p, a total protein image, dt, time step, scalar η_{t}, total number of iterations, scalar λ_c, weight of the curvature force F_c, S_r, scheme of evolution for the coefficient of the region-based force F_r. W_l, side of the rectangular local area. Output: ϕ, the final level set function.

LATPSTACS(f_d, f_p, dt, η_{t}, λ_c, S_r, W_l)
  initialize level set function $\phi(0)$ based on edge detection in f_d and Euclidean distance transform
  compute all the values of λ_r based on S_r
  convert I_p to a binary image by thresholding
  for $\eta = 1$ to η_{t} **do**
    for all points on the contour **do**
      compute F_r, F_{curv}, RHS of (4.1)
    end for
    for all contours **do**
      find the coordinates $i_{\max}$, $j_{\max}$, $i_{\min}$ and $j_{\min}$
      compute the local area given by the span, $i_{\min} - W_l/2$ and $i_{\max} + W_l/2$, horizontally and $j_{\min} - W_l/2$ and $j_{\max} + W_l/2$, vertically
      extend the force values to the local areas, get $F_{\hat{\phi}}$
    end for
    $\phi_{\text{tmp}} = \phi(\eta) + dtF_{\hat{\phi}}$
    for all points where the sign of $\phi(\eta)$ is changing **do**
      if nonsimple point **then**
        prevent change of sign of ϕ: $\phi_{\text{tmp}} = \epsilon \cdot \text{sign}(\phi(\eta))$
      end if
    end for
    $\phi(\eta + 1) = \phi_{\text{tmp}}$
  end for
  return ϕ

results as that of TPSTACS.

Increasing W_l, we found that for images of size 1024×1024 pixels, updating the level-set function at each iteration only in the neighborhood of $W_l = 200$ pixels around each contour, we were able to reduce the run time of the algorithm by 18%. The contours obtained by the method were almost the same as those returned by the TPSTACS because the forces driving the evolution are computed in exactly the same way. The area similarity measure between TPSTACS and LATPSTACS was computed to be roughly 98%. However, this too did not prove to be a useful solution as the high density of cells and increased dimensions of the local area ensured that the problem quickly degenerated to the original. Moreover,

	Velocity extension [sec]	Local-area computation [sec]	Entire iteration[sec]
TPSTACS	39.10	0	40.10
LATPTACS	21.90	9.50	32.60
Difference	−17.2	+9.50	**−7.5**

Table 4.2: Average time taken during one iteration of $TPSTACS$ and $LATPSTACS$ for computing the velocity extension function, local-area computation and the entire iteration on 10 images of size 512×512 pixels, running 10 iterations on each image with the local area $W_l = 20$ pixels.

the gain in computational speed was marginal, given the large margin for improvement.

4.6 Multiresolution STACS

We wanted to further improve the run time of the algorithm without compromising on the segmentation quality. MR techniques, as described in Section 3.5.2, have been successfully used to detect edges and aid in the contour evolution for improving the segmentation quality or speed [81, 89, 92]. While we are not interested in the edges or the transform domain coefficients themselves, we recognized that the coarse version of the image resulting from such a transformation will aid its segmentation [80].

As a proof of concept, we used a 2D analysis filter bank (see Fig. 3.1) with the simplest MR filters, the Haar basis, and performed a 3-level decomposition. We expected (a) the downsampling to greatly reduce the run time and (b) the smoothing to render the cells easier to discern. In fact, while we may still use the parallel DNA channel that is available to initialize the level-set function, due to the lowpass filtering the cells are discernable enough to let us apply a blob-detection algorithm directly to them. Fig. 4.7 shows an example of a 2D HeLa cell image segmented without using any parallel channel in the initialization phase.

The procedure we follow for using multiresolution STACS (MRSTACS) for segmentation is as follows: (*1*) We decompose the given image f to K levels using a two-channel filter bank. The choice (length and shape) of the filter(s) used in the filter bank may be determined based on the shape of the cell and type of marker used (although, experiments have revealed this does not make a huge difference to the final outcome). We consider only the coarsest version of the transformed image. (*2*) We initialize the level-set function at

resolution $k = K$, $\phi^{(k)}$, where $\eta = 0$ as described above using either a parallel channel or by performing blob detection on the transformed image. As the size of the image is smaller by 2^k in each dimension, just a few iterations of the level-set function is sufficient to achieve a coarse segmentation at resolution k.

While we have the image at resolution $k-1$, we need to construct the corresponding $\phi^{(k-1)}$. As we do not have the level-set function at the original resolution or even the detailed versions of the level-set function at resolution k, we would have to construct $\phi^{(k-1)}$ using only $\phi^{(k)}$. One of the ways to perform this *construction* is to interpolate $\phi^{(k)}$. There is a large pool of functions such as the family of β-splines that we could select from. However, if we want to continue to preserve topology, then the question which begs an answer at this juncture is, "what does a nonsimple point in $\phi^{(k)}$ translate to in $\phi^{(k-1)}$?" An answer to this is not immediately clear. As our level-set function can be interpreted as a signed distance transform, we need to preserve Euclidean distances. For this, we may use the β^0-spline or the nearest neighbor interpolation function to ensure the sign of the nonsimple point remains unchanged in the interpolated version (see Fig. 4.6 for the action of two different interpolation functions). That is,

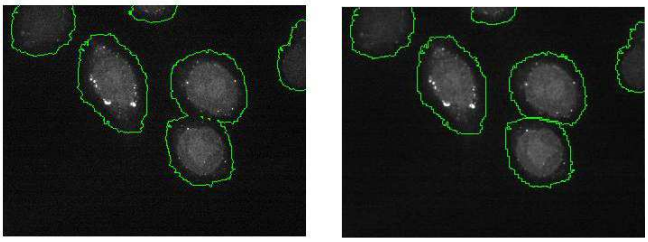
$$\phi^{k-1}(m,n) = \sum_i \sum_j \phi^k(2i,2j)\beta^0[m/2-2i, n/2-2j], \tag{4.4}$$

where

$$\beta^0(m,n) = \begin{cases} 1 & \text{if } |m|,|n| < 1/2, \\ 1/2 & \text{if } |m|,|n| = 1/2, \\ 0 & \text{otherwise.} \end{cases}.$$

The only drawback of the nearest neighbor interpolation is that it is relatively not smooth. To smooth out some of the jaggedness in the contour, we could run a few more iterations of the algorithm (with the contour smoothness force as the dominating term).

Thus, starting from the final segmentation at an initial coarse resolution $k = K$, we successfully lift the result to a higher resolution $k-1$ and refine it, all the way till the desired maximum resolution $k = K_0$. The number of iterations required at each higher



(a) (b)

Figure 4.6: Images showing the effect of two different interpolation functions used to lift ϕ to a higher level. (a) Bicubic interpolation causes merges. (b) Nearest-neighbor interpolation preserves the topology of the image. (Original images courtesy of Dr. R. F. Murphy [2].)

resolution $k-1$ using the final segmentation at resolution k as the starting point, is fewer than if the initialization and segmentation had been performed at just that resolution $k-1$. Thus, it takes fewer iterations and, consequently, much less time to refine the segmentation outcome on the original image than it would if we performed the segmentation directly on the original image. The pseudocode for MRSTACS is provided in Algorithm 5.

4.6.1 Results

We decompose an image of size 1024×1024 to $K = 3$ levels and use the same forces and values of the parameters λ_r and λ_c as TPSTACS and recompute the model statistics only. Due to the subsampling as well as a good initialization due to the smoothing, both provided by MR, the number of iterations required is much less ($2-3$ instead of roughly 45). With this method (see an example in Fig. 4.7), we compute an initial coarse segmentation in less than 10 seconds and the final contours in less than 30 minutes (on an Intel Pentium M 1.6 GHz platform), between 1-2 orders of magnitude faster than without MR.

Discussion. While time saving is the biggest advantage of MRSTACS and smoothing also helps to discern the cell more easily, there are some limitations to MRSTACS. Firstly, as we increase the number of decomposition levels, the sensitivity to the model parameters and threshold used for the segmentation increases. This is because, even at $k = 3$, that is one-eighth the resolution of the original image, a single pixel corresponds to $2^3 \times 2^3 = 64$ pixels. This implies that if we are off by a single pixel, the error is magnified by a factor of roughly 64. Refining the segmentation outcome at higher resolutions mitigates this problem

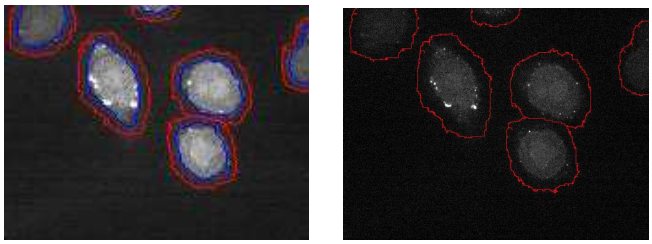
Algorithm 5: [**MRSTACS**] Input: f_d, a DNA image, f_p, a total protein image, K, the maximum level of decomposition, K_0, the maximum level of refinement, dt, time step, %This may be varied at each resolution scalar $\{\eta_{\mathrm{t}}^k\}$, number of iterations at resolution k, scalar λ_c, weight of the curvature force F_c, S_r, scheme of evolution for the coefficient of the region-based force F_r. Output: $\phi^{(K_0)}$, the final level set function.

MRSTACS(f_d, f_p, K, K_0 dt, $\{\eta_{\mathrm{t}}^{(k)}\}$, λ_c, S_r)
  initialize level set function $\phi^{(K)}(0)$ based on edge detection in I_d and Euclidean distance transform
  for $k = K$ to K_0 **do**
    compute all the values of λ_r based on S_r
    decompose f_p to k levels, $I_p^(k)$
    convert $f_p^(k)$ to a binary image by thresholding
    for $\eta = 1$ to $\eta_{\mathrm{t}}^{(k)}$ **do**
      for all points on the contour **do**
        compute F_r, F_{curv}, RHS of (4.1)
      end for
      extend the values to the whole domain, get F_ϕ
      $\phi_{\mathrm{tmp}} = \phi^{(k)}(\eta) + dtF_\phi$
      for all points where the sign of $\phi^{(k)}(\eta)$ is changing **do**
        if nonsimple point **then**
          prevent change of sign of $\phi^{(k)}$: $\phi_{\mathrm{tmp}} = \epsilon \cdot \mathrm{sign}(\phi^{(k)}(\eta))$
        end if
      end for
      $\phi^{(k)}(\eta + 1) = \phi_{\mathrm{tmp}}$
    end for
    Compute $\phi^{(k-1)}$ from $\phi^{(k)}$ using (4.4)
  end for
  return ϕ

to an extent. However, we would have to be careful about the choice of the threshold values at each level. Hence, if the application does not require one to strictly quantify the performance against HS and a coarse but quick segmentation is required, MRSTACS is the ideal choice as it produces tight contours and is fast.

4.7 Pseudo-3D STACS

As mentioned, the different variations of STACS discussed thus far utilize the DNA channel to initialize the level-set function, used to segment the total-protein image. Further, we use the concept of topology preservation. There are a few problems with this approach.



(a) (b)

Figure 4.7: Segmentation results for MRSTACS on HeLa images. (a) Coarse segmentation at level three. (b) Segmentation on the original image. (Images courtesy of Dr. A. D. Linstedt [5]).

From the standpoint of accuracy, the DNA channel is not present in all of the slices of the z-stack (3D cell image), even though in the corresponding total-protein image, the cell is discernable. As a result, TPSTACS is not able to segment these cells, particularly in the peripheral slices, where corresponding initial contours are not detected. From the standpoint of performance, performing the initialization procedure several times, even though the slices of a z-stack are not independent, poses an unnecessary computational load. In addition, in the ideal situation, we would like to have an initial contour that is as close as possible to the final contour calculated by TPSTACS (as it happens during the refining phase of MRSTACS). Finally, by staining the DNA solely for the purposes of initialization, we unnecessarily increase the number of extraneous material added to the cell. If we are able to perform a reasonable blob detection on the original image itself (as we show is possible using MRSTACS), then only the channels necessary for the biological application need be imaged.

Given a parallel DNA channel, the pseudo-3D STACS (P3STACS) algorithm aims to address the problem of missing cells in the total-protein channel due only to the absence of DNA in the parallel channel. An analysis of the signal contained in the DNA and total-protein channels (reveals that most of the DNA is contained towards the middle slices in the z-stack. As the cell rests on the slide, it tends to spread out at the cell-slide interface, relative to its free top end. Thus, a large part of the cell-signal is present towards the bottom and tapers off towards the top. Thus, the strategy is to initialize the algorithm in one of the middle slices and use the final segmentation contour obtained on that slice to initialize the

level-set function for the previous and/or succeeding slices in the stack. The evolution itself of the level-set function for a 2D image is no different from TPSTACS (or any of its 2D variants). Using the information along the z-stack contributes to the appellation "pseudo 3D" for this approach. Algorithm 6 provides a pseudocode.

4.7.1 Results

This algorithm was tested with exactly the same parameters as those used to test TPSTACS, except the initialization on the DNA channel was run only for the middle slice. Every other slice was initialized using the final segmentation outcome of its previous (or succeeding) slice. P3STACS segmentation results, starting from the middle slice to the top of the stack, are shown in Fig. 4.8.

Apart from being able to detect cells that otherwise went undetected, P3STACS also affords a saving in the runtime. We summarize this saving corresponding to the image shown in Fig. 4.8 in Table 4.3. In this case, initialization using DNA channel was performed on slice 12.

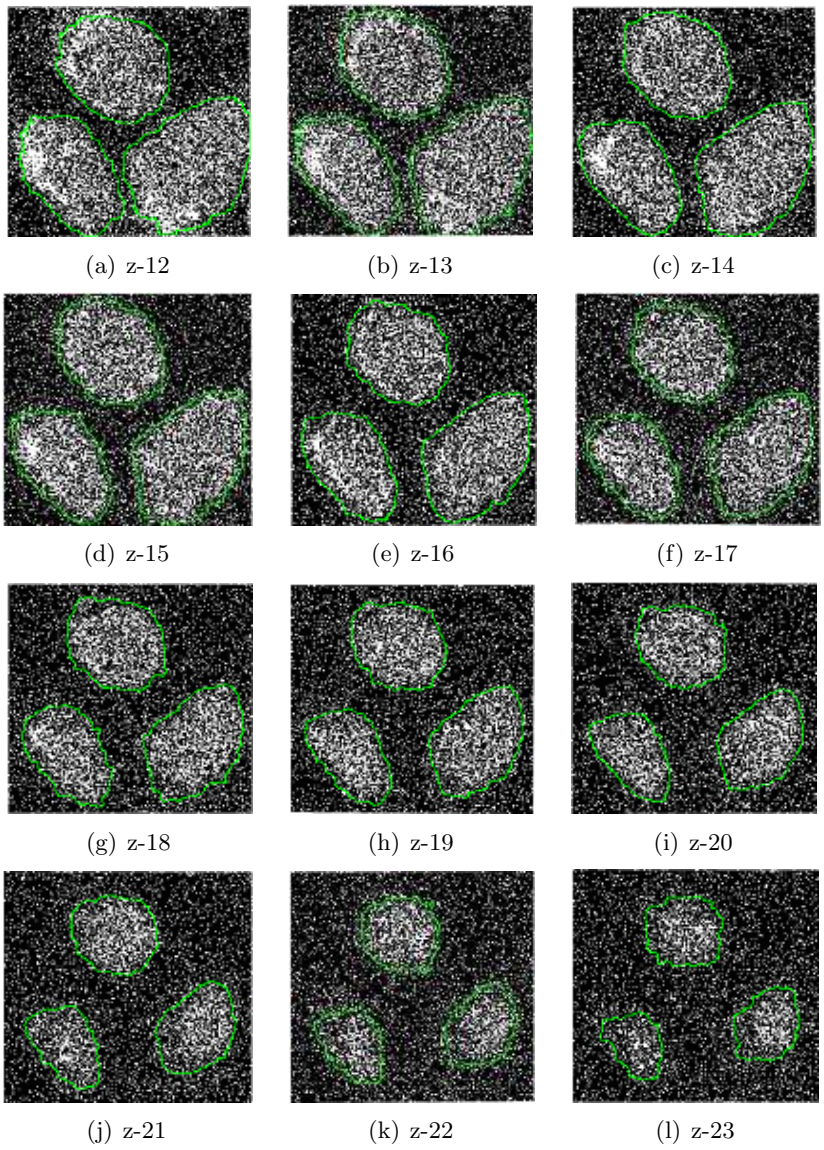
As a final word on P3STACS, we note that using the information from adjacent slices can be advantageous, but dangerous if faulty. To prevent an error from propagating or to recover from one, instead of propagating the results throughout the stack, we may stop to initialize afresh every few slices. This is particular true for thick cells that have a large number of slices (such as 40) in a z-stack.

In summary, in this chapter we have seen how STACS can be adapted to fluorescence microscope data and seen various modifications of the algorithm to meet different requirements, as well discussing its pros and cons.

Algorithm 6: **[P3STACS]** Input: f_d, a z-stack of DNA images, f_p, a z-stack of total protein images, dt, time step, scalar η_{t}, maximum number of iterations, scalar λ_c, weight of the curvature force F_c, S_r, scheme of evolution for the coefficient of the region-based force F_r. Output: ϕ, the final level set function.

P3STACS(f_d, f_p, dt, η_{t}, λ_c, S_r)

compute $A_p = [a_{p,l}]$ and $A_d = [a_{d,l}]$, approximate area of cells and DNA at each slice l
on slice $\mathrm{argmax}_l\{a_{d,l}\}_{l=1}^{L}$ of f_d, initialize level-set function ϕ based on edge detection and Euclidean distance transform
compute all the values of λ_r based on S_r
for the middle slice, convert I_p to a binary image by thresholding
for $\eta = 1$ to η_{t} **do**
  for all points on the contour **do**
    compute F_r, F_{curv}, RHS of (4.1)
  end for
  extend the values to the whole domain, get F_ϕ
  $\phi_{\mathrm{tmp}} = \phi(\eta) + dtF_\phi$
  for all points where the sign of $\phi(\eta)$ is changing **do**
    if nonsimple point **then**
      prevent change of sign of $\phi^{(k)}$: $\phi_{\mathrm{tmp}} = \epsilon \cdot \mathrm{sign}(\phi^{(k)}(\eta))$
    end if
  end for
  $\phi(\eta + 1) = \phi_{\mathrm{tmp}}$
end for
for all slices succeeding (or preceding) $f_d(l)$ in the stack **do**
  moving from the middle of the stack towards the extremities, one slice at a time, initialize level-set function ϕ using the final result from the previous (or next) slice
  η_t for the slice to be segmented is set at $0.85\dot{a}$ where $\dot{a}$ is the difference between the approximate areas of cells in the two consecutive slices
  for $\eta = 1$ to η_{t} **do**
    for all points on the contour **do**
      compute F_r, F_{curv}, RHS of (4.1)
    end for
    extend the values to the whole domain, get F_ϕ
    $\phi_{\mathrm{tmp}} = \phi(\eta) + dtF_\phi$
    for all points where the sign of $\phi(\eta)$ is changing **do**
      if nonsimple point **then**
        prevent change of sign of $\phi^{(k)}$: $\phi_{\mathrm{tmp}} = \epsilon \cdot \mathrm{sign}(\phi^{(k)}(\eta))$
      end if
    end for
    $\phi(\eta + 1) = \phi_{\mathrm{tmp}}$
  end for
end for
return ϕ



(a) z-12 (b) z-13 (c) z-14

(d) z-15 (e) z-16 (f) z-17

(g) z-18 (h) z-19 (i) z-20

(j) z-21 (k) z-22 (l) z-23

Figure 4.8: Segmentation results on applying P3STACS to the upper half of a z-stack of 24 images. The images have been cropped to highlight the change in cell areas with height, moving away from the slide. The caption above each image indicates the level of the image in the z-stack. (Original images courtesy of Dr. R. F. Murphy [14].)

Slice number	TPSTACS time[sec]	P3STACS time[sec]	Time saving[%]
$z-13$	2.0309	1.3163	35.2
$z-14$	2.0092	1.1327	43.7
$z-15$	1.9681	0.9677	50.8
$z-16$	1.9705	0.9005	54.3
$z-17$	1.9329	0.8500	56.0
$z-18$	1.9133	0.8036	58.0
$z-19$	1.9090	0.7404	61.2
$z-20$	1.7751	0.7076	60.1
$z-21$	1.6997	0.6711	60.5
$z-22$	1.6172	0.6405	60.4
$z-23$	incorrect	0.6419	NA

Table 4.3: Computational time for the traditional TPSTACS method for a z-stack slice of size 64×64 pixels and the corresponding computational time for P3STACS together with the saving.

Chapter 5

Derivatives of STACS for Other Modalities

Segmentation is a ubiquitous problem in imaging, particularly in biomedicine. Although a lot of advances have been made in automating the segmentation of medical images, most of them are not available for use. This is partly because of their proprietary nature, and or that they are not made available to researchers whose primary focus is not computation and hence do not want to invest time reimplementing ideas reported in the literature. At other times, the specificity of applications and/or rigidity of algorithms render the algorithms of no particular use for a new application. Some of our collaborating teams have images which they want segmented, preferably automatically, but no available algorithm that works to their satisfaction.

Segmentation demands posed by some of the images obtained from other imaging modalities overlap enormously with those that underlie the design of TPSTACS. While TPSTACS and its variants were designed primarily for the segmentation of fluorescence microscope images, as noted in Section 3.3, (3.3) allows us to include any number of forces adapted to the data. Given below are some of the applications of the algorithm, with suitable adaptations within the framework, to segment images from other modalities. We present these examples to demonstrate the flexibility of the STACS framework as it inspired the design of our new

framework in terms of having a flexible mathematical core, which can be suitably adapted to segment different types of data through the use of data-specific modules.

5.1 DIC Microscopy Images of the Yeast

The segmentation task is to separate yeast cells from each other and from the background in DIC images of budding yeast colonies. DIC microscopy is often used to study transparent organisms such as yeast. It is based on interferometry and provides a high clarity image of the biological specimen lending insight to properties such as its optical density. We now discuss segmentation of yeast images, introduced in Section 2.2.1.

5.1.1 Dataset

We are given a set of over a 1000 images of closely spaced budding yeast cells (yeast colonies) and are required to segment the individual cells in the DIC channel (see Fig. 5.1(c)). The dataset also has a parallel DNA channel and a GFP labeled specific protein channel (see Fig. 5.1(a) and 5.1(b)). This is a publicly available dataset [7].

We extend STACS to segment images of this dataset as DIC is commonly used for studying transparent organisms and affords edge information which we use to extend the repertoire of forces that can be used with STACS, and thus demonstrate its flexibility. We, however, do not rigorously quantify the performance of the adapted STACS algorithm on the entire dataset. We limit ourselves to a qualitative assessment of the performance of the DIC-Yeast STACS (DYSTACS) algorithm in this section. A recent work that discusses a graphical model approach to the segmentation of this dataset with promising success presents a rigorous analysis of the performance of the graphical model algorithm as well as the SW algorithm that is widely considered as one of the most accurate cell segmentation algorithms [76].

5.1.2 Algorithm—DYSTACS

Initialization. Although the dataset we use has a parallel DNA channel, the information is ambiguous and insufficient to yield an initial contour. Likewise, a specific protein chan-

nel comprising the distribution pattern of a GFP labeled protein is of no avail to us for initialization.

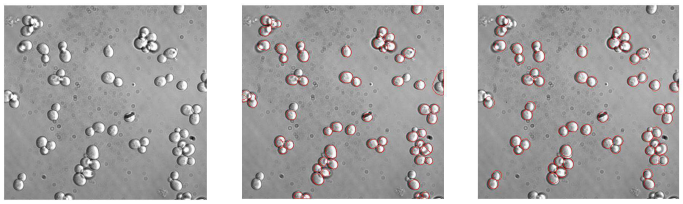
Thus, we initialize the contour by first distinguishing the clumps of cells from the background by thresholding the gradient image. Next, we get rid of blobs with holes larger than a pre-determined size, as invalid objects. Similarly we get rid of very small fragments that are unlikely to be cells. This results in a clump of cells that are still mostly merged. Using a combination of k-means clustering and the watershed algorithm, we merge the fragmented portions of a cell, while splitting spuriously merged cells. Unfortunately the above process is not infallible. Even in our best results, we end up mistakenly splitting cells due to the high variation in intensity, while failing to detect a few legitimate cells.

Evolution. The level-set function is evolved based on a region-based force and an edge-based force whose influence is designed to increase with the iteration number. For the region-based force, we use the mean intensities inside and outside the cells to drive the segmentation. We note that instead of the mean, perhaps the variance might be more discriminating for DIC images. While the intensity difference between the foreground and background is usually weak, the edges are quite pronounced in the DIC channel. Thus, edge maps are built using Canny edge detection (see Section 3.1). These edge-maps can be used as a stopping criterion to prevent the contours from picking up the background. Further, using topology preservation maintains the number of contours detected in the initialization stage, preventing merges to which this data is highly prone.

5.1.3 Results

Experimentation reveals that the challenge in this data is the initialization itself as we tend to get a good segmentation if we are able to first detect the cells accurately. Fig. 5.1(b) shows a representative result of the segmentation with automated initialization. Since the procedure is highly dependent on the initialization, Fig. 5.1(c) shows the result with a manual segmentation.

Discussion. Unfortunately, the automated initialization we use at present is plagued by errors. Even in our best results, we end up mistakenly splitting cells due to the high



(a) (b) (c)

Figure 5.1: Segmentation results for DIC microscope images of budding yeast. (a) A DIC microscope image of yeast cells. (b) Segmentation results after automated initialization and (c) segmentation result after a manual initiailization. (Original images courtesy of the Yeast GFP fusion localization database [7].)

variation in intensity, while failing to detect a few legitimate cells. Thus there is much scope for improvement in the initialization phase. The satisfactory performance of the algorithm after manual initialization suggests that the procedure would benefit much from a minimal user interaction (such as marking a single point in each cell). Moreover, as noted above, DIC presents a strong variation in the foreground region (within cells) as opposed to the background. Thus, we expect that using variance maps instead of mean intensities might improve the detection and segmentation of the cells [126]. Once we have a robust automated algorithm, a next step would be to compare our results with the recently proposed algorithm for the segmentation of these images based on graphical models [76].

5.2 MRI Images of the Heart

The task is to segment the heart walls—endocardium and epicardium—from a 2D time series of MRI images of a human chest cavity. The motivation is to understand wall motion in normal hearts to be able to characterize abnormal wall motion due to ischemia and aid in its early diagnosis. We now discuss the application of STACS to the problem detailed in Section 2.2.2.

5.2.1 Dataset

We are given a set of 25 MRI images of a human chest cavity about 40ms apart that correspond roughly to one heartbeat of a clinically healthy volunteer. The size of each image is 196×156 pixels. Along with these images, a set of HS images were provided as a reference for evaluating the performance of our algorithm. This dataset was provided by

Dr. Laine and his team at Columbia University [9].

We note that while the original STACS algorithm was also developed to segment the heart (and its chambers), the forces cannot be applied directly. This is primarily because the dataset underlying the design of STACS was a transplanted heart in a mouse. Thus, the considerations are different from those required for a MRI of the human chest cavity. Below we present our MRI Cardiac STACS (MCSTACS) algorithm with suitable forces for the segmentation of this dataset.

5.2.2 Algorithm—MCSTACS

The endocardium usually appears brighter than most structures in the chest cavity. Further, the epicardium is present around it. Thus, the strategy is first to segment the endocardium and use the result to segment the epicardium.

Endocardium initialization. The initial contour for an image obtained by first thresholding the image. Mere thresholding, however, picks up some other structures. Morphological operations are used to eliminate very small objects. To eliminate any remaining spurious detections, we use the fact that the endocardium is the most circular of these structures. Thus the object with the least eccentricity is retained. The signed distance transform is then used to initialize the level-set function.

Endocardium segmentation. During the course of evolving the contour, we use topology preservation as there is only one endocardium (as well as one epicardium). We use a region-based force that considers the mean pixel intensities inside and outside the endocardium as the inside and outside mean models respectively. Further, we use a shape-based force, computed as a dilated version of the initial endocardium detection to ensure the contour remains roughly circular in shape and does not leak out to include other structures. Finally, the curvature force used in TPSTACS (see (3.3)) is used to maintain the smoothness of the contour. The annealing schedule emphasizes the ballooning force in the beginning, the region-based force in the middle and the curvature force towards the end of the evolution procedure to ensure a satisfactory segmentation. Since the initialization detects almost all of the endocardium, just a few iterations of the algorithm is sufficient to refine the contour

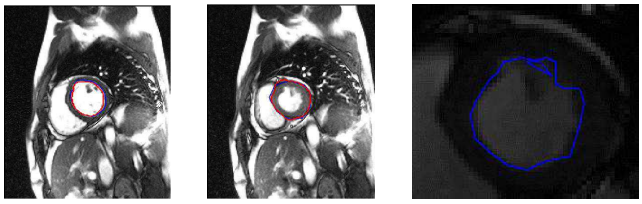
and add the missing portions.

Epicardium initialization. The final contour obtained for the endocardium serves as the initial contour for the segmentation of the epicardium.

Epicardium segmentation. The same algorithm as that used to segment the endocardium with appropriately tuned statistical models for the region-based force, is iterated a few times to segment the epicardium. A shape-based force, derived as a dilated version of the initial endocardium detection, is used as a stopping force to ensure the contours do not leak outside the heart into other structures in the chest cavity. In addition to these two forces, we use a constant expansion force or a *balloon force* to ensure the contour only expands [62] as the epicardium encompasses the endocardium.

5.2.3 Results

A representative image of the outcome of the endocardium segmentation is shown in Fig. 5.2(a). The outcome of the epicardium segmentation is shown in Fig. 5.2(b). We observe that the algorithm works well even when there is no edge information present. An entire frame of size 192×156 is completely segmented with a high degree of accuracy in less than fifteen seconds on a 1.8GHz Pentium M Processor. The average area similarity compared with the hand segmented ground truth is 93% for the endocardium and 94% for the epicardium. The best and worst performances together with the averages are reported in Table 5.1.



(a) (b) (c)

Figure 5.2: (a) Segmentation of the endocardium and (b) epicardium. (Red contour: TPSTACS. Blue contour: hand segmented ground truth.) [15] (c) An example of a self-loop and nonsmooth contour in the hand segmented ground truth. (Original images courtesy of Dr. A. F. Laine [9].)

Discussion. A high segmentation accuracy and reasonable runtime indicate MCSTACS is a viable alternative to HS of heart walls from chest MRI images. A higher accuracy for

		AS	
	Best[%]	Worst[%]	Average[%]
Endocardium	95.85	90.21	93.01
Epicardium	95.81	92.67	94.20

Table 5.1: Segmentation results for the application of endocardium and epicardium segmentation. Area similarity with HS of the segmentation performance of MRI-chest STACS [15].

epicardium segmentation despite the relatively lower accuracy for its initial contour—the endocardium segmentation—could be explained as follows. For a few of the frames, the discrepancy between the contour from the algorithm and the ground truth is due to the inconsistency and/or inaccuracy in the hand segmented contour (see Fig. 5.2(c)) rather than any inaccuracy on the part of the algorithm. Though the ground truth is provided by an expert, such anomalies are expected because the process of hand segmenting the images is a tedious one.

Some of the future directions include tuning the algorithm to be robust to noise in the data especially in peripheral sectioning planes and also in case of data from high-speed MR imaging or in ultrasound images that are also used in such studies. Further, akin to P3STACS, we can use the time series information and cyclic nature of the heartbeat to our advantage in initializing MCSTACS for a given time-series data.

5.3 fMRI Images of the Brain

The task we focus on for this application is segmentation of white-matter. In brain fMRI images, the functional activity under study is detected in the grey matter, which is obtained by growing layers of it on top of segmented white matter. Since the functional information is not particularly useful in distinguishing the white matter, we treat the fMRI dataset as a time series of MRI, which is three-dimensional and can be sectioned into 2D images (see Fig. 2.7(b)). We focus on the segmentation of brain MRI images and discuss below the adaptation of STACS to the problem detailed in Section 2.2.3

5.3.1 Dataset

We report the results of testing our algorithm on the MRI images of a child brain made available to us by the VISTA laboratory at Stanford University [11]. It consists of 400 images—100 in the coronal plane, 140 in the sagittal and 160 in the axial sectioning planes—each of size 217×180 pixels. The VISTA lab also provided us with human expert-based segmentation for the left hemisphere to evaluate our algorithm.

5.3.2 Algorithm—VBSTACS

The challenge in segmenting MRI brain images stems from the highly convoluted nature of the grey matter. While voxel-based classification methods that use image statistics and histogram-based thresholding are partially successful, they are affected by intensity variations and the need to maintain distinct templates for various brain pathologies [127, 128]. Active-contour-based methods, on the other hand, have considerable advantages due to their topological flexibility [129].

Previous work. An automated initial segmentation used by the Stanford team results in several anomalies which necessitate hours of involved manual postprocessing to clean up the results [44]. The technique uses prior knowledge of the structure of the human brain to aid the segmentation and has been made publicly available as a MATLAB toolbox called mrGray [130].

Due to the difficulties in segmenting the gray matter directly, the authors segment only the white matter and the cerebrospinal fluid (CSF) regions in the image. Once the white matter has been segmented, the proposed technique reconstructs the gray matter surface using a method of "constrained growing-out from the white matter boundary". White matter segmentation is performed by building statistical models of pixel intensities for the white matter and nonwhite matter regions of the brain followed by a step that assigns, to each pixel intensity value, a label based on a novel maximum *a posteriori* probability (MAP) estimation algorithm. However, the segmented white matter regions obtained from this step occasionally have holes and handles within them. Since actual white-matter regions cannot

possess such holes or handles, all voxels that lie in these holes or handles are relabeled as white matter. Finally, layers of gray matter are grown iteratively, starting from the edge of the white matter region. The number of layers of gray matter that need to be grown is taken as an input from the user. Care is taken to ensure the connectivity of gray matter regions within and across layers. In the algorithm, the quality of the white matter segmentation directly affects the quality of the final 3D gray matter topology.

In our effort, we address the issue of the white-matter segmentation using an active-contour based approach, as opposed to the MAP-based approach used in [44]. The improved segmentation can then be used as an input to the hole-filling and gray-matter-growing steps of mrGray. The specific goal of this work is to improve the accuracy of segmentation so as to significantly reduce, if not eliminate entirely, the need for manual postprocessing, and thus reduce wasted resources both in terms of money and time.

Initialization. We obtain an initial contour for the white-matter segmentation by thresholding the image, f. The threshold is empirically determined and easy to pick as the white matter is usually brighter than the gray matter and CSF. Then, a few morphological operations are performed to eliminate isolated regions in the periphery and retain a meaningful result. This circumvents the need for manual intervention in initializing the contour (as required for original STACS).

Evolution. The initial contours evolve based on the forces applied on them. Since the model statistics of the brain MRI image are fairly strong, with the white matter being fairly distinct from the nonwhite matter, we chose a region-based force to drive the segmentation. To compute the actual force, we first represent both the white matter and the nonwhite matter by their respective statistical models. We compute the statistical models by selecting a few representative slices from the volume as training data and computing the discriminating statistics for the foreground and background regions. We exclude the images used as training data later in the testing phase to quantify the algorithm's performance. The two statistical models are based on the mean pixel intensities of the white matter (foreground) and nonwhite matter (background) regions. The foreground and background regions exhibit

significant intensity variations, with the regions near the edges exhibiting lower or higher intensities than the means of their respective regions. Therefore, while deriving the model statistics from the training images, we compute the model mean intensities using only small windows around the foreground-background edges (specifically, the edge between the white and non-white matter). The evolution of the level-set function follows (4.1).

This annealing schedule that modulates the region-based force ensures that the contour changes shape rapidly in the beginning of its evolution and gradually slows down as it approaches its desired location. Other forces such as the shape and smoothness-based ones are inapplicable since the white matter region possesses neither smooth edges nor a well-defined shape that can be used as a template. Moreover, while the edge-based force seems like a good candidate for inclusion, it actually degrades the quality of our final segmentation due to a number of spurious edges (such as the edges between the gray matter and CSF regions) in the edge map of the image. Therefore we exclude the edge-based force from the algorithm.

Voting. The modified STACS implementation in our discussion so far implicitly assumed segmentation on a 2D data set, while the brain MRI image data set is actually three-dimensional. We can harness additional information available along the third dimension to improve the robustness of the algorithm and the results of the 2D segmentation. We may slice a given 3D image into a collection of parallel 2D images along three orthogonal axes or planes (commonly referred to as axial, coronal and sagittal planes by biologists). Although performing segmentation on a collection of 2D slices from any one plane is sufficient to label each voxel as either white matter or nonwhite matter, we can build some redundancy into the system by segmenting all slices along each one of the three planes. Since there are three orthogonal planes that pass through each voxel in a 3D image, the segmentation result of each of these three planes would independently label a voxel as either white matter or nonwhite matter. The "vote" from each plane can then be combined using a simple majority voting procedure to yield the final label for that voxel.

5.3.3 Results

A qualitative assessment of the segmentation outcome is made by visual inspection. Fig. 5.3 shows the segmentation along the three sectioning planes.

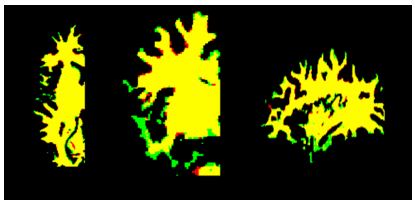


Figure 5.3: L to r: Results of segmenting the white matter of the left-hemisphere in the axial, coronal and sagittal planes. The yellow/red/green labels correspond to true positives/false positives/false negatives, respectively [16]. (Original images courtesy of Dr. B. A. Wandell and Dr. M. B.-S. Chechik [11].)

Table 5.2 reports a quantitative assessment of four possible implementations: voting-based segmentation (Voting), segmentation based on axial slices only (Axial), segmentation based on coronal slices only (Coronal) and segmentation based on sagittal slices only (Sagittal). From the table we note that the voting-based technique achieves the best performance on each performance measure.

	$T^{(+)}[\%]$	$F^{(+)}[\%]$	$AS[\%]$
Axial	86.26	2.32	82.83
Coronal	89.02	2.55	80.76
Sagittal	86.21	1.79	81.92
Voting	**89.13**	**1.63**	**85.07**

Table 5.2: Segmentation results for the application of white-matter segmentation in brain fMRI images. Quantitative measures of the segmentation performance of voting-based STACS. $T^{(+)}$ indicates true positives that is, pixels correctly identified as belonging to the white matter region, $F^{(+)}$ indicates false positives that is, pixels incorrectly identified as white matter and AS denotes area similarity [16].

Discussion. Typically, for AS given by (4.3.1), $AS \geq 70\%$ is considered excellent agreement with the ground truth [131]. Here we used a more stringent variation of the measure,

$$AS = \frac{n(HS \bigcap VBSTACS)}{n(HS \bigcup VBSTACS)}.$$

Thus, we note that the outcome of VBSTACS for fMRI images is in good agreement with the ground-truth provided to us. Further, one of the major advantages of voting-based STACS over segmentation using only one axis is that the voting scheme is able to pick up white-

matter regions at the extremities of the MRI image, whereas these are often missed in single-axis segmentation. Fig. 5.4 shows the segmentation results using (a) coronal segmentation and (b) voting-based segmentation for one of the peripheral axial slices. Clearly, coronal segmentation entirely misses the entire white matter region in this slice while voting-based segmentation is able to identify it quite accurately.

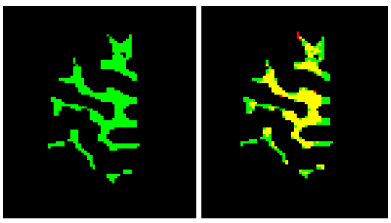


Figure 5.4: A peripheral slice in the coronal plane segmented using (a) coronal segmentation only and (b) voting-based segmentation [16]. (Original images courtesy Original images courtesy of Dr. B. A. Wandell and Dr. M. B.-S. Chechik [11].)

It is noteworthy, however, that the algorithm takes nearly 36 hours to segment nearly 600 images (approximately 12 hours per sectioning plane). While this is still faster and less tedious than three days of manual labor, which is what it takes in the absence of an automated method, the run time is still quite high.

5.4 Data-Specific Modules and Design Considerations

Above and in the previous chapter, we have seen how an active-contour-based framework, STACS, can be adapted to segment different imaging modalities. As described in Chapter 1, this is reminiscent of the problem we are endeavoring to solve. The computational task in each of these cases—segmentation—is the same, while the specific challenges, and hence modifications, are different. We can thus think of any instantiation of the framework to be given by a *state vector* with appropriate parameters for each of the possible *states*, indicating for example, the type of initialization, choice of forces and their weights, the stopping criterion and auxiliary modules like topology preservation to be used for the task at hand.

In the sections below we summarize some of these states and possible assignments they

may take on based on the applications we have discussed so far. For each state, we review the data-specific issues we have come across in the algorithms presented so far and highlight a few others. We also briefly discuss a few application-specific modules that use the segmentation outcome as the input and may be included in the system for relevant applications. As all these considerations provide the impetus for choosing a combination of modules for a given modality/dataset to best segment the images from it, we also note wherever relevant some of the limitations in the present design and identify what we would like to incorporate in a new framework. Finally, we discuss the issue of making these tools available for end users and algorithm developers to facilitate testing and further development.

5.4.1 Initialization

Active contour models, particularly geometric active contour based methods (see Section 3.3.2) are sensitive to initialization. We have seen that in small applications, such as the original STACS problem, manual segmentation was viable [67]. For fluorescence microscope cell images that have a parallel DNA channel, or in the case of segmenting the human heart from chest MRI images or white matter from brain fMRI images, we may use thresholding and morphological operations to obtain initial contours. In the case of fluorescence microscope images without a parallel channel, it is possible to use MR techniques to perform blob detection directly on the cell (total protein/membrane protein) image. Finally, for DIC yeast images as well as tissue-type images that do not have a parallel DNA image, a more involved initialization procedure may be necessary—one that uses the edges as well as rules based on a few representative cells—to eliminate spurious detections. Further, if the dataset is tractable, particularly for z-stack and time-series images, we note that the segmentation would benefit from a minimal manual intervention in correcting the automated initialization. This could be through just clicking a point on a cell that has not been detected or that is required to be deleted. There have been methods, particularly when topology preservation is not an issue, that have initialized the algorithm with a uniform set of circles for 2D images (or cylinders for 3D volumes). As the level-set based approach is capable of adapting to the topology, having a large number of contours is not a

problem. However, the convergence may be expedited and more accurate with an accurate initialization.

Ideally, in any further design, we would aim to minimize the dependence of the algorithm on the initialization.

5.4.2 Topology Preservation

Topology preservation can be used in the sense of respecting the underlying image topology in terms of statistical homogeneity or some underlying shapes in the image. The formulation of level-set methods inherently affords this type of topology preservation because a single level-set function can be used to represent multiple contours and during the course of evolution, the level-set function gracefully adapts to topological changes in an image by merging or splitting these contours.

In the case of cell segmentation and heart-wall segmentation from chest MRI images, we have used topology preservation in the sense of conserving the number of contours from the initialization phase. The criterion for topology preservation is based on digital topology that studies a 3×3 neighborhood around every point on the contour and prevents change of sign if the point is so-called nonsimple. This does not add a significant overhead to the evolution procedure itself. However, this procedure overrides the basic advantage of level-set methods over their parametric counterparts of gracefully adapting to topological changes in an image. Being an external constraint applied just when the topology is about to change, it introduces abruptness in the contour. Further, such a topology preservation scheme is useful only when the initialization accurately provides the desired number and approximate location of the desired final contours. Otherwise, topology preservation would only serve to propagate errors in the initialization phase. In this sense topology preservation is closely tied to the initialization phase and makes it worth investing time to improve the initialization to facilitate accurate segmentation. An alternative digital topological constraints is topological flows, which introduces topology preservation as one of the forces that drive the evolution phase [132]. Topology preservation is not an issue in applications such as MR brain image segmentation.

In our further design, if topology preservation is necessary, we would like it to be a part of the evolution rather than a separate external constraint.

5.4.3 Multiresolution

As seen in Section 3.5.2, MR affords many advantages. First of all, MR can be used to reduce the runtime of an algorithm, as we demonstrated in Section 4.6. This is done by appropriately applying the segmentation algorithm designed for the original resolution on a very coarse approximation of the image and successively refining the result [80]. Next, particularly in the case of fluorescence microscope images of punctate patterns, the smoothing built into MR improves the discernibility of cells. This allows us to perform a blob detection on a channel such as the total protein in the absence of a parallel DNA image. Moreover, the vast choice of filters as well as the number of levels and extent of redundancy make MR a powerful module. However, the extent to which MR is used (or not) depends on the nature of the data and specific task. For example, if are working on a very low resolution image to begin with, it would not benefit from the downsampling provided by MR. Likewise, if there is a strong edge-information, then using a coarse (smoothed) version of the image would not be advisable.

Ideally, MR would be a part of the framework with the choice of parameters determining the extent and nature of its use for a particular segmentation task.

5.4.4 Evolution Forces

In any active-contour-based framework, forces form the crux of any algorithm. There are many options for the external force, based on the imaging modality and application. For example, region-based force can be statistical measures such as the mean number of pixels in a binary (thresholded) image, mean intensity values or mean and variance of pixel intensity, or just the variance or texture features derived from a gray-scale image. For vector-valued images, color can also be used in the region-force [133]. Shape-based forces can be useful when the object of interest has a shape that can be approximated and parameterized. Cardiac image segmentation is a candidate for this force (for example, both the original

STACS as well as MCSTACS benefit from shape-based forces). Some images, such as those of DIC, have discernable edges and edge-based forces can be useful to attract the contour to the image boundaries or as a stopping criterion. Examples for the internal force include smoothness force based on curvature of the contour or total length of the area included in the contour.

Thus ideally, we would build a vast repertoire of forces based on the particular applications we deal with so that any combination of forces may be called upon as need-based. In designing these forces, we would pay particular attention to the specific features of the objects of interest, particularly for fluorescence microscope images of punctate patterns, to extract those that will aid in the task segmentation.

5.4.5 Stopping Criterion

Ideally, the forces used to drive the segmentation are designed to balance out, so the algorithm (curve evolution) comes to a natural halt. In practice, however, the large variation in image properties together with numerical errors introduced due to translating the continuous-domain theory to a discrete-domain implementation, seldom lets this happen.

The stopping criterion we have used in all the STACS-based algorithms is that of an iteration number. An optimal number of iterations can be empirically determined *a priori* based on training data. Training data would consist of a few representative images, along with ground truth. However, depending on the specimen, imaging modality, extent of noise in the images and volume of the data, this can be an involved process. For the segmentation of a z-stack of 3D images for example, the size of the cells may change along the z-stack, and hence the iteration numbers have to be appropriately changed.

Another stopping criterion that could be used is the number of pixels that have changed their state (from "out" to "in" with regard to the evolving curve) between iterations. When the initial contour is contained in the object of interest, the difference in area between contours is large in the beginning as the curve expands rapidly. As it gets closer to the desired object's boundary, the number of pixels included in the curve or excluded from it is small. However, this too cannot serve as an exact measure, particularly when multiple

objects are being tracked and the curves segmenting them are growing at different rates.

Ideally, in our future design forces are constructed based on a digital grid and evolve in such a way that the algorithm comes to a natural halt (that is, zero pixels change their state).

5.4.6 Application-Specific Post Processing

As explained in Chapter 2, segmentation is usually the first step after image acquisition. It is almost never an end in itself. Various post-processing modules are likely, depending on the application. For example, in the case of brain fMRI, the postprocessing comprises gray-matter growing and visualization. For PSL images, the processing that succeeds segmentation is classification. We can design some simple postprocessing modules to either render the segmentation outcome suitable to be input to existing postprocessing systems, or if simple enough, provide the required postprocessing directly. An example of the latter category would be counting the number of cells segmented and measuring their volume. While this is a small step for us, it would be useful to our collaborators who can invest their time in designing the experiments and inference rather than HS and manual postprocessing.

In our future design, we would endeavor to include a few such postprocessing modules, suited to the applications we focus on.

5.4.7 Graphical User Interface

Embracing the ideas of reproducible research [134,135], we intend to make the software implementing our algorithms freely available. The prototype scripts used for our publications are available as reproducible-research compendiums, accompanying our papers on the web (see for example [136]). While this affords our collaborators as well as other end users and algorithm developers to test our algorithms, we recognize that providing a software package with a user-friendly GUI, would especially facilitate the use of the tools we develop. In particular, as imageJ [137] is popular with biologists and publicly available, we intend designing plugins for ImageJ.

The advantage of using a popular platform such as ImageJ and providing a graphical

interface is that it would ease the interaction for a first-time user, as the GUI will provide example images along with sample results. This way the users can train themselves by changing the parameters of the algorithm. In the process, they will gain more intuition than if given the definitions of the underlying concepts, such as the forces. As discussed in Chapter 1, the GUI will integrate the algorithms with the mathematical core and the different modules to tackle various data-specific scenarios that we come across.

To elaborate further, for the end users segmentation is typically the first step towards answering their biological question. They need not concern themselves with all the modules and understand the state-vector values for a particular configuration etc. All they would have to do is indicate to the software the experimental conditions, such as the imaging modality, dimensionality, resolution at which the image was taken (if possible) and the nature of the image (such as a cell image that has a total protein channel and no parallel channels). The GUI will also provide auxiliary post-segmentation tasks that the biologists may find useful such as measuring the cell volume. The GUI will guide them in the process of assigning the appropriate value to the state vector that will in turn decide the data-specific modules to be appended. We intend to provide example images from the data sets on which the algorithm has already been trained, as well as set the default parameters based on these example images. The user will be guided to tune these parameters based on intuitive descriptions such as the cell size, for segmenting their images. We do recognize that in some tasks, such as high-throughput imaging or high-content screening, 2D images cannot possibly be segmented one at a time. In this core, the user investing a few minutes to configure the algorithm through the GUI will greatly enhance the quality of the segmentation output of the batch processing. Providing the tool as a plugin to ImageJ would also be useful to algorithm developers.

5.5 Beyond the Active Contour Framework

While active contour methods are the state-of-the-art tool in the image segmentation community, the punctate patterns of fluorescence microscope cell images (see, for example, Fig 4.3 and Fig. 4.7) makes the direct application of such techniques difficult [138]. Multi-

scale methods, on the other hand, are well suited to extract the significant features of such images.

The nature of the fluorescence microscope images motivates us to explore the use of multiscale features for their segmentation. At the same time, our success with the active contour algorithms discussed in Chapters 3 and 4 encourages us to continue to benefit from the advantage of the flexibility they afford. Recently, there has been an effort to formulate multiscale active contours [84]. Our motivation stems from the need for an elegant framework that has an efficient implementation. Further, the idea of computing forces at the points on the contour using an FFT implementation has been proposed recently in the context of topology preserving flows [132]. Yet, this design does not get rid of the time consuming extension function itself.

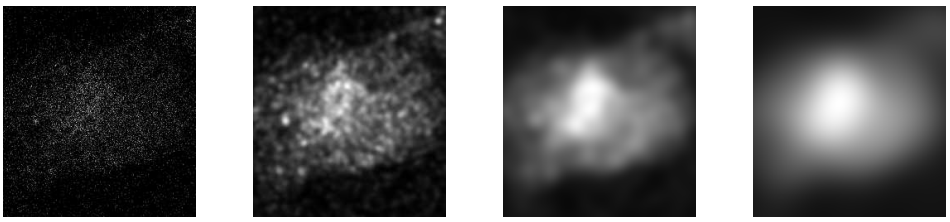
In a sense, though these level set formulations are based on the idea of embedding the contour, the extension function, which keeps track of and accesses the points on the current contour explicitly, is tantamount to implicitly parameterizing the contour. Thus, we come up with the novel idea of using *multiscale active contour (MSAC) transformations* for segmentation, which does not parameterize the geometrically embedded contour in anyway. We draw inspiration from works along very similar lines [65, 139]. These algorithms use the level-set function to keep a track of the foreground and background *regions* and use a discrete grid to evolve the contours respectively.

Chapter 6

Multiscale Active Contour Transformation

In this section we present a framework that is active-contour based but uses a transform perspective to improve efficiency, and a multiscale perspective that is particularly suited to the segmentation of fluorescence microscope images. Our basic goal in this framework is to eliminate the velocity-extension function that is the bottleneck in the active-contour based algorithms discussed in Chapters 4 and 5.

When confronted by punctate patterns in images, such as those of fluorescence microscopy, one naturally seeks to connect the dots, that is, to estimate the densities of bright pixels on local neighborhoods of various sizes. Mathematically speaking, the computation



(a) (b) (c) (d)

Figure 6.1: Multiscale blurs of a fluorescence microscope image. (a) The original image. (b) A slight blur begins to reveal the cell's edge. (c) With more blur, the edge of the cell becomes clearly defined. (d) With too much blur, the edges become rounded.

of such *local densities* may be represented as a multiscale smoothing transformation of the original image. To be precise, let $f : \mathbb{R}^d \to \mathbb{R}^c$ be a fluorescence microscope image, where d is typically 2 or 3 and c is the number of channels, with each channel corresponding to a different fluorescent probe. Although the proposed techniques can be extended to the case $c > 1$, for the sake of simplicity, we assume $c = 1$ for the remainder of this section. Given a windowing function g of an appropriate degree of smoothness, boundedness, and decay (for example, a Gaussian), the corresponding multiscale transform of f is $Wf : \mathbb{R}^d \times (0, \infty) \to \mathbb{R}$,

$$(\mathrm{W}f)(x; a) \;=\; \int_{\mathbb{R}^N} f(x) g(\frac{x - y}{a}) \,\mathrm{d}y. \tag{6.1}$$

As depicted in Fig. 6.1 for several particular scale parameters a, such a transform has a blurring effect, with the blur increasing with a. Although blurring is usually regarded as an unwanted artifact in image processing, for fluorescence microscope images, a moderate degree of blur is actually useful: edges and shapes become more apparent as the blur resolves the speckled image. Indeed, more generally, if f were a distribution composed of a weighted sum of Dirac measures, then for each $a > 0$, $\mathrm{W}f(\cdot\,; a)$ would be the sum of translations of a dilated version of g. Mathematically speaking, the above transformation (6.1) is relatively simple, being nothing more than a convolutional operator whose kernel contains a scale parameter. Today, such transformations are commonly called continuous wavelet transforms. Although trivial to prove, it is important to note that transform (6.1) preserves the action of the group of rigid transformations, and therefore preserves the underlying geometry of the original image. As microscope images are, in general, taken directly from above, and the vertical axis is the only axis along which we should expect the cells to be aligned, ideally our algorithms should not be affected by translations, dilations or rotations.

Specifically, given $x_0 \in \mathbb{R}^d$, $a_0 > 0$ and a $d \times d$ orthogonal matrix Θ, the corresponding translation, dilation and rotation operators are $\mathrm{T}_{x_0}, \mathrm{D}_{a_0}, \mathrm{R}_\theta : L^p(\mathbb{R}^d) \to L^p(\mathbb{R}^d)$,

$$(\mathrm{T}_{x_0} f)(x) \;=\; f(x - x_0), \quad (\mathrm{D}_{a_0} f)(x) \;=\; f(a_0^{-1} x), \quad (\mathrm{R}_\Theta f)(x) \;=\; f(\Theta^{\mathrm{T}} x).$$

The present day active contour methods are not sensitive to translations, dilations or rotations of the image. Thus, the MSAC transform that we propose should not introduce these defects and should at the least do no worse than the existing algorithms. Indeed, for any windowing function g, the corresponding transformation (6.1) is easily shown to preserve translations and dilations, according to the following rules:

$$(\mathrm{WT}_{x_0} f)(x; a) \;=\; (\mathrm{W}f)(x - x_0; a), \quad (\mathrm{WD}_{a_0} f)(x; a) \;=\; a_0^N (\mathrm{W}f)(a_0^{-1}x, a_0^{-1}a).$$

Transformation such as (6.1) will also preserve rotations, provided the windowing function g is itself radially symmetric: $g(x) = \widetilde{g}(\|x\|)$ for some $\tilde{g} : [0, \infty) \to \mathbb{R}$. Then, $(\mathrm{WR}_\Theta f)(x; a) = (\mathrm{W}f)(\Theta^{\mathrm{T}} x; a)$.

Although multiscale transformations have only recently begun to be applied to segmentation problems [83, 84], convolution transformations such as (6.1) (without the scale parameter) have been exploited by the segmentation community for decades. We argue that the scale parameter is essential, as the resolutions of the cameras used in the acquisition process will vary across biological research teams, and, more importantly, as any single fluorescence microscope image will contain multiple regions in which the marked proteins appear with varying density. Moreover, as our object of interest was the entire cell and not just its boundary, we do not want to continue computing only the "contour" as such. This criterion is in line with our goal of obviating the velocity-extension function. Algorithms that use a grid to evolve the level-set function or those that use the level-set function only to keep track of the "masks" of the regions of interest are much more efficient than strictly using a signed distance interpretation of the level-set function [139–141].

Inspired by these algorithms and to eliminate the extension function, we decided *(1)* to use the sign of the values in the embedding function to keep track of the regions of interest, while the values no longer strictly represented distances from the current contour and *(2)* to redesign the computation of forces so they would be computed *everywhere*. Thus, we cast the forces as convolutions rather than as differential equations computed only at the points on the contour. While departing from the distance transform interpretation

allowed the segmentation to proceed in large jumps and reduced the number of iterations required to achieve the desired segmentation, redesigning the forces allowed us to use FFTs to dramatically improve the computational speed at each iteration.

We called this the Multiscale active-contour (MSAC) transform as force computations are implemented as transforms and the filters therein have the potential to be used at multiple scales [15].

6.1 MSAC Transform Segmentation of Fluorescence Microscope Cell Images

We have already seen that forces used in MSTACS provide an accurate segmentation of fluorescence microscope cell images, in separating the cells from the background. In this section, we discuss how the forces in the traditional active-contour framework described in (4.2) and (4.2.2) can be recast in the MSAC Transform framework to segment fluorescence microscope cell images.

6.1.1 Dataset

We use a set of 15 z-stacks containing 3-8 HeLa cells provided to us by Dr. A. D. Linstedt and his team [5]. Each 2D image is of size 1024×1344 pixels. The HeLa cells are double labeled with the COPII subunit Sec13 (a cytosolic protein peripherally associated with the membrane). As the staining produces a diffused cytoplasmic pattern, the challenge posed by this data is that the signal is not uniform within the cell. Moreover, this set does not have a parallel channel of nuclear images that allows for an automated and accurate initialization.

6.1.2 Algorithm

Initialization. We initialize the embedding-function as identically zero. We surmise that the image statistics and contour properties would drive the segmentation to a reasonable segmentation outcome despite the inaccurate initialization.

Evolution. We use the region-based and contour smoothness based forces to drive the segmentation as these have proved to be useful from previous methods. The multiscale

transform in (6.1) can be combined with the active contour method by suitably coding in the information of the evolving masks. For example, in TPSTACS, the region-based force for the fluorescence microscope image in (4.2) is computed as the difference between the sum of the mean densities (or intensities) around a small neighborhood of every point on the current contour and the sum of the precomputed mean models. This may be computed without looping over the number of points on the contour as follows:

$$\begin{aligned} M_{\text{in}} &= \frac{\int_{\mathbb{R}^d} f_{\text{in}}(x) g(a^{-1}(x-y))\,\mathrm{d}y}{\int_{\mathbb{R}^d} C_{\text{in}}(x) g(a^{-1}(x-y))\,\mathrm{d}y + \epsilon}, \\ M_{\text{out}} &= \frac{\int_{\mathbb{R}^d} f_{\text{out}}(x) g(a^{-1}(x-y))\,\mathrm{d}y}{\int_{\mathbb{R}^d} C_{\text{out}}(x) g(a^{-1}(x-y))\,\mathrm{d}y + \epsilon}, \\ F_r &= M_{\text{in}} - \hat{M}_{\text{in}} + M_{\text{out}} - \hat{M}_{\text{out}}, \end{aligned} \tag{6.2}$$

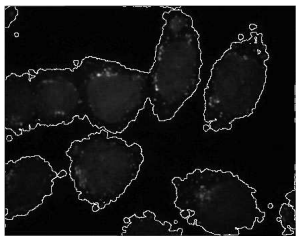
where d is the dimension of image f, g is a lowpass filter with scale parameter a, M_{in} and M_{out} are the mean intensities inside and outside the current contour, $\hat{M}_{\text{in}}$ and $\hat{M}_{\text{out}}$ are the model mean intensities inside and outside the object(s) of interest, f_{in} is the portion of the image contained in the current contour, f_{out} is the portion that is considered as the background by the current contour, C_{in} and C_{out} are the binary masks of the regions inside and outside the contour, respectively, $\epsilon > 0$ is a small correction factor to avoid dividing by zero, and F_r is the region-based force.

As the region-based force alone is insufficient to produce smooth contours, we use a contour smoothness force akin to the curvature force in (4.2.2). The new contour-smoothness force is computed using the embedding function ϕ without computing its second order derivatives (unlike the STACS force described in (4.2.2)) as

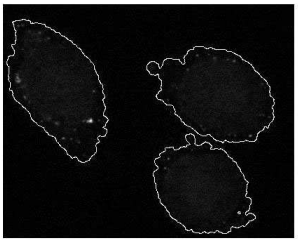
$$F_c \;=\; \int_{\mathbb{R}^d} \phi(x) g(b^{-1}(x-y))\mathrm{d}y - \phi, \tag{6.3}$$

where g is a lowpass filter with scale parameter b.

Specific filters in the force computations may be selected based on the objects of interest. For example, for segmenting HeLa cells, experiments reveal that a Gaussian filter is suitable



(a) Topology preservation an issue



(b) Topology preservation not an issue

Figure 6.2: Segmentation contours on a HeLa image (stained for sec13). The embedding function, ϕ, was initialized to be identically zero. (a) The merges are due to the absence of topology preservation. (b) Cells are sufficiently spaced to prevent merges despite the absence of topology preservation [15]. (Original images courtesy of Dr. A. D. Linstedt [5].)

for the forces in (6.2.2) and (6.3). [1] Moreover, the filters chosen can be applied at multiple scales to adapt to different resolutions of the image as well as different scales of the objects of interest at any given a resolution.

6.1.3 Results

We initialize ϕ to be identically zero and obtain reasonable contours in 3 iterations, each iteration taking roughly 8 seconds on an average (on an Intel Pentium M 1.6 GHz platform) for images of size 1024×1344. Sample results are shown in Fig. 6.2.

Discussion. In essence, the algorithm is similar to MSTACS, other than the fact this does not require the velocity-extension function because all values of the level-set function are computed during the force computation phase. Apart from being adaptable and meeting the basic goal of improving the computational efficiency, the design affords us an additional benefit: An involved initialization was not necessary for MSAC transform, as the all-zero embedding function would allow for a satisfactory separation of the region of interest with as few as three iterations (see Fig. 6.2(b)).

We still need to set the number of iterations *a priori* as the image statistics are such that they never actually equal the model statistics and thus (6.2.2) is always a nonzero quantity, forcing some movement or the other, even if oscillatory.

[1] The choice of filters used to compute the forces in the MSAC transform framework need not necessarily be restricted to strictly lowpass filters.

Further, when topology preservation is an issue as in Fig. 6.2(a), we have to combine this with a good initialization and topology preserving procedure. Unlike the claim in [15], it turns out that the digital topology-based constraint cannot be easily combined with MSAC Transform. This is because the simple-point criterion is tested only in a 3×3 neighborhood around every point on the current contour, whereas the jumps in each iteration of MSAC transform could be of the order of hundreds of pixels. Thus, a point which is tested to be simple at an iteration could actually be crucial when an entire region is changing sign. However, one could explore other criteria based on thinning [120]. Any topology preserving module can be expected to increase the run time of the algorithm.

6.2 MSAC Transformation of fMRI Brain Images

This algorithm is, like its predecessor, extendable to other modalities. As a proof of concept we briefly describe in this section the use of MSAC Transform to segment brain MRI images.

6.2.1 Dataset

The data comprises a brain MRI of 600 2D images, roughly 200 in each sectioning plane provided by the VistaLab [11]. We are also provided an intermediate segmentation result obtained using mrGray [130]. This does not allow us to rigorously quantify the performance of Voting-based MSAC Transform (VBMSAC-T).

6.2.2 Algorithm

We rewrite the VBSTACS algorithm in terms of VBMSAC-T. In essence, the initialization is exactly the same as discussed in Section 5.3.

The evolution of ϕ is based on the region-based force and computed in each sectioning plane as

$$F_r = \sum_{\mathrm{r}=1}^{2} (M_{\mathrm{r}} - \hat{M}_{\mathrm{r}}); \; M_{\mathrm{r}} = \frac{\int_{\mathbb{R}^d} f_{\mathrm{r}}(x) g(a^{-1}(x-y))\,\mathrm{d}y}{\int_{\mathbb{R}^d} C_{\mathrm{r}}(x) g(a^{-1}(x-y))\,\mathrm{d}y + \epsilon},$$

where d is the dimension of image f, g is a lowpass filter with scale parameter a, M_{r} is the mean intensity of region r, the region inside (or outside) the current contour, $\hat{M}_{\mathrm{r}}$ is the

model mean intensity, f_{r} is the portion of the image contained in r, C_{r} is the binary mask of r and $\epsilon > 0$ is a small correction factor to avoid dividing by zero.

As F_r is cast as a convolution, we can use a FFT to expedite the computation. The level-set function at iteration i is evolved as $\phi^{(i+1)} = \phi^{(i)} + \lambda_r^{(i)} F_r^{(i)}$.

We use the same voting procedure as in VBSTACS to determine white and nonwhite regions in a given brain MR image.

6.2.3 Results

Applying F_r at a fixed scale a, we were able to improve the runtime by an order of magnitude, without any degradation in the segmentation quality. The voting-based segmentation using MSAC transform completed in roughly 1 hour on the entire volume of nearly 600 2D images versus the 12 hours taken by the VBSTACS for each sectioning plane [16].

VBMSAC-T is more suited to applications such as brain MR image segmentation rather than fluorescence microscope cell image segmentation as topology preservation is not an issue and in fact, merging and splitting of contours based on image topology is advantageous.

Note that although there is a provision of using multiple scales to evolve the contour, we have only presented the algorithm and results for a fixed value of the scale. We can combine this algorithm with the multiresolution perspective, that is, smoothing followed by down-sampling, and iteratively refine the contour to further increase the algorithm's efficiency. We can also use multiscale edge detection to improve the algorithm's efficacy [142].

6.3 Towards Active Masks

The previous discussion raises several issues to be solved:

- From (6.2.2), the region-based force is only zero when the computed statistics match the model statistics. As this is rarely the case, especially in noisy images, a stopping criterion such as the number of iterations or the extent of change in the masks between iterations needs to be used to halt the procedure. Moreover, setting the iterations *a priori* would involve some amount of runtime tuning to assess the optimal number to be chosen.

- The topology preservation scheme used in TPSTACS was based on the topological number computed in a 3×3 neighborhood of each point on the contour. In the MSAC transform, the contour evolves in increments of hundreds of pixels. Thus, the topological numbers and hence their implications are meaningless, eliminating the topology preservation property. Therefore, the MSAC transform proved to be a useful approach to problems that did not require topology preservation (see Fig. 6.2(b)), but it could not be used without further modification for those that did (see Fig. 6.2(a)).

- Most of the segmentation literature in connection with active-contour methods is phrased in terms of finding a contour. As segmentation is usually performed to choose the object of interest, that is, the contour and all that is inside it, given a discrete grid, it is not immediately clear what the contour is.

In summary, our revised goal is not only to retain the advantages of the MSAC transform of blurring the images to efficiently extract features, having the provision of applying the method at multiple scales and not being critically dependent on the initialization, but also to incorporate topology preservation as a force, incorporate MR analysis to mimic the action of CCD cameras of the microscope, have the algorithm come to a natural stop and have the ability of generalizing the algorithm easily to segment higher-dimensional data. These design criteria inspired the design of the AM algorithm which we describe next.

Chapter 7

Active-Mask Framework

Level-set-based active contours provide a well-performing and mathematically rigorous method for image segmentation. In particular, the active-contour framework provides us with the flexibility to combine specific forces, as in (3.3), to cause the algorithm to simultaneously seek several desired goals. For example, a force may cause the contour to seek a certain smoothness, line up with the inherent edges present within the image, press for a given texture to be enclosed by the contour, or move towards the final contour of a given shape.

As described previously, we encountered several problems in applying active-contour methods to fluorescence microscope data. As no slight modifications to the active-contour theory proved effective, we turned to the development of new segmentation algorithms that nevertheless possessed the flexible, force-based framework of active contours. *(1)* In particular, whereas active contours segment an image by evolving a curve, we segment by evolving whole regions. That is, we formulate our algorithm in terms of *masks*—characteristic functions telling us whether or not a given pixel lies within a given segmented region. Multiple masks may be used to represent multiple regions within an image. A mask is different from a contour, in that a contour gives only the boundary of such a region. *(2)* While forces are used to evolve a curve to obtain a desired contour, we design functions to distribute various regions in the image to unique masks, and call these functions *distributing functions*.

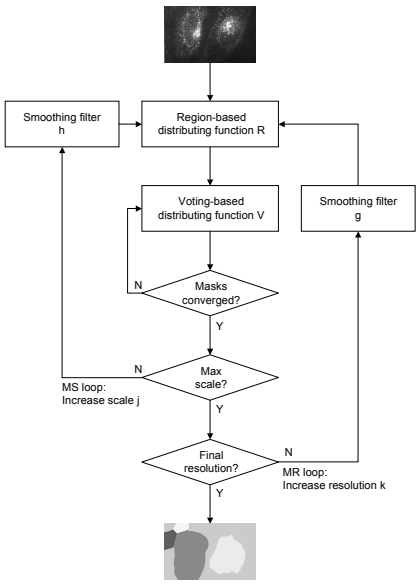


Figure 7.1: Block diagram of the AM algorithm.

Just like specific forces in the active-contour framework, we can design specific distributing functions to cause the algorithm to simultaneously seek several desired goals. Since it is the masks that evolve, we call the framework the *active-mask (AM)* framework.

A block diagram of the AM algorithm is shown in Fig. 7.1 and the pseudo code in Algorithm 7. Given an input image, multiple masks are evolved at a resolution k and scale j, based on a region-based distributing function R and a majority voting-based distributing function V that are applied repeatedly until the configuration of masks ceases to change[1]. The output of the algorithm is a set of masks, each of which represents a segment of the original image. This section, which describes each of these steps, is organized as follows. In Section 7.1, we describe the advantage of using masks instead of contours and show how they can be computed. In Section 7.2, we motivate the need for multiple masks and explain how we compute them. In Section 7.3, we describe the design of filters used in the AM algorithm. In Section 7.4, we describe the AM algorithm and, in particular, the two distributing functions we use to evolve the masks to segment fluorescence microscope

[1]While iterative voting is at the crux of the AM framework, the voting-based distributing function V is but an example for the type of function that can be used. Likewise, there can be many different other functions used instead of or in addition to the region-based distributing function R to skew the voting.

cell images. Finally, in Section 7.5, we describe the full-blown, MR-MS version of the AM algorithm.

We begin with how the apparently minor change in perspective from working with contours to working with masks helped solve some of the problems traditionally associated with active contours.

7.1 Masks Versus Contours

One problem with the level-set formulation is that it does not easily generalize to digital images. For a discrete-variable level-set function $\phi : \mathbb{Z}^d \to \mathbb{R}$, the zero level-set (contour) may be empty. To rectify this problem, one usually keeps track of where ϕ changes sign, that is, finds the *sublevel set* $\{n \in \mathbb{Z}^d \big| \phi(n) \geq 0\}$. This is essentially the approach we shall adopt in our AM framework. Given a d-dimensional image f, our goal is to find a binary-valued image ψ of identical size, where $\psi(n) = 1$ implies that n lies inside the contour while $\psi(n) = 0$ implies the opposite. In short, ψ will be the truth table for whether or not a given pixel lies within the corresponding segmented region—*mask*.

In addition to simplifying one's perspective over discrete domains, such a mask-based formulation easily permits the implementation of MR schemes to speed the algorithm's performance. Our choice of a filter—Haar—in such a MR scheme was guided by two principles: simplicity and the ability to model the signal readout behavior of the CCD camera of a fluorescence microscope [143]. Let H be the Haar averaging filter over $\mathbb{Z}^d$, that is,

$$(Hf)(n) = 2^{-d} \sum_{m \in [0,1]^d} f(2n + m). \tag{7.1}$$

The result of the above operation is a smoothed, downsampled version of the original image f. For example, for $d = 1$, the output Hf is half the size of f and is given by $Hf(n) = (f(2n) + f(2n+1))/2$. Thus, we consider a MR segmentation algorithm in which we first segment a coarse version of f, namely $H^K f$ (H applied to f K times), which is smaller than f by a factor of 2^K in each dimension. Once our segmentation algorithm, described in detail below, has iteratively produced a segmentation mask $\psi^{(K)}$ for $H^K f$, we

use this information as a starting point for the segmentation of the slightly more detailed image $H^{K-1}f$. The algorithm for $H^{K-1}f$ will begin with the *lifted* version of the coarsest mask $\psi^{(K)}$. The lifted version $\psi^{(K-1)}$ is obtained by copying each binary value of $\psi^{(K)}$ into 2^d values of ψ^{K-1}:

$$\psi^{(K-1)}(n) = \psi^{(K)}(\lfloor 2^{-1}n \rfloor), \tag{7.2}$$

where the flooring operation is performed coordinatewise. For example, for $d = 1$ and if $\psi^{(K)} = 1101$, $\psi^{(K-1)} = 11110011$. Here, a contour-based implementation of this same idea would be unnecessarily complicated: one would need to first determine the inside of a given contour to find a mask, then apply (7.2) to this mask to obtain a new one of twice its size, and finally take the boundary of this resulting mask to find the new contour. In short, MR segmentation lends itself more naturally to mask-based, rather than contour-based, segmentation algorithms.

This mask-based approach also retains a classical advantage of level-set methods versus snakes when dealing with high-dimensional datasets. For example, for a three-dimensional image f, a corresponding mask ψ will often provide a simpler and more-easily manipulated description of the segmented region when compared to a complicated two-variable parametrization of its boundary.

We now turn to a more subtle issue, namely our use of binary-valued masks ψ in which the segmented region such as a cell in f, is defined at ψ^{-1}, the preimage of $\{1\}$, as opposed to the traditional, continuously-valued level-set functions ϕ in which the segmented region is defined as $\phi^{-1}[0, \infty)$. This change was inspired by the traditional level-set theory's propensity to adapt gracefully to topological changes, an advantage as contours of multiple objects can be represented using a single level-set function, but a drawback as it likely merges distinct cells into a single blob. Using an external constraint, for example based on discrete topology [119], has some drawbacks such as undesirable abruptness in the contour [132]. Thus, to adapt to topology in a meaningful way, our solution was to permit multiple masks. That is, use an M-valued ψ to represent a collection of M binary-valued masks $\{\chi_m\}_{m=1}^{M}$ corresponding to M regions in f. We now explain multiple masks in more detail, beginning

with the relevant details of the level set formulation that inspired the change.

7.2 Multiple Masks

In the discrete domain level-set formulation of active contours, the set $\{n \in \mathbb{Z}^d \big| \phi(n) \geq 0\}$ is the region inside of the contour, while $\{n \in \mathbb{Z}^d \big| \phi(n) < 0\}$ is the region outside. Unfortunately, such a formulation forces one to only have two segmentation regions: in and out. If the exact number of contours is known beforehand, then the level-set function can be suitably initialized and an external constraint used to preserve topology. However, when the exact number of contours is not known and no external constraint is used, the definition of just two regions causes problems of unwanted merges. Thus, whenever the regions to be segmented are close to each other (see an example of cells in fluorescence microscope images in Fig. 7.2(a)), multiple level-set functions have been used [144, 145]. By using L level-set functions $\{\phi_k\}_{k=1}^{L}$, the domain is divided into 2^L distinct regions, corresponding to the possible binary sequences generated as the signs of the ϕ_k's. In our proposed method, in addition to allowing an arbitrary number of segmented regions, we will have the ability to completely disregard masks that are superseded by others, thereby reducing the computational load.

We encountered three major problems when applying level set-based segmentation methods to fluorescence microscope images. *(1)* To correctly segment a simple image consisting of two adjacent cells, the level-set function ϕ would necessarily be positive inside both cells, and would need to dive to be negative in a thin region between them. Such thin, delicate regions of sign change are susceptible to typical errors that arise in the iteration of the level-set function, making them numerically unstable and sensitive to noise. *(2)* In reality, there is often no physical space between the cells: they are adjacent, and a proper segmentation should discern which pixels lie within the first cell, which within the second, and which outside of either. *(3)* Even when the algorithm performs successfully and no unwanted merges occur, one must nevertheless go through the simple, but tedious post-processing step of separating the set $\{n \in \mathbb{Z}^d \big| \phi(n) \geq 0\}$ into maximally connected subsets, each of which represents either a cell or a component of the background. All of these problems may

be overcome by using multiple masks.

Ideally, we search for an algorithm that iteratively adjusts multiple masks and ultimately results in each cell being perfectly covered by a single mask, with a final mask for the background. To simplify the mathematical expression of the algorithm that follows, we consider the d-dimensional digital image as a function $f \in \ell^2(\mathbb{Z}^d)$ that has the property that $f(n) = 0$ whenever at least one coordinate n_k of the multi-integer $n = (n_1, \ldots, n_d)$ lies outside of the interval $[0, N_k)$. That is, f is a member of:

$$\ell^2(\Omega) \;=\; \big\{ f : \mathbb{Z}^d \to \mathbb{C} \;\big|\; f(n) = 0 \text{ for all } n \notin \Omega \big\}, \tag{7.3}$$

where $\Omega = \prod_{k=1}^{d} [0, N_k)$ has the characteristic function:

$$\chi_\Omega(n) = \begin{cases} 1, & 0 \le n_k < N_k, \quad k = 1, \ldots, d; \\ 0, & \text{otherwise.} \end{cases}$$

Letting M be a positive integer, a collection of M masks is a function ψ that assigns each pixel $n \in \Omega$ a value $\psi(n) \in \{1, \ldots, M\}$. Here, n is an element of the mth mask if $\psi(n) = m$, while outside of Ω, ψ is zero. To summarize, a collection of masks is an element of:

$$\mathcal{S}_M(\Omega) = \big\{ \psi : \mathbb{Z}^d \to \{0, \ldots, M\} \big| f(n) = 0 \Leftrightarrow n \notin \Omega \big\}, \tag{7.4}$$

where the masks themselves are binary-valued characteristic functions derived from ψ (see (7.7)). The segmented regions are $\psi^{-1}\{m\} = \{n \in \mathbb{Z}^d \big| \psi(n) = m\}$, $m = 1, \ldots, M$ and $\psi^{-1}\{m\}$ could denote, for example, a cell in f. Below, we discuss how active-contour-inspired forces may be designed to iteratively refine ψ so as to ultimately produce an M-channel segmentation (each channel represents a region) of the original image. Here, the role of the smoothness force will be recast in terms of a local majority voting that decides to which mask a voxel belongs taking into account the masks to which voxels in its local neighborhood belong. The equivalent of other forces will play a role in skewing this voting. As the role of the local majority voting as well as the functions used to skew the voting

is that of distributing the masks to various regions in the image, we call them *distributing functions*. Thus, distributing functions in the AM framework are the counterparts of forces in the active-contour or level-set paradigm.

7.3 Local Averaging

A distributing function at the crux of our AM segmentation algorithm draws on the idea of majority voting based on local averages. For any function $f \in \ell^2(\mathbb{Z}^d)$ and any given pixel $n \in \mathbb{Z}^d$, we will need to compute weighted averages of the values $f(m)$ over all m sufficiently close to n. The weights themselves are taken to be the values of some nonnegative lowpass filter $g \in \ell^1(\mathbb{Z}^d)$. For example, one may take g to be a truncated version of $\exp(-b^{-2}|n|^2)$ for some scale parameter $b > 0$. The local averages of f with respect to the coefficients in g may be given in terms of the convolution:

$$(f * g)(n) = \sum_{m \in \mathbb{Z}^d} f(n-m)g(m). \tag{7.5}$$

One type of local average of f may be obtained by dividing (7.5) by its DC value $\sum_{n \in \mathbb{Z}^d} g(n)$. However, near the boundary of the image, such averages take into account many values of the image that were artificially defined to be zero, and as such, are disproportionately small. This issue may be overcome by symmetrically extending the image across the boundary. An alternative fix we use, is to compute the averages over only the portion of f that lies within $\Omega = \prod_{k=1}^{d}[0, N_k)$. In particular, we compute our local averages using the nonstandard, noncommutative convolution:

$$(f \star g)(n) = \begin{cases} \dfrac{(f * g)(n)}{(\chi_\Omega * g)(n)}, & n \in \Omega; \\ \\ 0, & n \notin \Omega. \end{cases} \tag{7.6}$$

If $f \in \ell^2(\Omega)$, then $f \star g \in \ell^2(\Omega)$ is well-defined for any nonnegative, nonzero $g \in \ell^2(\mathbb{Z}^d)$. In practice, we take g to be finite tap, and compute both the numerator and denominator of (7.6) by FFT-implemented circular convolutions of zero-padded versions of f, g and χ_Ω.

In so doing, the segmentation of even large images may be computed with relative efficiency.

7.4 The AM Framework

The crux of the AM framework is based on the voting-based distributing function (see Fig. 7.1 and Algorithm 7). At a given resolution and scale, it is iterated until equilibrium is achieved, at which point, the scale is increased, leading to an another iterating process. Once the scale has converged, the resolutions is increased and the whole process is repeated once more. Thus, the basic step of the AM algorithm is as follows: Given a collection of multiple masks at iteration η, $\psi_\eta \in \mathcal{S}_M(\Omega)$ as defined in (7.4), we begin by forming the masks themselves as characteristic functions of the collection, as:

$$\chi_{\psi_\eta^{-1}\{m\}}(n) = \begin{cases} 1, & \psi_\eta(n) = m; \\ 0, & \text{otherwise}, \end{cases} \tag{7.7}$$

for every $m = 1, \ldots, M$. As $\left\{\psi_\eta^{-1}\{m\}\right\}_{m=1}^{M}$ partitions Ω, its characteristic functions (7.7) satisfy:

$$\chi_\Omega = \sum_{m=1}^{M} \chi_{\psi_\eta^{-1}\{m\}}. \tag{7.8}$$

Next, we allow each mask to spread its influence by passing its characteristic function through some fixed nonnegative lowpass filter g, performing d-dimensional FFT-based non-standard convolutions $\chi_{\psi_\eta^{-1}\{m\}} \star g$, as defined in (7.6). Because of (7.8), the values of these convolutions sum to 1:

$$\begin{aligned} &\textstyle\sum_{m=1}^{M} (\chi_{\psi_\eta^{-1}\{m\}} \star g)(n) \\ &= \tfrac{1}{(\chi_\Omega * g)(n)} \textstyle\sum_{m=1}^{M} (\chi_{\psi_\eta^{-1}\{m\}} * g)(n) \\ &= \tfrac{1}{(\chi_\Omega * g)(n)} \Big(\textstyle\sum_{m=1}^{M} \chi_{\psi_\eta^{-1}\{m\}} * g\Big)(n) \\ &= \tfrac{(\chi_\Omega * g)(n)}{(\chi_\Omega * g)(n)} \qquad = 1. \end{aligned} \tag{7.9}$$

For any $n \in \Omega$, the value

$$V_{m,\eta}(n) = (\chi_{\psi_\eta^{-1}\{m\}} \star g)(n), \tag{7.10}$$

is the voting-based distributing function and it represents the degree to which n wants to belong to mask m, based upon the influence of its neighboring pixels at iteration η. In other words, that m producing the largest value of $V_{m,\eta}(n)$ represents the new mask to which pixel n will belong.

For example, consider the ηth iteration of a 2D image with the filter g being 1 on a 3×3 square centered at the origin and is otherwise 0. Further, suppose that we have $M = 3$ masks and that the values of ψ_η in a 3×3 neighborhood of some pixel $n = (1, 1)$ are:

$$\begin{array}{|c|c|c|} \hline \psi_\eta(0,0) & \psi_\eta(0,1) & \psi_\eta(0,2) \\ \hline \psi_\eta(1,0) & \psi_\eta(1,1) & \psi_\eta(1,2) \\ \hline \psi_\eta(2,0) & \psi_\eta(2,1) & \psi_\eta(2,2) \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 2 & 2 & 2 \\ \hline 2 & 1 & 3 \\ \hline 2 & 1 & 1 \\ \hline \end{array}.$$

Computing $V_{m,\eta}(1, 1)$ for $m = 1$, 2 and 3, yields 3/9, 5/9 and 1/9, respectively, implying that in the next iteration, the pixel $(1, 1)$ will want to change its membership from mask 1 to mask 2.

Indeed, for any d, g and M, if we iteratively define new masks according to this local majority voting scheme:

$$\psi_{\eta+1}(n) = \underset{m=1,\ldots,M}{\operatorname{argmax}} (V_{m,\eta})(n), \tag{7.11}$$

we have essentially pitted the masks against each other; each mask tries to conquer any neighboring pixels, and may only be stopped by another mask attempting to do the same. Experiments reveal that iterating (7.11) will eventually result in masks in equilibrium (that is, further iteration of (7.11) cause no further change). When the masks are in equilibrium, we say the iteration has *converged*. The minimum thickness of these masks at equilibrium seems to be mostly dependent on the size of the support of the filter g. Similarly, the smoothness of the boundaries between distinct masks at equilibrium seems to depend greatly on the smoothness of the filter g. If we are looking for smooth boundaries, we should

avoid the blocky g described in the example above, and instead choose a g weighted in a more isotropic manner. In this sense, (7.11) is the AM version of the smoothing forces traditionally associated with active-contour algorithms.

However, iteratively applying (7.11) will not successfully segment the image in question, as (7.11) does not take the image itself into account. As with active contours, one needs to combine this smoothness requirement with other requirements to push the segmentation in the right direction according to the features of the underlying image. In the AM framework, we regard these as functions which skew the majority voting scheme. The AM iteration is formally defined to be:

$$\psi_{\eta+1}(n) = \underset{m=1,\ldots,M}{\operatorname{argmax}} \big[V_{m,\eta}(n) + R_m(n)\big], \tag{7.12}$$

for $n \in \Omega$ with $\psi_{\eta+1}(n) = 0$ otherwise, where the functions $\{R_m\}_{m=1}^{M} \subseteq \ell^2(\Omega)$ are fixed functions which depend upon the image alone. In general, these R_m's may take into account edges, textures, and morphological features present in the image. Because of (7.9), one should scale the R_m's to not exceed 1 in magnitude, as then, their value will completely override the smoothing action of the majority voting. Note that other R's are possible (as well as more than one).

For the purpose of segmentation of fluorescence microscope images, we took the R_m's in (7.12) to depend purely on the density of the underlying pixels. For this particular class of images, pixels inside a cell are fundamentally distinguished from those outside by the average intensity of the pixels in their vicinity. Having arbitrarily decided to always let the first mask attempt to represent the background, with the remaining $(M-1)$ masks attempting to each realize an individual cell, we therefore constructed a function R_1 which would skew the voting of points of lower image density towards the first mask, and those of higher image density away from it. In practice, we constructed R_1 via a soft-thresholding of a smoothed version of the original image f (see Fig. 7.1 and Algorithm 7). Letting h be some lowpass filter:

$$R_1(n) = \alpha \operatorname{sig}\Big(\beta\big((f \star h)(n) - \gamma\big)\Big), \tag{7.13}$$

where $\alpha, \beta, \gamma \in \mathbb{R}$ were experimentally determined parameters and sig : $\mathbb{R} \to \mathbb{R}$ is any sigmoid-type function which asymptotically achieves the values ± 1 at $\pm\infty$, respectively. Here, γ should be taken to be the average intensity of those pixels which lie on the boundary between being inside every cell and being outside them all. Meanwhile, β determines the harshness of the threshold, while α should be taken in $(-1, 0)$ and close to -1. Under these conventions, for a typical pixel n inside a cell, we have $(f \star h)(n) > \gamma$, and so $R_1(n) \approx \alpha$, which, as $\alpha < 0$, will skew the voting in (7.12) so as to prefer any mask besides the first. Similarly, when n lies outside all cells, we have $R_1(n) \approx -\alpha$, and so the voting in (7.12) will favor the first mask. As smoothness alone is sufficient to distinguish the cells from each other, we took $R_m(n) = 0$ for all $m \neq 1$ and all $n \in \mathbb{Z}^d$.

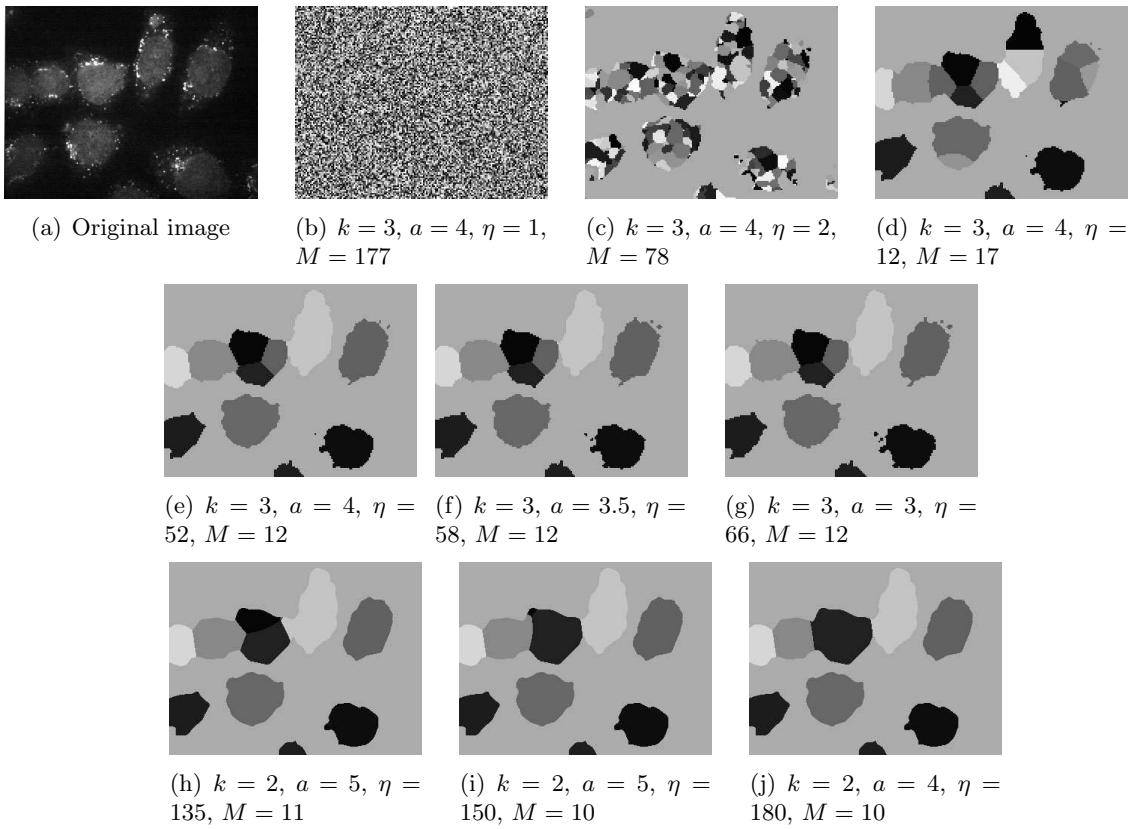
As we will see in Chapter 8, under this definition of the R_m's, repeated applications of (7.12) seem to converge to a final segmentation in a reasonable amount of time, even when the initial segmentation ψ_0 is taken to be random. This is possible because, even in the first iteration, the strong preference that R_1 exhibits for the outside will quickly distinguish foreground from background. With the inside versus outside question settled, the remaining masks compete for supremacy, in a manner similar to the unbiased voting (7.11). Inevitably, larger masks will consume smaller ones; if $\psi_\eta(n) \neq m_0$ for all $n \in \mathbb{Z}^d$, then $\psi_{\eta+1}(n) \neq m_0$ for all $n \in \mathbb{Z}^d$. That is, if a given mask is no longer present at a given iteration, it is gone forever, and may be treated as such. That is, one may effectively regard the number of masks M as getting smaller, reducing the number of convolutions one must compute in (7.12). The successive reduction in the number of masks as the iterations converge is illustrated in Fig. 7.2. Fig. 7.2(a) shows a fluorescence microscope image of HeLa cells with eight distinct (and six whole) cells. Fig. 7.2(b) shows the initial state with $M = 177$ masks. In the first iteration, R coarsely separates the background from the foreground. Thus, in the very next iteration, $\eta = 2$, the number of masks is drastically reduced and the empty masks are discarded to yield $M = 78$ as shown in Fig. 7.2(c). Subsequently, the masks in the foreground region fight amongst themselves, with the geometry of the cells prevailing. Thus, in Fig. 7.2(d), at $\eta = 12$, we see the masks showing an increased correspondence to

the cells and the number of masks is steadily decreasing (in this case, $M = 17$).

Ultimately, the iterations will converge when the remaining masks that correspond to the cells achieve an equilibrium. This will typically happen when the inherent geometry of the cells causes a stalemate. In essence, when two round cells touch, a separate mask will grow to dominate each cell. The boundary between the two masks will coincide with the boundary between the two cells, as the narrowing of the cells near their boundary creates a narrow pass which neither side is able to conquer. Thus, the repeated application of (7.12) will often result in a first mask which contains the background, while the remaining nonzero masks each describe a nonzero cell. However, as noted in Sections 3.3 and in Chapter 4, no algorithm is perfect and splits and merges of cells will occasionally occur. In general, these anomalies depend on the design of the distributing functions and how sensitive the filters are to the features that distinguish the objects of interest. In particular, for the segmentation of fluorescence microscope cell images with the distributing functions described above, if the region-based distributing function has not been smoothed adequately, the masks during the local majority voting may split a single cell into two or more regions based on kinks on the surface of single cell. Alternatively, if the scale parameter for the voting-based distributing function is too small, the function may be too sensitive to small indentations in the foreground, causing spurious splits. Likewise, if the scale parameters of the filters are too large, cusps between touching cells may be smoothed out, resulting in spurious merges. As the scale parameter plays an important role, we try to overcome the problem of spurious merges and splits automatically by applying the distributing functions at multiple scales.

7.5 The MS-MR-AM Algorithm

We now discuss a MR-MS version of (7.13) in more detail. Here *MR* refers to the fact that we first segment a downsampled version of the image as in (7.1), and then lift this segmentation to the next highest resolution as described in (7.2) for the sake of speed [15,80]. Meanwhile, *MS* refers to the fact that at a given resolution, we may very slowly change the scale parameter $a > 0$ which scales the lowpass filter h from which the region-based distributing



(a) Original image

(b) $k = 3$, $a = 4$, $\eta = 1$, $M = 177$

(c) $k = 3$, $a = 4$, $\eta = 2$, $M = 78$

(d) $k = 3$, $a = 4$, $\eta = 12$, $M = 17$

(e) $k = 3$, $a = 4$, $\eta = 52$, $M = 12$

(f) $k = 3$, $a = 3.5$, $\eta = 58$, $M = 12$

(g) $k = 3$, $a = 3$, $\eta = 66$, $M = 12$

(h) $k = 2$, $a = 5$, $\eta = 135$, $M = 11$

(i) $k = 2$, $a = 5$, $\eta = 150$, $M = 10$

(j) $k = 2$, $a = 4$, $\eta = 180$, $M = 10$

Figure 7.2: An illustration of the evolution of AM with random seeds [17]. Note that unlike in pseudocode Algorithm 7, here we report iteration number η cumulatively. (a) A COPII image (enhanced). (b) In the first iteration $\eta = 1$, with an initial number of masks, $M = 177$, decomposition level, $k = 3$ and scale of the smoothing filter, $a = 4$. (c) At $\eta = 2$ and $k = 3$, the foreground is separated coarsely from the background, with $M = 78$ and $a = 4$. (d) At $\eta = 12$, $k = 3$ and $a = 4$, $M = 17$. (e) At $\eta = 52$, $k = 3$ and $a = 4$, $M = 12$. As zero pixels change at this stage, the scale parameter is annealed, at the same resolution, $k = 3$. (f) At $\eta = 58$, $k = 3$ and $a = 3.5$, $M = 12$. As zero pixels change, the scale parameter is further annealed. (g) At $\eta = 66$, $k = 3$ and $a = 3$, $M = 12$. As zero pixels change at this stage, the resolution is pulled back to $k - 1$. (h) At $\eta = 135$, $k = 2$ and $a = 5$, $M = 11$. (i) At $\eta = 150$, $k = 2$ and $a = 5$, $M = 10$. (j) At $\eta = 180$, $k = 2$ and $a = 5$, $M = 10$. Zero pixels change at this stage. We may anneal the scale parameter and pull the resolution back to the original resolution; however, we note that a satisfactory segmentation has already been achieved at this stage. It takes $\approx$ 1.7min, including writing the resulting mask at each iteration, to evaluate. The exact time taken each time the algorithm is rerun depends on the initial random configuration. Note: Images at different resolutions have been scaled to the same size for display purposes. This final result is shown in pseudo color in Fig. 8.5(a). (Original image courtesy of Dr. A. D. Linstedt [5].)

function (7.13) is computed. For example, with

$$h(n) = \exp(-a^{-2}|n|^2), \tag{7.14}$$

experiments revealed that choosing a to be large results in a quickly converging algorithm whose ultimate masks nevertheless overestimate the size of the cells in addition to over-smoothing their boundaries. While choosing a to be small yields better masks, it slows convergence. For the best of both worlds, we took a simulated annealing approach.

Specifically, a is initially taken to be large to rapidly obtain a coarse segmentation. Then, after (7.12) has converged for this specific a, these masks form the initial guess for the rerunning of (7.12) with a slightly smaller a. When the change in a is slight, the second iteration will converge in a few steps. This process is then repeated, gradually pulling a down to a point where experimentation has revealed the ultimate segmentation as being a good match to the ground truth. This annealing may occur at any resolution, that is, we pass through a decreasing sequence of scale factors $\{a_{k,j}\}_{j=1}^{J_k}$ for any fixed resolution k and downsampling $H^k f$ of the original image f. The complete MS, MR AM algorithm, beginning with some randomly chosen $\psi_1^{(K,1)} \in \mathcal{S}_M(\Omega)$, is shown in Fig. 7.1 and Algorithm 7, with index k referring to the MR loop, and index j referring to the MS loop.

The evolutionary behavior of AM algorithm is illustrated in Fig. 7.2. For any fixed k and j, we iteratively apply (7.12) until the masks are in equilibrium. We then use these masks as the starting point for the next step in the annealing. When the annealing for a given resolution is completed, we lift the resulting masks up to a higher resolution and begin the process again. The final masks are given by the function $\psi_{\eta_{\max}}^{(K_0,J_{K_0})}$. When $K_0 = 0$, the masks given by the function $\psi_{\eta_{\max}}^{(0,J_0)}$ provide a full-resolution, multiscale-annealed multiple-mask segmentation of the original image. The scale b of the filter g used in the voting-based distributing function, chosen to be a fixed constant here, can also be annealed. The annealing can either be explicit, as it is for a, or implicit, as merely some function of resolution k. The smaller the scale b, the longer it takes for $\psi^(k, j_k)$ to converge, but the boundaries between the foreground masks are more accurate. The parameters α, β

and γ are fixed constants which are experimentally determined for optimal performance. Experimental results detailing the performance of this algorithm, both in terms of speed and accuracy, are given in the following section. We are currently mathematically investigating the convergence of the algorithm; some of these are outlined in the following section.

7.6 Convergence Issues

AM achieves experimental convergence, that is, the evolving masks actually arrive at a *zero change* state, at which point the procedure comes to a halt. This section describes briefly some of the approaches we have taken in investigating the theoretical convergence of the distributing functions described above in the AM framework.

Objective function. Unlike forces in the active-contour framework, we do not formulate distributing functions in the AM framework as energy functionals or the problem of segmentation as a minimization problem. Hence, we do not have an objective function as such whose properties can be studied/are known. As AM achieves experimental convergence for a lowpass filter under the conditions mentioned in Section 8.2, we start with the premise there is an underlying objective function that is being optimized. As a first step in identifying such an objective function, we computed the change in some of the boundary properties. If the change of a property (or a combination of properties) is monotonic, it would act as our lighthouse in proving the procedure's convergence. The properties we have investigated so far include: total curvature, mean curvature, maximum curvature, length of the boundary, area of the *blobs*, and various combinations of these properties. However, experiments have revealed that none of these is monotonic.

Is AM a fixed-point algorithm? For the sake of simplicity, we consider the number of masks to be $M = 2$. Without loss of generality, we suppose that one of the masks is -1 and the other $+1$ and that $d = 2$. The majority-voting function is replaced by the sign function, $T : \mathbb{L}^2(\Pi^d) \rightarrow \mathbb{L}^2(\Pi^d)$. Further, as the region-based function R serves to skew the voting, we disregard R, in a first attempt to understand the behavior of iterative smoothing and thresholding (in this case, applying T and G)[2] on ψ. In essence, the AM algorithm given

[2] $G\psi = g * \psi$.

by (7.11) is captured in a simple form by $TG\left((TG)^{\eta-1}\psi\right)$. To prove convergence of this procedure, it is sufficient to show that the number of zero-crossings in ψ is monotonically nonincreasing with iteration.

The action of iterative smoothing is well studied as the maximum (or minimum) principle [146] (see [82] for a discrete version of the maximum principle). The conditions under which a lowpass filter produces a coarse version of the image have been presented in the formulation of the diffusion equation [147]. It has been rigorously proved that a version of the maximum principle from the theory of parabolic differential equations is equivalent to the condition of applying such a lowpass filter on an image so as not to produce any new zero-crossings (or edges) in the image [148].

The challenge in applying these ideas to AM, stems from the fact that T is not continuous. Although it continues to be nonlinear, we may approximate T by a sigmoid, $\sigma_\theta(u) = \frac{\exp(\theta u)-1}{\exp(\theta u)+1}$ as $\sigma_\theta \xrightarrow{\theta\to\infty} T$. Thus, for some $x \in \mathbb{R}^d$, we have $T(Gx) \approx \sigma_\theta(Gx)$.

A function Λ has a fixed point x, if $\Lambda(x) = x$ for some x. In particular, let $\Lambda = \sigma_\theta(G)$. If the set of fixed points of all θ is denoted by $\mathfrak{F}_\theta$, we ask, $\mathfrak{F}_\theta \xrightarrow{\theta\to\infty} \mathfrak{F}$? In other words, does the iteration converge?

If $\Lambda^{(\eta)}(x) = \Lambda\left(\Lambda^{(\eta)-1}(x)\right)$, exists, it is exponential. A sufficient condition to prove convergence would be to prove $\|\Lambda(\Lambda(x)) - \Lambda(x)\| \leq c\|\Lambda(x) - x\|$, for some $c < 1$. This condition implies Λ is a contraction, which further implies the existence of a unique fixed point for Λ. However, in the case the Λ we are studying, uniqueness is not desirable because we know that $TG(0) = 0$ is a fixed point. For some $x \neq 0$, we notice convergence happens and the iteration stops. We want to investigate such nontrivial fixed points of Λ. Thus we have to weaken the conditions for unique fixed points to allow for multiple fixed points. As Λ depends on G, it is desirable to describe our objective in terms of properties of G. In other words, what are the conditions on G so that $TG(x)$ is a fixed point?

Most of the fixed point literature deals with conditions for existence of a unique fixed point in the context of investigating the existence of a unique solution to a differential equation. Our difficulty in calling upon the vast literature on fixed point algorithms is that

the conditions for existence of a *unique* fixed point are generally much stronger than the conditions for just the existence of a fixed point [149]. As we do not have an objective to begin with (and coming up with one requires an understanding of the fixed points of Λ), the problem of weakening the conditions to obtain the objective, is circular.

Convergence in 1D. In the following discussion, we define convergence as that state of ψ which remains unaltered on operating TG any number of times on it.

Here, we consider the action of T but simplify the problem to $d = 1$, and restrict the lowpass filter g to a train of finitely many δ's. That is, G is of the form

$$G\psi = \psi * g;\ g = \frac{1}{N} \sum_{n=-\lfloor \frac{N}{2} \rfloor}^{\lfloor \frac{N}{2} \rfloor} \delta_n,$$

where $N \leq \text{length}(\psi)$ and $\mod(N, 2) = 0$ so G has an odd number of δ's. Trivially, when $N = 0$, $TG\psi = \psi$. We will now consider a few nontrivial examples for the case $N = 2$. In these examples, we assume a circular signal extension. In practice, we do not use such a signal extension but rather use a modified convolution as described in (7.5).

$N = 2$	Case 1:	ψ	1	1	1	1	1	
		$TG\psi$	1	1	1	1	1	
	Case 2:	ψ	1	1	-1	-1	-1	
		$TG\psi$	1	1	-1	-1	-1	
	Case 3:	ψ	1	-1	1	-1	1	
		$TG\psi$	1	1	-1	1	1	
		$(TG)^2\psi$	1	1	1	1	1	
	Case 4:	ψ	1	-1	1	-1	1	-1
		$TG\psi$	-1	1	-1	1	-1	1
		$(TG)^2\psi$	1	-1	1	-1	1	-1

Based on the above example, we make a few observations. (*1*) When the ψ contains points of the same sign, $TG\psi$ is trivially ψ. (*2*) When ψ contains more than $N/2$ consecutive

points of the same sign, iterating TG any number of times will not alter their sign. (*3*) When ψ contains more than $N/2$ consecutive points of the same sign that flank $N/2$ (or fewer points) of a different sign, repeated application of TG will eventually alter the sign of the $N/2$ or fewer points to the sign of the *majority* that surrounds them. (*4*) When the length of every subsequence of points in ψ that has the same sign is equal to $N/2$, iterating TG on ψ results in the points in ψ alternating their signs. Studying a few examples with $N = 4$ and $N = 6$ lend further insight and a few more interesting observations.

$N = 4$	Case 1:	ψ	1	1	1	−1	−1	−1	1					
		$TG\psi$	1	1	1	−1	−1	−1	1					
	Case 2:	ψ	1	1	−1	−1	1	1	−1					
		$TG\psi$	1	−1	1	1	−1	1	1					
		$(TG)^2\psi$	1	1	1	1	1	1	1					
	Case 3:	ψ	1	1	−1	−1	1	1	−1	−1				
		$TG\psi$	−1	−1	1	1	−1	−1	1	1				
		$(TG)^2\psi$	1	1	−1	−1	1	1	−1	−1				
	Case 4:	ψ	1	−1	1	−1	1	−1	1	−1				
		$TG\psi$	1	−1	1	−1	1	−1	1	−1				
	Case 5:	ψ	1	−1	1	1	−1	−1	1	−1	1	1	−1	−1
		$TG\psi$	−1	1	1	−1	1	−1	−1	1	1	−1	1	−1
		$(TG)^2$	1	−1	1	1	−1	−1	1	−1	1	1	−1	−1

Cases 1-4 are consistent with the examples considered for $N = 2$. In short, from Case 1, we see that if the length of every subsequence of points in ψ is greater than $N/2$, then iterating TG on ψ produces no change in ψ.

Case 2 lends strength to the observation that if there is at least one subsequence of points of the same sign in ψ of length greater than $N/2$ adjacent to a subsequence of points of a different sign and of length $N/2$ or smaller, then ψ converges to the sign of the *majority*

subsequence in one or more steps. Henceforth we refer to the *majority* subsequence as a *stable region*. Thus, by our understanding, a stable region is a set of sequences of greater than $N/2$ adjacent points whose sign does not change regardless of the number of iterations of TG on ψ. A stable point is a member of the stable region. Further, the length of a sequence of adjacent stable points may remain the same or increase under the action of TG by consuming points from the adjacent *unstable region/s*. Thus, an unstable region is a set of points which changes its sign under the action of TG. A question this raises is: for a given initial sequence of points in ψ, how many iterations of TG on ψ will it take for ψ to converge? We will revisit this question shortly. We note here that the above definitions of stable and unstable regions do not account for subsequences of fewer than $N/2$ adjacent points of the same sign, which may not change their sign under the action of TG (such as in Case 4 above).

Finally, Case 3 and Case 4 seem to combine to make a stronger statement than observation (*4*), which is, when the length of every subsequence of points in ψ that has the same sign is less than or equal to $N/2$, iterating $(TG)^{\eta}, 0 < \eta < \infty$ on ψ either makes no difference or results in the points in ψ alternating their signs[3]. Also, it appears that when the length of every subsequence of points in ψ that has the same sign is less than $N/2$, iterating TG on ψ makes no difference to the values of ψ. However, we will examine a few examples of $N = 6$ and $N = 8$, before making these statements. Case 5 points to an altogether new combination for an alternating sequence (it is easier to see this by focusing on the lengths of the subsequences of points having the same sign). We can expect to more interesting cases as N increases.

[3]For instance, in Case 4, $\eta = 1$ and corresponds to the sequence being invariant and in Case 3, $eta = 2$ and corresponds to the case of alternating signs or periodic invariance (or "convergence with a period 2".)

$N = 6$	Case 1:	ψ	1	−1	1	−1	1	−1	1	−1	1	−1		
		$TG\psi$	−1	1	−1	1	−1	1	−1	1	−1	1		
		$(TG)^2\psi$	1	−1	1	−1	1	−1	1	−1	1	−1		
	Case 2:	ψ	1	1	−1	−1	1	1	−1	−1	1	1	−1	−1
		$TG\psi$	−1	−1	1	1	−1	−1	1	1	−1	−1	1	1
		$(TG)^2\psi$	1	1	−1	−1	1	1	−1	−1	1	1	−1	−1
	Case 3:	ψ	1	1	1	−1	−1	−1	1	1	1	−1	−1	−1
		$TG\psi$	−1	−1	−1	1	1	1	−1	−1	−1	1	1	1
		$(TG)^2\psi$	1	1	1	−1	−1	−1	1	1	1	−1	−1	−1
$N = 8$	Case 1:	ψ	1	−1	1	−1	1	−1	1	−1	1	−1	1	−1
		$TG\psi$	1	−1	1	−1	1	−1	1	−1	1	−1	1	−1
	Case 2:	ψ	1	1	−1	−1	1	1	−1	−1	1	1	−1	−1
		$TG\psi$	1	1	−1	−1	1	1	−1	−1	1	1	−1	−1
	Case 3:	ψ	1	1	1	−1	−1	−1	1	1	1	−1	−1	−1
		$TG\psi$	−1	−1	−1	1	1	1	−1	−1	−1	1	1	1
		$(TG)^2\psi$	1	1	1	−1	−1	−1	1	1	1	−1	−1	−1

Clearly, we see a pattern emerging. We note that for this simple case of $M = 2$ and a point $p \in \mathbb{Z}^+$ such that $\psi(p) \in \{-1, 1\}$ and subject to the action of G, the smoothing operator described above with the length of the filter g being $N+1$ and T, the sign function, some of the previous claims can be captured by the theorems below.

Theorem 1 *Any subsequence*[4] *of more than* $N/2$ *points with the same sign forms a stable region.*

Proof Suppose ψ contains a subsequence of $N/2+1$ points of the same sign. Without loss of generality, let us say these points are $+1$'s and are adjacent to subsequences (on both

[4]Subsequence refers to a set of adjacent points of ψ.

sides) of -1's. Then, by definition, repeated application of the filter should result in the points of this set not changing their signs. Let us center the filter G at any point a of this region of +1's. As more than half of the points within G overlaps with points of the same sign as a, the sign of a stays the same. Consequently, we conclude this region is invariant under the repeated application of TG on ψ. Therefore this sequence satisfies the definition of a stable region. ∎

Theorem 2 *In any ψ that has at least a stable region, the iteration will converge and any unstable regions in the initial ψ will be eliminated.*

Proof Suppose ψ contains a subsequence of adjacent points of the same sign that forms a stable region. Without loss of generality we may assume the points in the stable region are +1's. Let us further assume that the subsequence shares a border with an unstable region. Thus, at the border, point b of the stable region has the value +1. b has at least $N/2$ (+1)'s to one side of it and a point a with the value -1 to its other side with no more than $N/2-1$ (-1)'s adjacent to a. (These assignments follow from the previous theorem.) Now, if we center our filter at this a, there happen to be $N/2$ (+1)'s to one side that forms the stable region and at least one +1 in the $N/2$ points that are on the other side of a. This results in the sign of a being changed from -1 to +1. Thus, the stable region of +1's grows by a point. Extending the argument along these lines, we end up with the consequence stable sequence grows till there remain no more unstable regions in ψ. ∎

Theorem 3 *Assume ψ is composed only of alternating subsequences of length $L \leq \frac{N}{2}$ with as many points of one sign in ψ as there are points of the other sign. ψ is invariant under TG or it transforms into its negative. The effect of TG is the same as $T\hat{G}$ where $\hat{G}$ corresponds to the filter of length $\hat{N}/2 \equiv \frac{N}{2} \mathrm{mod} 2L$. TG has the opposite effect of $T\hat{G}$ on ψ, if $\hat{N} - N$ is an odd multiple of $2L$.*

Proof As the general proof is very cumbersome, we will first discuss the case for $L = 2, N = 4$ and $\hat{N} = 6$ or 8. In the table above for $N = 4$, ψ in Case 3 corresponds to $L = 2$. Let us center the filter G on any point a in ψ. Clearly, there are as many points of one sign

as the other sign adjacent to a. These equal number of adjacent points of opposite sign cancel out the effect of each other. Thus, the only unbalanced point is a and hence the sign of a stays the same. Let us now consider a filter $\hat{G}$ with length $\hat{N}/2 = 8$ and center the filter $\hat{G}$ at any arbitrary point a in ψ. We notice that this filter considers a pair of points on either side of a in addition to the points considered by G. Any point in a subsequence of points of the same sign in ψ repeats after every (multiple of) four points. Consequently, adding four points to G adds two points of both signs. They cancel out the effect of each other and as before, the sign of a remains unaltered. Let us now consider a filter $\hat{G}$ with length $\hat{N}/2 = 6$ and center the filter $\hat{G}$ at any arbitrary point a in ψ. We thus consider only two points (or a length equal to half a period) in addition to the points considered by G. In effect, $\hat{G}$ overlaps with only two additional points of the same sign as a whereas it overlaps with four points of the opposite sign. This causes the sign of a to change. The example is illustrated in the table of examples for N=6 as Case 2. This argument can be generalized for all $L < \infty$ as follows.

Suppose ψ consists of alternating subsequences of length $L \leq \frac{N}{2}$ and is invariant under the action of TG. This means that when G is centered at any point in ψ, G overlaps with as many points of one sign as it does with points of the other sign. When $2kL, k \in \mathbb{Z}$ points are added to G, or length$(\hat{G}) = 2kL, k \in \mathbb{Z}$, the filter continues to overlap with as many points of one sign as the other. This is because kL points of each sign have been included. They cancel out the effect of each other, leaving ψ unaltered under the action of $T\hat{G}$. Alternatively, when we add $(2k+1)L, k \in \mathbb{Z}$ points to G or consider a filter $\hat{G}$ such that length$(\hat{G}) = (2k+1)L, k \in \mathbb{Z}$, we include as many points as half the period of the alternating subsequence. In effect, we include L points of one sign in excess of the other. This causes point on which the filter is centered to change its sign. This argument holds for point in ψ and hence every point in ψ changes its sign. Thus, $T\hat{G}$ has the opposite effect as TG on ψ. ∎

The above theorems state only some of the many properties and sufficient conditions for convergence. For an exhaustive listing of the sufficient conditions for convergence, together

with the number of steps required for the given sequence to converge, requires a rule-based approach for each case. It turns out that we cannot solve all the possibilities by hand for even a 7−tap filter. Working out the convergence rules on the computer leads to more than 800 cases. Moreover, the action of the skewing function has not been accounted for in this formulation. At one extreme, if the skewing function is identically zero, then there would be no change in the current formulation. At the other extreme, if the skewing force is so large, that the voting hardly makes a difference, then the masks align themselves purely according to the skewing function. In between the extremes are many possibilities which would depend on the specific skewing function as well as the voting function (dimension of ψ, properties of G and type of T) being used. The formulation would be closer to the actual problem but far more complicated to solve if we considered different weights applied to the smoothing filter g or used a different form of g (such as an exponential function). Finally, extending the analysis from 1D to 2D and higher dimensions is nontrivial.

Experimental convergence based on zero crossings. As we noted earlier, it is well established that iterative smoothing, using a lowpass filter such as a Gaussian (similar to the one used by V (7.11)) does not produce any new zero crossings. It is the action of the nonlinear majority-voting function in combination with the smoothing that has not been investigated. We used the same simplification as before, which is $M = 2$ and $\psi(p) \in \{-1, 1\}, p \in (Z^+)^d$ and ran experiments for $d \in 1, 2, 3$. In 1D, a zero-crossing is a change in sign from +1 to -1 or vice versa. In 2D and 3D we can easily extend the number of masks M to greater than 2 and define zero crossings as follows. In 2D, a point (p_1, p_2) has as many zero crossings as the number of its 4-adjacent (or 8-adjacent) neighbors that belong to a different mask. Similarly, in 3D, a point (p_1, p_2, p_3) has as many zero crossings as the number of its 6- (or 18- or 26-) adjacent neighbors that belong to a different mask. If the total number of zero crossings Γ, measured as the sum of the zero crossings of each point in ψ, is monotonically nonincreasing, then it forms the experimental proof of convergence of the procedure and establishes we are indeed studying the right property (or one of the right properties) to mathematical convergence. Indeed, computer simulations have revealed that

Γ is monotonically nonincreasing and, in fact, decreases with iteration at a very rapid rate. As this is still under investigation, we do not report these results here.

Other lines of investigation include drawing parallels to problems in areas such as graph theory, gaming theory and coding theory, where the action of majority voting or dynamical systems has been rigorously studied. Understanding the convergence of the distributing functions R and V described in this chapter, will help us characterize the properties of these functions rigorously as well as aid in designing other distributing functions for the AM framework. Thus, this is an important focus of our ongoing work in further developing the AM framework.

Algorithm 7: **[AM Segmentation]**
Input: Image f, initial number of masks M, maximum decomposition level K, the desired level of refinement K_0, smoothing filter scale parameters $a_{k,j}$, weight of the force to skew voting α, harshness of the threshold β, average intensity of pixels that lie on the foreground-background border γ and size of the voting window b.
Output: A collection of masks, ψ.

ActiveMasks(f,M,K,K_0,a,α,β,γ,b)

Initialization
$k = K$, $j = 1$, $\psi_1^{(k,j)}(n) = \mathrm{rand}(M)$
Beginning of multiresolution loop
while $k \geq K_0$ **do**
Compute smoothing filter for voting-based distributing function
$g^{(k)}(n) = \exp(-(2^{-k}b)^{-2}|n|^2)$
Beginning of multiscale loop
while $j \leq J_k$ **do**
Compute smoothing filter for the region-based distributing function at scale $a_{k,j}$
$h(n) = \exp(-a_{k,j}^{-2}|n|^2)$
Soft threshold the smooth image
$G_1(n) = \alpha\,\mathrm{sig}\big(\beta\big((f \star h)(n) - \gamma\big)\big)$
Compute a smoothed coarse version of the image and the region-based distributing function
$R_1 = H^k G_1$
$R_m = 0$ for $m = 2, \ldots, M$
Beginning of majority-voting loop
Iterate (and update M*) until the mask* $\psi^{(k,j)}$ *converges*
while $\psi_{\eta+1}^{(k,j)} \neq \psi_{\eta}^{(k,j)}$ **do**
$\psi_{\eta+1}^{(k,j)} = \underset{m=1,\ldots,M}{\mathrm{argmax}} \left[(\chi_{(\psi_\eta^{(k,j)})^{-1}\{m\}} \star g) + R_m\right]$
$M = \max(\{m \in \{1, \ldots, M\} : \psi_{\eta+1}^{(k,j)}(n) = m\})$
end while
$\psi_1^{(k,j+1)} = \psi_{\eta_{\max}}^{(k,j)}$
$j = j + 1$
end while
Lift mask ψ *to the next higher resolution* $k - 1$
$\psi_1^{(k-1,1)}(n) = \psi_{\eta_{\max}}^{(k,J_k)}(\lfloor 2^{-1}n \rfloor)$
$k = k - 1$
end while
return ψ

Chapter 8

AM for Fluorescence Microscopy

In this chapter we report the results we obtain on applying the active mask algorithm to the application described in Section 2.1.2. We discuss how the parameters of the algorithm can be tuned to adapt to the images to be segmented. Further, we compare the performance of the AM algorithm to that of the SW (as the SW algorithm is widely considered the most accurate segmentation algorithm in fluorescence microscopy). Finally, we discuss briefly the broader impact of the AM algorithm and some of the future directions of research and development of the AM framework.

8.1 Datasets

The data was generated in two experiments and are courtesy of Dr. A. D. Linstedt [9].

DS-1. The first dataset consists of 15 z-stacks containing $3-8$ cells each. Each stack contains 40 2D slices of size 1024×1344 pixels. The HeLa cells were double labeled with the COPII subunit Sec13 (a cytosolic protein peripherally associated with the membrane), and the Golgi marker protein, giantin, in two parallel channels (see Fig. 7.2(a) and Fig. 8.5(b)). Sec13 staining has a diffuse cytoplasmic background, which is used to mark the boundary of the cell (see Fig. 4.7 and Fig. 7.2(a)). The resolution was 0.05μm in x/y directions and 0.3μm in z direction. These images were used in the study and the findings based on hand segmentation were reported in [36]. We developed the cell-volume computation and

Golgi-body segmentation algorithm based on two of the stacks and tested it on others [19].

DS-2. The second dataset consists of 5 z-stacks containing $2-4$ cells each. Each stack contains 20 2D slices of size 1024×1344 pixels. The HeLa cells were labeled with Sec13. While the cells imaged in DS-2 belong to the same cell line as cells imaged in DS-1—the HeLa cell line—they are thinner, and thus a fewer slices per stack. The resolution was 0.05μm in x/y directions and 0.3μm in z direction.

Hand Segmentation. To assess the performance of the algorithm, we used the hand segmented (HS) version of the DS-2 set as ground truth. The HS version of the DS-1 set was not consistent enough to allow a rigorous comparison. The images were hand segmented in three separate sessions by two experts to minimize the effect of an individual's bias on a given day and giving them sufficient respite to segment as accurately and precisely as possible (see Fig. 8.2(a)-(b)).

8.2 Algorithm—Parameter Selection

We now discuss the considerations behind selecting parameters input to Algorithm 7. A table summarizing all the parameters, including image dependent ones, is given in the reproducible-research compendium that accompanies the paper [17].

Dimension d. In Section 8.3.1, we report the results when $d = 2$ in (7.3). In Fig. 8.4, we demonstrate the result when $d = 3$.

Masks M. We initialized the algorithm with $M = 256$ random masks. The larger the M, the lower the possibility of unwanted merges or of a single mask representing multiple regions.

Image resolutions K and K_0. In this work, we decomposed to $K = 3$ levels (that is, one-eighth the original resolution), lifting the result as per (7.2), to $K_0 = 2$ (that is, one-fourth the original resolution). We did not lift the result to the original resolution and refine any further as we obtained a satisfactory segmentation at $k = 2$. In our experiments, the two values of k adequately demonstrate the advantage of using multiple resolutions in anomalous cases where just the coarse segmentation at one-eighth the original resolution is

not accurate enough (see Fig. 7.2(g)-(j)). Thus we report our final results on images of size 256×336 pixels.

Filter h, and scale parameter a. Filter h is taken as,

$$h(n) = \exp(-n^2/a_{k,j}^2), \tag{8.1}$$

at resolution k and scale j.

For the MS evolution phase, we did not overly tune the parameters. We determined the scale parameter $a_{K,1}$ for filter h in (8.1) based on the resolution k as well as the approximate size of a cell at the coarsest resolution, $k = K$ and annealed the value at a fixed k. Thus, we began with $a_{3,1} = 4$ at one-eighth the original resolution of the input image and pulled back to $a_{3,3} = 3$ in decrements of 0.5. At $K - 1$, that is, one-fourth the original resolution, we used just one value, $a_{2,1} = 5$. In practice, the more gradual the annealing is, the faster the convergence is for different j at a fixed k. Further, while it is not necessary to decrement the values during the annealing process (that is, they can even be incremented), a larger a allows us to obtain a faster and coarser segmentation, whereas smaller scale values take longer to compute but provide a finer segmentation. We used the same values of a on all the images, without tuning this per image or even per stack. Scaling $a_{k,j}$ by the size of the image enables us to use the values without having to adjust the values of $a_{k,j}$ for different decomposition levels K that may be input to the algorithm.

Filter g and scale parameter b. Filter g may be taken to be any nonnegative lowpass filter. In particular, we have

$$g(n) = 0.5 \left(1 - \frac{2}{\sqrt{\pi}} \int_0^{\frac{n^2-b^2}{b^2+c}} \exp(-t^2/2) \mathrm{d}t \right). \tag{8.2}$$

We set the scale parameter $b = 1$ for the voting window g in (8.2). We annealed this value based on k. In particular, for $k = 3$, $b = 2$ and for $k = 2$, $b = 4$. In selecting the parameter we were guided by the twin goals of expediting convergence while not compromising on the quality of the boundary.

$c = 0.125 \ll b^2$. Since $b^2 + c \approx b^2$, c may be neglected.

Region-based function R and weight α. We may use any sigmoid-type function in R. In particular, we use the error function, $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2/2)\mathrm{d}t$. Thus, we have

$$R_m(n) = \alpha \left(\int_0^{\beta(\hat{f}(n)-\gamma)} \exp(-t^2/2)\mathrm{d}t \right), \tag{8.3}$$

where $m = 1$ and $\hat{f} = f * h$, is a smoothed (and downsampled) version of f.

$\alpha = -0.9$ is the weight of R_m's given by (8.3).

Image-Dependent Parameters, β and γ. We determined the constants β and γ as per the considerations described in Section 7.5. While the tuning was minimized to a large extent through annealing, tuning is not entirely eliminated. Indeed, as β and γ in (8.3) represent the harshness of the threshold and average value of pixels on the foreground-background border, they have to be adapted to each stack. We determined them based on one image from each stack and used the same numbers for all the other images from that stack.

These coarse adjustments are possible in our framework because the performance of the algorithm is robust within a small range of the parameter values and hence the values do not necessarily need to be highly tuned. It may be possible to obtain better results than we report with a more involved tuning procedure, if desired. However, it is not practical to expect an involved tuning of the parameters in high-throughput applications, and thus, we did not want to resort to such a tuning. The parameter values used for each stack as well as the scripts used to obtain the results reported the reproducible-research compendium that accompanies the paper [17].

8.3 Results

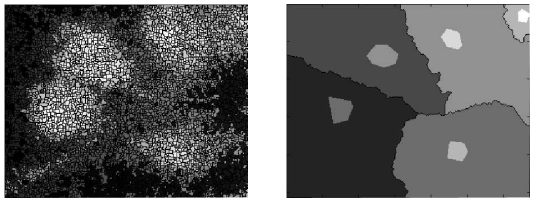
An example of the evolution behavior of AM algorithm is shown in Fig. 7.2. In summary, at a given k and $a_{k,j}$, the region-based distributing function (7.13) first separates the background coarsely from the foreground in one iteration and the number of masks drops drastically.

Thereafter, the masks are refined based on the voting-based distributing function (7.11), with empty masks being eliminated at each iteration. The segmentation outcome at a given resolution k and scale j is successively refined by annealing a and updating k.

8.3.1 Competing Algorithm: SW

We compared the performance of our AM algorithm to that of the SW algorithm. While the general idea of a region-growing method underlies its framework, various paradigms of the watershed algorithm abound in literature requiring the tuning of the algorithm. This tuning of the preprocessing or postprocessing modules for SW is involved and necessary for SW to perform well. Further, it is often tedious as it is rule-based and the rules are not easy to generalize. As the algorithm's performance critically depends on these pre/postprocessing modules, we spent a fair amount of time tuning them to get the best possible performance. When we use the watershed algorithm described in [18], the algorithm bins regions in the image based on their gray scale values. We obtain a large number of regions (spurious splits) despite blurring the fluorescence microscope images to reduce the effect of shot noise (see Fig. 8.1(a)). This result requires a heavy postprocessing of defining rules to merge these highly fragmented regions to provide a reasonable number of regions [18]. Thus, we resorted to using perfect seeds drawn by hand to initialize the algorithm and used the classic region-growing paradigm of the SW to segment the image [102], whereby SW produced the same number of segmented regions as that of the seeds. However, even this proved to be insufficient as the entire image was assigned to one region or another (see Fig. 8.1(b)). Thus, the resulting masks for the cells were not tight and included a significant portion of the background. We then further tuned the seeding procedure to seed regions in the background to prevent the foreground regions from expanding to the image borders. The result produced tighter masks (see Fig. 8.2(c)) which we used for comparison with results produced by AM.

We discuss the performance of the algorithm in terms of a qualitative assessment (visual inspection), quantitative assessment, as well as a report of the runtime.



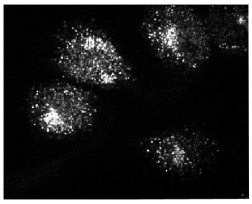
(a) Random seeds (b) Perfect foreground seeds

Figure 8.1: SW results: (a) SW with random seeds results in 5179 regions that need to be merged based on rules [18]. (b) SW using perfect foreground seeds (light gray) but no background seeds results in regions that include a significant portion of the background with each cell. (Original image courtesy of Dr. A. D. Linstedt [5].)

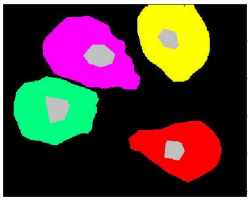
8.3.2 Qualitative Evaluation

We performed a qualitative assessment of the algorithmic results by comparing them with the HS masks (see Fig. 8.2). Very often, both SW as well as AM seemed to include a slightly larger area than that marked by the hand segmentation.

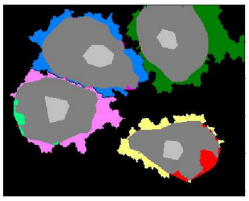
The results of AM seemed to match the area marked by the hand segmenters better than the masks produced by SW (see Fig. 8.2(c)-(e)). Further, it appeared that AM initialized with random seeds performed almost as well as when it was initialized with perfect seeds. Although this is the case for a majority of the images, due to a random initial configuration, there are some anomalous results as well, as the one shown in Fig. 8.3(a). If indeed the problem was only that of an unfavorable initial configuration, often just rerunning the algorithm without changing any parameter produces a better result. If the problem resulted from insufficient information or a choice of parameters that induced the masks to split a cell at a particular scale, annealing the scale parameters of the filter(s) in the distributing function(s) eliminates spurious splits during the course of the evolution itself (see Fig. 7.2(b)-(j)). Introducing details available in a higher-resolution version of the image can also help recover from such spurious splits. However, these techniques built into the mask evolution phase do not help in recovering from a spurious merge during the course of the evolution. As it is not always possible to rerun the algorithm, if seeds are available in the form of reliable information from a parallel channel, for example, they do help in limiting anomalies resulting from the initial randomness (see Fig. 8.3(b)).



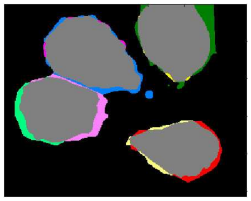
(a) Original image



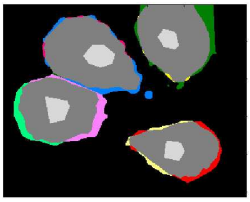
(b) HS and perfect seeds



(c) SW (perfect seeds)



(d) AM (random seeds)



(e) AM (perfect seeds)

Figure 8.2: (a) An image of HeLa cells marked with COPII at one-fourth the original resolution. (b) HS masks, together with "perfect seeds" (drawn by hand). (c) SW initialized with perfect seeds and overlaid on the HS masks for comparison. $AO = 95.86\%$ and $AS = 82.48\%$. (d) AM initialized with random seeds and overlaid on the HS masks for that image. $AO = 94.60\%$ and $AS = 91.80\%$. (e) AM initialized with perfect seeds and overlaid on HS masks. $AO = 94.60\%$ and $AS = 91.80\%$ [17]. (Original image courtesy of Dr. A. D. Linstedt [5].)

(a)

(b)

Figure 8.3: AM results: (a) A merge—two cells covered by the same mask. (b) Given reasonable seeds, the anomaly from (a) disappears [17].

8.3.3 Quantitative Evaluation

To quantify the performance of the algorithm, we computed performance measures for each cell in each of 80 different 2D slices. As the density of cells varies per image, to account for the influence of neighboring cells on the performance of the algorithm for a cell, we averaged the performance measures per image before computing the overall average for the dataset.

Further, HS masks were provided for only whole cells whereas AM segmented every cell in the image, whether it was in the periphery or at the center, while SW tended to include peripheral cells in the same region as a neighboring central cell. For SW, this required

seeding the incomplete cells to avoid leakage of watershed regions of valid whole cells in the image. However, to compute the numbers, we considered only those cells that were chosen in the HS mask.

We did not compute performance measures for SW algorithm using random seeds as numbers would be meaningless since the performance was dismal without an involved post-processing to merge the highly oversegmented regions.

We used two standard performance measures—area overlap (AO) and area similarity (AS). Summaries of the results averaged per image and per cell respectively are presented in Tables 8.1 and 8.2.

	AO[%]		AS[%]	
	Mean	Std. Dev	Mean	Std. Dev.
Random seeding				
SW	NA	NA	NA	NA
AM	89.13	8.58	84.73	7.90
Perfect seeding				
SW	94.66	6.04	72.94	13.90
AM	89.35	7.67	86.06	7.15

Table 8.1: Performance measures: area overlap (AO) and area similarity (AS) measures (means and standard deviations) for the active mask (AM) algorithm and seeded watershed (SW). Averages are first computed for the cells in each image and then averaged over the entire set of test images [17].

$AO_c = \big(n(\mathrm{HS}_c \wedge \mathrm{AM}_c)\big)/n(\mathrm{HS}_c)$, for cell c gives the percentage of overlap between the HS and the algorithm (AM or SW) masks for c. It can be thought of as a measure of true positives. Considering the performance measures averaged per image, we see SW had an average $AO = 94.66\%$ with a sample standard deviation, $s = 6.04$ for perfect seeding. This is higher than that of AM which had an average $AO = 89.13\%$ with $s = 8.58$ and $AO = 89.35\%$ with $s = 7.67$ for random seeding and perfect seeding respectively. Thus, SW has a tendency to pick out more true positives than AM, which is relatively conservative.

However, AO does not give an accurate measure of how *similar* in area the masks produced by the algorithms are to the hand segmentation. It does not penalize an algorithm

	AO[%]		AS[%]	
	Mean	Std. Dev.	Mean	Std. Dev.
Random seeding				
SW	NA	NA	NA	NA
AM	89.26	12.38	85.11	11.34
Perfect seeding				
SW	94.90	6.80	73.10	15.17
AM	89.96	8.85	86.20	9.09

Table 8.2: Performance measures: AO and AS measures (means and standard deviations) for AM and SW. Averages are computed over the entire collection of cells. This does not take into account there may be more cells in some images than others or the cells may be more closely packed in some images compared to others [17].

that does not produce tight contours or includes a significant portion of the background. For example, the mask in Fig. 8.1 may yield an $AO = 100\%$, but that does not reflect how closely the mask matches the ground truth. Including a significant portion of the background in the cell mask may negatively influence further automated analyzes such as feature extraction. Thus, we also compute a more stringent measure called the area similarity AS.

AS normalizes twice the area that is common to the masks by the sum of the areas of HS and the algorithm, and thus penalize the algorithm that produces larger regions. For SW, $AS = 72.94\%$ with $s = 13.90$ for perfect seeding, while for AM, $AS = 84.73\%$ with $s = 7.90$ and $AS = 86.06\%$ with $s = 7.15$ for random seeding and perfect seeding, respectively. According to literature, an $AS \geq 70\%$ implies a good agreement of the algorithm's result with the ground truth [131]. By this token, both SW and AM algorithms perform satisfactorily, though the large standard deviations give AM a bit of an edge. Moreover, according to the numbers, AM has a lesser chance of detecting false positives in the image. The performance of AM with random seeding is, as highlighted even in the qualitative assessment, not significantly different from that of perfect seeding. This indicates that the algorithm almost always performs just as well with random seeding as it does with perfect seeding. Exceptions to this premise (see Fig. 8.3) have been discussed earlier in Section 8.3.2.

8.3.4 Runtime

While our AM implementation was in MATLAB (version 2007a) for flexible design and prototyping [150], we used library functions for SW available in C++. Thus, while the runtime numbers of the two algorithms cannot be compared, we list them here for completeness. In our future work, we expect at least an order of magnitude decrease in the runtime of AM once implemented in C/C++. Further optimization of the C code may yield anywhere between a 10x to a 1000x speedup, if the problem is not memory bound (that is, a lot of data and very few computations) [151].

AM Runtime. The runtime of AM depends on the initial random configuration, as well as the number of regions to be segmented and the size of the window over which the majority voting is performed. On an Intel Pentium M 1.6GHz Processor with 1.5GB memory, for a images of size 1024×1344, it took AM $0.8 - 1.6$ min on average at one-fourth the original resolution. Since the algorithm is moderated by the number of changing pixels, rather than a fixed number of iterations, we note that most of the changes towards the end are on the order of 10 pixels (which is less than 0.1×10^{-3} times the total number at one-fourth the resolution and nearly one-millionth of the pixels in the original resolution). As there is no significant difference between the segmented regions corresponding to masks with such small changes between iterations, one could terminate the algorithm when the number of changing pixels reaches a sufficiently small fraction of the total number of pixels in the image. Further, from [19] we know that the processing time for a stack can be drastically reduced if the mask from one image is used to initialize the segmentation of a neighboring image in the stack. As expected, while not by much, the convergence time for each image is further reduced if perfect seeds rather than random ones are used.

SW Runtime. For an images of size 256×336, it took SW 0.2 sec on average. Recall again that was using C++ precompiled library functions (although we designed the preprocessing of the images for SW in MATLAB 2007a).

8.4 Application-Specific Processing

In Section 2.1.2, we introduced a specific application that motivated the design of an automated segmentation algorithm for a class of biological images. As cell-volume computation and Golgi-body segmentation were of particular interest, we report how these quantities were computed, given the segmentation outcome of the AM algorithm. We note that these are just two of the application-specific postprocessing modules that may be used. One could imagine many other applications for which postprocessing modules, such as tracking masks in time-lapse images, could be designed.

8.4.1 Cell-Volume Computation

Given the AM segmentation outcome, each $\chi_{\psi^{-1}\{m\}}$ in (7.7) represents a distinct region. If we set $d = 2$ in (7.3), and segment the images of a stack in 2D, to obtain an approximation of the cell volume, we can simply sum the areas of the 2D masks, as it is done when hand segmentation of the 2D images is used to process the images. To ensure the cells along a stack are assigned the same mask number, we may initialize one of the slices in a stack where the cells are fairly discernible with an initial $M \gg C$ masks, where C is the expected number of cells. The segmentation outcome of this slice can be used to initialize the neighboring masks and so on. Post-processing to match a region in one mask with the most overlapping region in a successive mask may also be used in computing the volume from a 2D segmentation procedure [19]. This pseudo-3D segmentation (as the 2D segmentation benefits from the information in 3D) also speeds up segmentation of a stack over segmenting each image in the stack independently.

Alternatively, we could instantiate dimension $d = 3$ in (7.3) and segment in 3D. Such a segmentation affords an elegant way of visualizing the cells in the z-stack and the segmentation outcome is not hampered when a cell is occluded by another cell in a 2D section (see Fig. 8.4). Further, segmentation masks are contiguous and the volume of each cell can be computed in a straight-forward manner based on the number of pixels contained within each mask.

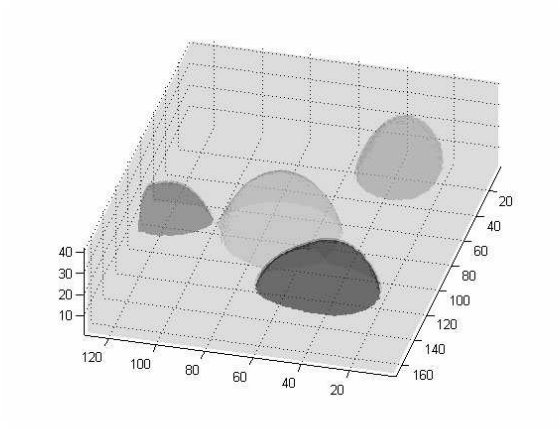
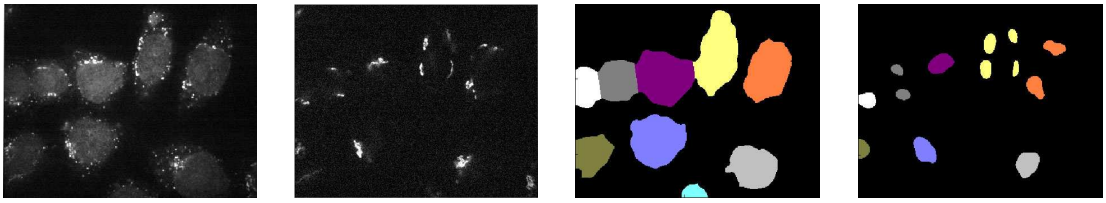


Figure 8.4: AM results: 3D segmentation of a z-stack [17].

8.4.2 Golgi-Body Segmentation

The Golgi-body is a double-membraned organelle that is comprised of cisternae stacked to increase its surface area to facilitate secretion. Thus, though there is only one Golgi-body in each cell, in a 2D section, a Golgi-body may appear as multiple fragments (see Fig. 8.5(b)). If the cells are close, then by using the Golgi channel alone it is not easy to associate the different pieces of the organelle in 2D with others that belong to the same cell. Given the AM segmentation of cells for the COPII channel of an image, we may use the masks to initialize the segmentation of the corresponding Golgi channel. The advantage is that multiple pieces of the Golgi body belonging to a cell are marked by the same mask (see Fig. 8.5(d)). Moreover, such a Golgi mask matches the cell to which the different Golgi fragments belong (see Fig. 8.5(c)). This facilitates the computation of the Golgi-volume and, subsequently, the ratio of the Golgi volume to the cell volume as required by the application.

Due to the low quality of images in the peripheral slices of DS-1, HS was not reliable enough to rigorously compute quantitative measures of performance for this application. However, visual inspection (see Fig. 8.5) establishes the use of this method. The code to reproduce this result is provided on the web page of the reproducible-research compendium that accompanies the paper [17].



(a) Cell image (b) Golgi image (c) Cell mask (d) Golgi mask

Figure 8.5: (a) Original HeLa cell image (COPII, also shown in Fig. 7.2(a)). (b) Original HeLa Golgi image (giantin). (c) AM segmentation of the parallel cell image (in pseudo color). (d) AM segmentation of the corresponding Golgi image [19]. Cell segmentation helps associate the different fragments of the Golgi-body in a 2D slice of a cell to their corresponding cells. (Original images courtesy of Dr. A. D. Linstedt [5].)

8.5 AM Segmentation of DIC Stem Cell Images for Tracking

Like all its predecessor frameworks, AM can be extended to segment other modalities by suitably incorporating new forces. Here we present the results of a preliminary study on applying AM to the segmentation of DIC stem cell images used in a tracking application [152]. We use the same distributing functions as those used to segment fluorescence microscope images with appropriately chosen scale parameters. The segmentation is preceded by a preprocessing stage to render the images suitable for segmentation with AM.

Stem cells are cells that proliferate thorough mitosis and can differentiate into various types of specialized cells. The enormous potential of stem cells in treating diseases like Alzheimer's (neuro-stem cells) and in applications such as regenerative medicine (as skin or bone cells) among others, has escalated the significance of understanding the behavior of stem cells under various experimental conditions such as the presence of growth factors, other enzymes and the matrix on which they grow [153].

The most common imaging modalities used to study these cells are DIC and phase-contrast microscopy as they do not require introducing probes into the specimens, which might interfere with the pristine nature of these cells. DIC and phase-contrast microscopy also provide the necessary information of cell morphology and thus prove useful for studying stem cells. Imaging studies on stem cells are usually conducted as time-lapse experiments that enable biologists to observe the rate of proliferation and nature of migration of these cells with time. Understanding these properties of stem cells requires that the cells

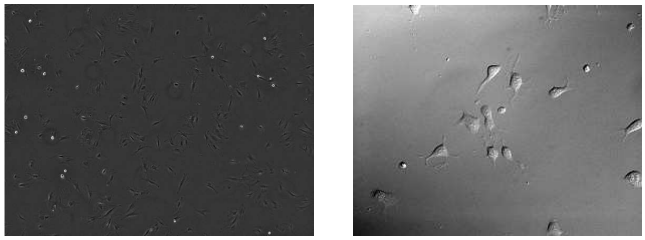
be *tracked* with time. As the number of cells in each frame can be enormous and the number of frames, depending on the resolution in time and duration of the imaging, very large, the problem is intractable for manual processing. Even when a small frame is selected for the study, in the interest of efficacy, efficiency and reproducibility, automating the tracking of cells proves advantageous.

To be able to track a stem cell, it is necessary to first detect it in the image. As these cells are motile and deform both during motion as well as proliferation, detecting just the centroid of the cell or a cluster of points within the cell is insufficient. As segmentation provides the entire object of interest, it is one of the necessary processes that precede the tracking step. Among the segmentation algorithms, active contours or snakes provide the boundaries of these cells with a high degree of accuracy and are thus, popularly used. As we have discussed in Chapter 7, the AM framework is tailored to digital image segmentation and, by design, would like it to perform no worse on an application than the active-contour framework adapted to it. Moreover, if the AM algorithm is applied on an image and the outcome used to initialize an image at a later time point, there is an added advantage. When cells that have been segmented in a frame proliferate, the *daughter cells* inherit the same mask number as the parent cell and inherently, provide tracking information. As DIC is a popular imaging modality, we describe below some of a very preliminary effort towards applying the AM algorithm to this problem.

8.5.1 Dataset

DIC microscope images from NIH and phase-contrast microscope images from MRTC at Carnegie Mellon University, were made available to us by Dr. T. Kanade et. al. [20]. These consist of nearly 1000 time-series images of stem cells, taken five minutes apart. Sample images of stem cells from DIC and phase-contrast microscopy are shown in Fig. 8.6.

As seen in the images, a major challenge in this problem is either the cells are too many and the contrast too low or the complete extent of the cells is not clear.



(a) Phase contrast (b) DIC

Figure 8.6: Same images of stem cells as seen under (a) DIC and (b) phase contrast microscopy. (Original images (from two separate studies) courtesy of Dr. T. Kanade and his team [20].

8.5.2 Algorithm

Rather than design a new algorithm (distributing functions), we applied a few preprocessing steps to the DIC images. The aim of the preprocessing is to render the DIC images suitable to be segmented by the distributing functions designed for fluorescence microscope images (with appropriately chosen parameters).

Preprocessing. In DIC images, rather than the mean, the local variance distinguishes the foreground from the background. So instead of computing local averages, we first compute a local variance map for the image. Each pixel is replaced by the variance of the grayscale values in a small neighborhood around it. Then, we perform adaptive, nonlocal filtering to eliminate high variances in the background or shot noise that might lead to spurious detection. The adaptivity refers to fact the smoothing filter is oriented to smooth along the edges and not across them. Thus the edges in the image are preserved. We then threshold the image and apply morphological operations to further eliminate spurious detections. This binary image can be applied on the original image to extract the foreground information relevant for segmentation and eliminate the background. The resulting image can be used to compose the region-based distributing function that can be used to skew the local majority voting distributing function used by AM to segment it.

AM Segmentation. We initialize ϕ for an image in the time series with a large number of random masks. In the case of DIC images that have fewer cells, we could take $M = 256$. The disadvantage of starting with a very large M is that the algorithm would have a higher

chance of getting stuck in a local optimum (a single cell is split into multiple regions). We segment the image based on the region-based distributing function obtained from the above preprocessing step and the local majority voting-based distributing function described in Chapter 7, but with very small scale parameters. This is because stem cells are much smaller than the fluorescence microscope images of HeLa cells. Moreover, stem cells are comparatively distinct, unlike the punctate patterns of protein location images, obviating the need for aggressive blurring. As in the case of fluorescence microscope cell images, the algorithm comes to a natural halt.

We use the segmentation outcome ψ of this image to initialize the ψ for the subsequent image in the time series. This allows us to track the movement of the cell if the time resolution is such that a cell in one frame is spatially closer to itself in the succeeding frame rather than any other cell. Moreover, at such a resolution, if the cell undergoes division, then the daughter cells inherit the same mask label as the parent. These implicitly afford tracking information for the cells.

8.5.3 Results

We computed the validity of cell detection based on HS as reference. This is summarized in Table 8.3. A true positive is any cell that has been detected by AM and by HS. A false positive is the detection of any spurious (noncell) object and finally, a false negative is an object marked as a cell by a manual segmenter but undetected by AM.

Cell detection	Quantity[%]
True positives	76.47
False positives	17.65
False negatives	5.88

Table 8.3: Quantitative assessment of AM segmentation of DIC stem cell images.

Discussion. The result we report is for an image with relatively few cells (as shown in Fig. 8.6(b)). We note that this result is a very preliminary one as we have barely tuned the algorithm for this problem. However, this is promising in that we achieve a true detection

rate of more than 75%. In practice, as the number of cells is very large, especially in phase contrast images, rarely are all the cells in a frame tracked. Usually, studies focus on a small frame in the image. Further, we have a false positive rate that is rather high. While we would endeavor to minimize this number in our future design, we note that for the particular application of tracking stem cells, a higher false positives are considered better than a higher false negative rate.

8.6 Future Directions

The theory of AM and experimental results have opened up various venues of further exploration. One of our first steps would be to investigate mathematical convergence of the distributing functions described in Chapter 7. This would be crucial for rigorously characterizing the behavior of these functions and lend insight to designing new functions for different applications.

There are various different functions that we could use in addition to (or instead of) the R and V described in Chapter 7. For instance, the R we have described distinguishes only the foreground from the background. There are many applications (such segmentation of tissue images or color images), where there are multiple regions of interest with different properties. Thus, R would be vector-valued and could include an edge-map or texture features relevant to the problem. Likewise, V may take on various different forms. The V we have used takes into account for the voting only points along the interface between two masks within a neighborhood given by the scale parameter b and gives equal weight to each of these points. A simple experiment of weighting the vote of each point based on different functions of its distance from the interface (or the mask's center) revealed highly different behavior. One could well imagine different functions, such as one that takes into account the total size of a mask itself, for the voting procedure for different applications.

As described in Chapter 1, there are a host of other data-dependent issues such as different types of initialization procedures and application-specific postprocessing modules, which could be designed to be used with the AM framework. In addition to expanding the

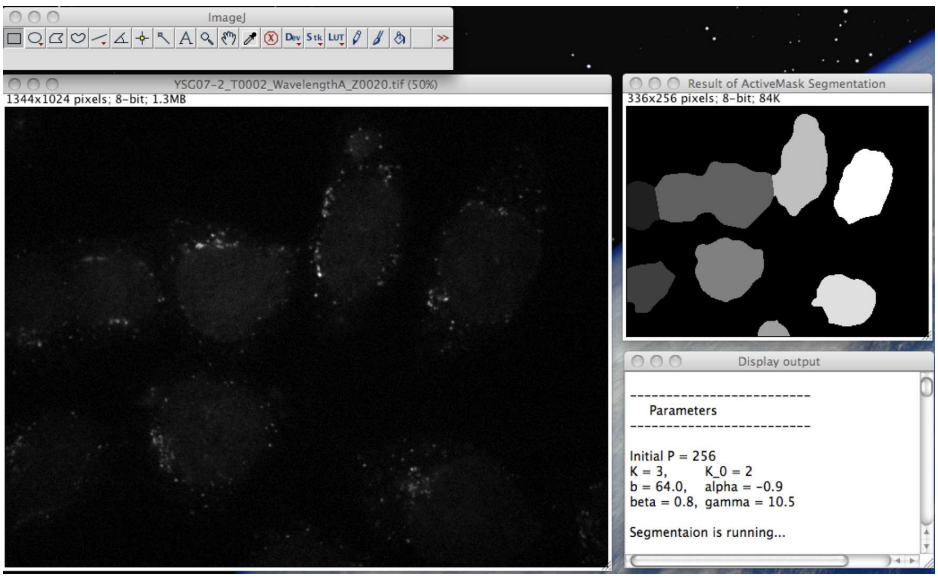


Figure 8.7: A screen shot of the user interface of a preliminary version of the ImageJ plugin of the active mask algorithm.

algorithm itself, we are also working in parallel to design into a plugin for ImageJ, so the algorithm is accessible to end users as well as other tool developers for testing and further development. Fig. 8.7 shows a screen shot of the preliminary version of this plugin.

Conclusions

We have introduced the task of segmentation in the context of biomedical imaging. In particular, we have focused on segmentation of punctate patterns in fluorescence microscope images. We have demonstrated the benefit of a flexible mathematical framework at the segmentation core and data-specific modules, through various instantiations of an active-contour based framework—STACS. Thereafter, we have highlighted the advantage of multiscale transformations, particularly for the segmentation of fluorescence microscope images and those that do not require topology preservation. These design considerations and results inspired the change in perspective from "contours" to "masks".

In this work, we have proposed a novel automated segmentation framework for fluorescence microscope images based on active masks, suited to digital images of any dimension, particularly those with punctate or diffuse staining patterns. By the same token, the AM algorithm may also be used to segment data exhibiting similar properties (punctate patterns with some underlying structure) such as, satellite images of the earth taken at night. AM is able to achieve almost as good a segmentation outcome with random seeding as it does with perfect seeding. This framework lends itself naturally to the incorporation of multiscale and multiresolution techniques. These facilitate segmenting fluorescence microscope cell images as well as increase the algorithm's computational efficiency. While the parameters to be tuned are intuitive, AM experimentally converges to a zero change state, eliminating the need for setting the number of iterations (or any other stopping criterion) *a priori*. All of these add to AM's flexibility and ease of use.

We compared the AM algorithm to the SW one, and shown AM to be highly competitive,

both qualitatively and quantitatively. We have also shown how the segmentation results may be used with application-specific postprocessing modules using the example of cell-volume computation and Golgi-body segmentation for the study of the influence of Golgi protein expression on the cell size. Thus, we have demonstrated that AM is a viable alternative to hand segmentation or any standard algorithm (such as SW or level set based methods) of such images.

We have highlighted some of the issues in proving theoretical convergence. In the short-term, we envisage to rigorously characterize the behavior of the algorithm to lend a deeper insight into its capabilities and limitations. We have also highlighted one of the next steps of further developing the AM framework by way of expanding the repertoire of distributing functions. Akin to forces in the active-contour framework, different distributing functions can be used to segment a wider range of images such as those of tissues as well as images that possess features such as edges, that have been traditionally relied upon for segmentation.

Bibliography

[1] P. Nair, B. E. Schaub, K. Huang, X. Chen, R. F. Murphy, J. M. Griffith, H. J. Geuze, and J. Rohrer, "Characterization of the TGN exit signal of the human mannose 6-phosphate uncovering enzyme," *Journ. Cell Science*, vol. 118, no. 13, pp. 2949–2956, 2005.

[2] The Murphy Lab at Carnegie Mellon University: http://murphylab.web.cmu.edu.

[3] M. Velliste and R. Murphy, "Automated determination of protein subcellular locations from 3D fluorescence microscope images," in *Proc. IEEE Int. Symp. Biomed. Imaging*, Washington, DC, 2002, pp. 867–870.

[4] A. Chebira, Y. Barbotin, C. Jackson, T. Merryman, G. Srinivasa, R. Murphy, and J. Kovačević, "A multiresolution approach to automated classification of protein subcellular location images," *BMC Bioinformatics*, vol. 8, no. 210, 2007.

[5] "LinstedtLab at Carnegie Mellon University," http://www.andrew.cmu.edu/user/-linstedt/Home.htm.

[6] G. Habeler, K. Natter, G. Thallinger, M. Crawford, S. Kohlwein, and Z. Trajanoski, "YPL.db: The yeast protein localization database," *Nucleic Acids Research*, vol. 30, pp. 80–83, 2002.

[7] "Yeast GFP fusion localization database," http://yeastgfp.ucsf.edu/.

[8] E. Angelini, A. Laine, S. Takuma, J. Holmes, and S. Homma, "LV volume quantification via spatiotemporal analysis of real-time 3-d echocardiography," *IEEE Trans. Med. Imag.*, vol. 20, pp. 457–469, 2001.

[9] "Heffner Biomedical Imaging Lab at Columbia University," http://medimage.bme.columbia.edu/.

[10] "HOPES: Huntington's Outreach Project for Education, at Stanford," http://www.stanford.edu/group/hopes/basics/braintut/f_ab09whtgrymt.gif.

[11] "VISTA: The vision, imaging science and technology activities Lab at Stanford," http://white.stanford.edu/.

[12] "Conservation technology information center," http://www.ctic.purdue.edu/KYW/-glossary/whatisaws.html.

[13] L. Coulot, H. Kischner, A. Chebira, J. Moura, J. Kovačević, E. Osuna, and R. Murphy, "Topology preserving STACS segmentation of protein subcellular location images," in *Proc. IEEE Int. Symp. Biomed. Imaging*, Arlington, VA, Apr. 2006, pp. 566–569.

[14] "MurphyLab at Carnegie Mellon University," http://murphylab.web.cmu.edu/.

[15] G. Srinivasa, M. Fickus, and J. Kovačević, "Multiscale active contour transformations for the segmentation of fluorescence microscope images," in *Proc. SPIE Conf. Wavelet Appl. in Signal and Image Proc.*, San Diego, CA, Aug. 2007, to appear.

[16] G. Srinivasa, V. S. Oak, S. J. Garg, M. C. Fickus, and J. Kovačević, "3D STACS segmentation of fMRI images of the brain," in *Proc. IEEE Int. Conf. Image Proc.*, Oct. 2008.

[17] G. Srinivasa, M. C. Fickus, Y. Guo, A. D. Linstedt, and J. Kovačević, "Active mask segmentation of fluorescence microscope images," *IEEE Trans. Image Proc.*, May 2008, submitted.

[18] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 13, no. 6, pp. 583–598, Jun. 1991.

[19] G. Srinivasa, M. C. Fickus, M. N. Gonzales-Rivero, S. Y. Hsieh, Y. Guo, A. D. Linstedt, and J. Kovačević, "Active mask segmentation for the cell-volume computation and Golgi-body segmentation of HeLa cell images," in *Proc. IEEE Int. Symp. Biomed. Imaging*, May 2008, pp. 348–351.

[20] "Tracking of Stem Cells at MRTC, Carnegie Mellon University," http://www.ri.cmu.edu/projects/project_579.html.

[21] "bimagicLab at Carnegie Mellon University," http://www.andrew.cmu.edu/user/-jelenak/Research/research_biomolecular.html.

[22] R. J. Zhang, Campbell, A. Ting, and R. Tsien, "Creating new fluorescent probes for cell biology," vol. 3, pp. 906–918, 2002.

[23] X. Michalet, A. Kapanidis, T. Laurence, F. Pinaud, S. Doose, M. Pflughoefft, and S. Weiss, "The power and prospects of fluorescence microscopies and spectroscopies," *Ann. Rev. Biophys. and Biomol. Str.*, vol. 32, pp. 161–182, 2003.

[24] S. Inoue, *Handbook of Biological Confocal Microscopy*, 2005, ch. Foundations of Confocal Scanned Imaging in Light Microscopy.

[25] V. Abraham, D. Taylor, and J. Haskins, "High content screening applied to large-scale cell biology," vol. 22, no. 1, pp. 15–22, 2004.

[26] K. Huang and R. Murphy, "From quantitative microscopy to automated image understanding," *Journ. Biomed. Optics*, vol. 9, pp. 893–912, 2004.

[27] R. Murphy, "Location proteomics: A systems approach to subcellular location," *Biochem. Soc. Trans.*, vol. 33, pp. 535–538, Mar. 2005.

[28] P. Perner, H. Perner, and B. Muller, "Mining knowledge for Hep-2 cell image classification," *Journ. Artif. Intelligence in Medicine*, vol. 26, pp. 161–173, 2002.

[29] A. Danckaert, E. Gonzalez-Couto, L. Bollondi, N. Thompson, and B. Hayes, "Automated recognition of intracellular organelles in confocal microscope images," *Traffic*, vol. 3, pp. 66–73, 2002.

[30] R. Formanek Jr., "Proteomics: Moving beyond the human genome," FDA Consumer magazine, Tech. Rep., 2005, http://www.fda.gov/fdac/-features/2005/605_proteomics.html.

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

[32] M. Aridor and L. Hannan, "Traffic jam: a compendium of human diseases that affect intracellular transport processes," *Traffic*, vol. 1, no. 11, pp. 836–51, 2000.

[33] ——, "Traffic jams II: an update of diseases of intracellular transport," *Traffic*, vol. 3, no. 11, pp. 781–90, 2002.

[34] J. Donaldson and J. Lippincott-Schwartz, "Sorting and signaling at the Golgi complex," *Cell*, vol. 101, no. 7, pp. 693–6, 2000.

[35] M. Puthenveedu and A. Linstedt, "Subcompartmentalizing the Golgi apparatus," *Curr. Opin. Cell Biol*, vol. 17, no. 4, pp. 369–75, 2005.

[36] Y. Guo and A. Linstedt, "COPII-Golgi protein interactions regulate COPII coat assembly and Golgi size," *Journ. Cell Biol.*, vol. 174, no. 1, pp. 53–56, 2006.

[37] F. Sherman, *The Encyclopedia of Molecular Biology and Molecular Medicine.* Weinheim, Germany: VCH, 1997, vol. 6, ch. Yeast genetics, pp. 302–325.

[38] R. Howson, W. Huh, S. Ghaemmaghami, J. Falvo, K. Bower, A. Belle, N. Dephoure, D. Wykoff, J. Weissman, and E. O'Shea, "Construction, verification and experimental

use of two epitope-tagged collections of budding yeast strains: Research papers," *Comparative and Functional Genomics*, vol. 6, pp. 2–16, Feb. 2005.

[39] B. Wandell, A. Brewer, and R. Dougherty, "Visual field map clusters in human cortex," *Philosoph. Trans. of the Royal Soc. B: Biol. Sci.*, vol. 360, no. 1456, pp. 693–707, 2005.

[40] R. F. Dougherty, V. M. Koch, A. A. Brewer, B. Fischer, J. Modersitzki, and B. A. Wandell, "Visual field representations and locations of visual areas V1/2/3 in human visual cortex," *Journ. Vision*, vol. 3, pp. 586–598, 2003.

[41] B. A. Wandell, "Computational neuroimaging of human visual cortex," *Ann. Rev. Neuroscience*, vol. 10, no. 22, pp. 145–173, 1999.

[42] N. Gogtay, J. N. Giedd, L. Lusk, K. M. Hayashi, D. Greenstein, A. C. Vaituzis, T. F. N. III, D. H. Herman, L. Clasen, A. Toga, J. Rapoport, and P. Thompson, "Dynamic mapping of human cortical development during childhood through early adulthood," *Proc. Nat. Acad. Sci. of USA*, vol. 101, no. 21, pp. 8174–8179, 2004.

[43] H. Zaidi, T. Ruest, F. Schoenahl, and M. L. Montandon, "Comparative assessment of statistical brain MR image segmentation algorithms and their impact on partial volume correction in PET," *NeuroImage*, vol. 32, pp. 1591–1607, 2006.

[44] P. C. Teo, G. Sapiro, and B. A. Wandell, "Creating connected representations of cortical gray mater for functional MRI visualization," *IEEE Trans. Med. Imag.*, vol. 16, no. 6, pp. 852–863, 1997.

[45] R. Gonzalez and R. Woods, *Digital Image Processing*, 2002.

[46] K. Fua and J. Muib, "A survey on image segmentation," vol. 13, no. 1, pp. 3–16, 1981.

[47] N. Pal and S. Pal, "A review on image segmentation techniques," vol. 26, no. 99, pp. 1277–1294, 1993.

[48] P. Sahoo, S. Soltani, A. Wong, and Y. Chen, "A survey of thresholding techniques," *Comp. Vis., Graph. and Image Proc.*, vol. 41, no. 2, pp. 233–260, 1988.

[49] P. Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection," vol. 24, no. 14, pp. 2345–2356, 2003.

[50] D. Pham, C. Xu, and J. Prince, "Current methods in medical image segmentation," vol. 2, pp. 315–337, 2001.

[51] D. Ziou and S. Tabbone, "Edge detection techniques-an overview," vol. 8, no. 4, pp. 537–559, 1998.

[52] L. S. Davis, "A survey of edge detection techniques," *Comp. Graphics and Image Proc.*, vol. 4, pp. 248–270, 1975.

[53] R. C. Hardie and C. G. Boncelet, "Gradient-based edge detection using nonlinear edge enhancing prefilters," *IEEE Trans. Image Proc.*, vol. 4, no. 11, pp. 1572–1577, 1995.

[54] V. Berzins, "Accuracy of laplacian edge detectors," *Comp. Vis., Graphics and Image Proc.*, vol. 27, pp. 195–210, 1984.

[55] J. Canny, "A computational approach to edge detection," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 8, pp. 679–698, 1986.

[56] K. S. Fu and J. K. Mui, "A survey on image segmentation," vol. 13, no. 1, pp. 3–16, 1981.

[57] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Comp. Vis., Graphics and Image Proc.*, vol. 29, no. 1, pp. 100–133, 1985.

[58] L. Najman and M. Couprie, "Watershed algorithms and contrast preservation," *Discrete Geometry for Computer Imagery in the Lecture Notes in Computer Science*, vol. 2886, pp. 62–71, 2003, http:/citeseer.ist.psu.edu/najman03watershed.html. [Online]. Available: http:/citeseer.ist.psu.edu/najman03watershed.html

[59] T. R. Reed and J. Buf, "A review of recent texture segmentation and feature extraction techniques," vol. 57, no. 3, pp. 359–372, 1993.

[60] R. Nock and F. Nielsen, "Statistical region merging," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 26, no. 11, pp. 1452–1458, 2004.

[61] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. Journ. of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[62] L. Cohen, "On active contour models and balloons," *CVGIP: Image Understanding*, vol. 53, no. 2, pp. 211–218, Mar. 1991.

[63] R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 17, no. 2, pp. 158–175, Feb. 1995.

[64] C. Xu and J. Prince, "Snakes, shapes and gradient vector flow," *IEEE Trans. Med. Imag.*, vol. 7, pp. 359–369, Mar. 1998.

[65] T. Chan and L. Vese, "Active contours without edges," *IEEE Trans. Image Proc.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[66] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic snakes," *Int. Journ. of Computer Vision*, vol. 22, pp. 61–79, 1997.

[67] C. Pluempitiwiriyawej, J. Moura, Y. Wu, and C. Ho, "STACS: A new active contour scheme for cardiac MR image segmentation," *IEEE Trans. Med. Imag.*, vol. 24, no. 5, pp. 593–603, May 2005.

[68] S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journ. of Computational Physics*, vol. 79, pp. 12–49, 1988.

[69] C. Xu, A. Yezzi, and J. Prince, "On the relationship between parametric and geometric active contours," vol. 1, Oct. 2000, pp. 483–489.

[70] J. A. Sethian, *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1999.

[71] V. Caselles, C. Francine, C. Tomeu, and D. Françoise, "Image segmentation techniques," *Numer. Math.*, vol. 66, no. 1, pp. 1–31, 1993.

[72] C. Pluempitiwiriyawej, "Cardiac MR image segmentation: STACS, a new active contour scheme," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, Sep. 2004.

[73] F. Jensen, *An Introduction to Bayesian Networks*, 1st ed., 1996.

[74] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[75] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *Int. Journ. Comp. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.

[76] S.-C. Chen, T. Zhao, G. Gorden, and R. Murphy, "A novel graphical model approach to segmenting cell images," in *Proc. BMES Annual Fall Meeting*, 2006.

[77] I. Daubechies, *Ten Lectures on Wavelets.* Philadelphia, PA: SIAM, 1992.

[78] S. Mallat, "Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$," *Trans. Amer. Math. Soc.*, vol. 315, pp. 69–87, Sep. 1989.

[79] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*, ser. Signal Processing. Englewood Cliffs, NJ: Prentice Hall, 1995.

[80] B. Leroy, I. Herlin, and L. Cohen, *Multi-resolution algorithms for active contour models*, ser. Lecture Notes in Control and Information Sciences, J. Sporring, M. Nielsen, L. Florack, and P. Johansen, Eds., Berlin/Heidelberg, 1996, vol. 219, proc. IEEE Int. Conf. Anal. and Opt. of Sys.

[81] S. Mallat and Z. Zhang, "Characterization of signals from multiscale edges," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 14, pp. 710–732, Jul. 1992.

[82] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 12, no. 7.

[83] A. Willsky, "Multiresolution Markov models for signal and image processing," *Proc. IEEE*, vol. 90, no. 8, pp. 1396–1458, 2002.

[84] X. Bresson, P. Vandergheynst, and J.-P. Thiran, "Multiscale active contours," *Int. Journ. Comp. Vis.*, vol. 70, no. 3, pp. 197–211, 2006.

[85] M. Unser and A. Aldroubi, "A review of wavelets in biomedical applications," *Proc. IEEE*, vol. 84, no. 4, pp. 626–638, 1996.

[86] A. F. Laine, "Wavelets in temporal and spatial processing of biomedical images," vol. 2, pp. 511–550, 2000.

[87] P. Liò, "Wavelets in bioinformatics and computational biology: State of art and perspectives," *Bioinformatics*, vol. 19, no. 1, pp. 2–9, 2003.

[88] P. Brigger and M. Unser, "Multiscale B-spline snakes for general contour detection," in *Proc. SPIE Conf. Wavelet Appl. in Signal and Image Proc.*, San Diego, CA, Jul. 1998, pp. 92–102.

[89] P. de Rivaz and N. Kingsbury, "Fast segmentation using level set curves of complex wavelet surfaces," in *Proc. IEEE Int. Conf. Image Proc.*, Vancouver, Canada, Sep. 1998.

[90] A. Grace and D. Pycock, "Multiresolution active contour models in textured-stereo images," 1996. [Online]. Available: citeseer.ist.psu.edu/279781.html

[91] L. Gui, X. Bresson, and J.-P. Thiran, "Multiscale image segmentation using active contours," Technical Report, 2005, ePFL-ITS.

[92] Y. Jin, E. Angelini, and A. Laine, *Handbook of Medical Image Analysis: Advanced Segmentation and Registration Models*. New York: Kluwer Academic Publishers, 2004, ch. Wavelets in medical image processing: De-noising, segmentation and restoration.

[93] N. Lin, W. Yu, and J. Duncan, "Combinative multi-scale level set framework for echocardiographic image segmentation," in *Int. Conf. Med. Image Comp. and Computer-Assisted Intervention*, vol. 2488, Tokyo, Japan, Sep. 2002, pp. 682–689.

[94] G. Papandreou and P. Maragos, "A fast multigrid implicit algorithm for the evolution of geodesic active contours," in *Proc. IEEE Int. Conf. Comp. Vis. and Patt. Recogn.*, 2004.

[95] P.-Y. Tsang, "Multi-resolution image segmentation using active contours," Ph.D. dissertation, University of Waterloo, 2004.

[96] H. Zhang, Z. Bian, Y. Guo, B. Fei, and M. Ye, "An efficient multiscale approach to level set evolution," in *Proc. IEEE Int. Conf. EMBS Society*, vol. 1, 2003, pp. 694–697.

[97] M. Morelock, E. Hunter, T. Moran, S. Heynen, C. Laris, M. Thieleking, M. Akong, I. Mikic, S. Callaway, R. DeLeon, A. Goodacre, D. Zacharias, and J. Price, "Statistics of assay validation in high throughput cell imaging of nuclear Factor κB nuclear translocation," *ASSAY Drug Dev. Tech.*, vol. 3, pp. 483–499, 2005.

[98] T. R. Jones, A. Carpenter, and P. Golland, "Voronoi-based segmentation of cells on image manifolds," *Lecture Notes in Computer Science*, pp. 535–543, 2005.

[99] M. Tuceryan and A. K. Jain, "Texture segmentation using voronoi polygons," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 12, no. 2, pp. 211–216, 1990.

[100] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," in *Int. Worskhop Image Proc.*, Rennes, France, Sep. 1979.

[101] S. Beucher, "The watershed algorithm applied to image trasnformation," *Scanning Micrscopy*, vol. 6, pp. 299–314, 1992.

[102] F. Meyer, "Topographic distance and watershed lines," *Signal Proc.*, vol. 38, no. 1, pp. 113–125, 1994.

[103] C. Wählby, "Algorithms for applied digital image cytometry," Ph.D. dissertation, Uppsala University, Uppsala, Sweden, 2003.

[104] A. Krtolica, C. Solorzano, S. Lockett, and J. Campisi, "Quantification of epithelial cells in coculture with fibroblasts by fluorescence image analysis," *Cytometry*, vol. 49, pp. 73–82, 2002.

[105] C. Wählby and E. Bengtsson, "Segmentation of cell nuclei in tissue by combining watersheds with gradient information," *Proc. Scandinavian Conf. on Image Anal.*, vol. 2749, pp. 408–414, 2003.

[106] A. Sarti, C. O. Solorzano, S. Lockett, and R. Malladi, "A geometric model for 3-D confocal image analysis," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 12, pp. 1600–1609, 2000.

[107] C. Solorzano, R. Malladi, S. Lelièvre, and S. Lockett, "Segmentation of nuclei and cells using membrane related protein markers," *Journ. Microscopy*, vol. 201, no. 3, pp. 404–415, 2001.

[108] D. Baggett, M. Nakaya, M. McAuliffe, T. P. Yamaguchi, and S. Lockett, "Whole cell segmentation in solid tissue sections," *Cytometry*, vol. 67A, no. 2, pp. 137–143, 2005.

[109] Z. Pincus and J. A. Theriot, "A general technique for the segmentation of individual cells in light micrographs," in *Int. Congr. Int. Soc. for Analyt. Cytology Image Anal. and Recogn.*, May 2006.

[110] B. Appleton and H. Talbot, "Globally optimal geodesic active contours," *Journ. Math. Imag. Vis.*, vol. 23, no. 1, pp. 67–86, 2005.

[111] S. Megason and S. Fraser, "Digitizing life at the level of the cell: High-performance laser-scanning microscopy and image analysis for in toto imaging of development," *Mech. of Dev.*, vol. 120, no. 11, pp. 1407–1420, 2003.

[112] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. Han, R. A. Lindquist, J. Moffat, P. Golland, and D. M. Sabatini, "CellProfiler: Image analysis software for identifying and quantifying cell phenotypes," vol. 7, 2006.

[113] E. Sifakis, C. Garcia, and G. Tziritas, "Bayesian level sets for image segmentation," *Journ. Vis. Comm. and Image Rep.*, vol. 13, pp. 44–64, 2002.

[114] Y. Boykov and V. Kolmogorov, "Computing geodesics and minimal surfaces via graph cuts," vol. 1, 2003, pp. 13–16.

[115] B. Sumengen and B. Manjunath, "Graph partitioning active contours (GPAC) for image segmentation," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 28, no. 4, pp. 509–521, 2006.

[116] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: A survey," vol. 1, no. 2, pp. 91–108, 1996.

[117] X. Chen, X. Zhou, and S. Wong, "Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 4, pp. 762–766, 2006.

[118] D. R. Padfield, J. Rittscher, T. Sebastian, N. Thomas, and B. Roysam, "Spatio-temporal cell cycle analysis using 3d level set segmentation of unstained nuclei in line scan confocal fluorescence images," in *Proc. IEEE Int. Symp. Biomed. Imaging*, Apr. 2006, pp. 1036–1039.

[119] X. Han, C. Xu, and J. Prince, "A topology preserving level set method for geometric deformable models," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 25, no. 6, pp. 755–768, Jun. 2003.

[120] T. Y. Kong and A. Rosenfeld, "Digital topology: introduction and survey," *Comp. Vis., Graph. and Image Proc.*, vol. 48, no. 3, pp. 357–393, 1989.

[121] A. Zijdenbos, B. Dawant, R. Margolin, and A. Palmer, "Morphometric analysis of white matter lesions in MR images: Method and validation," *IEEE Trans. Med. Imag.*, vol. 13, pp. 716 – 724, Dec. 1994.

[122] D. Adalsteinsson and J. A. Sethian, "The fast construction of extension velocities in level set methods," *Journal of Computational Physics*, vol. 148, pp. 2–22, 1999.

[123] J. Gomes and O. Faugeras, "Reconciling distance functions and level sets," *Vis. Comm. and Image Rep.*

[124] F. Gournay, "Velocity extension for the level-set method and multiple eigenvalues in shape optimization," *SIAM Journ. Control and Opt.*, vol. 45, no. 1, pp. 343–367, 2006.

[125] C. Xu and J. Prince, "Gradient vector flow: A new external force for snakes," in *Proc. IEEE Int. Conf. Comp. Vis. and Patt. Recogn.*, 1997.

[126] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Patt. Anal. and Mach. Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[127] N. Kovačević, N. J. Lobaugh, M. J. Bronskill, B. Levine, A. Feinstein, and S. E. Black, "A robust method for extraction and automatic segmentation of brain images," *NeuroImage*, vol. 17, pp. 1087–1100, 2002.

[128] J. R. J. Alaniz, V. M. B. nuelos, and O. Y. Suárez, "Data-driven brain MRI segmentation supported on edge confidence and a priori tissue information," *IEEE Trans. Med. Imag.*, vol. 25, no. 1, 2006.

[129] J. S. Duncan, X. Papademetris, J. Yang, M. Jackowski, X. Zeng, and L. Staib, "Geometric strategies for neuroanatomic analysis from MRI," *NeuroImage*, vol. 23, pp. S34–S45, 2004.

[130] "Segmentation and flattening toolbox, The Wandell Lab at Stanford," http://white.stanford.edu/~brian/mri/segmentUnfold.htm.

[131] J. J. Bartko, "Measurement and reliability: Statistical thinking considerations," *Schizophrenia Bullet.*, vol. 17, no. 3, pp. 483–489, 1991.

[132] G. Sundaramoorthi and A. Yezzi, "Global regularizing flows with topology preservation for active contours and polygons," Georgia Institute of Technology, Tech. Rep., Aug. 2005.

[133] T. F. Chan, B. Y. Sandberg, and L. A. Vese, "Active contours without edges for vector-valued images," *Journ. Vis. Commun. and Image Proc.*, vol. 12, no. 2, pp. 130–141, 2000.

[134] R. Gentleman and D. Lang, "Statistical analyses and reproducible research," *Bioconductor Project Working Papers*, no. 2, May 2004, http://www.bepress.com/bioconductor/paper2.

[135] J. Kovačević, "How to encourage and publish reproducible research," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, Honolulu, HI, Apr. 2007, pp. 1273–1276.

[136] A. Chebira, Y. Barbotin, C. Jackson, T. Merryman, G. Srinivasa, R. Murphy, and J. Kovačević, "A multiresolution approach to automated classification of protein subcellular location images," http://www.andrew.cmu.edu/user/jelenak/Repository/-07_ChebiraBJMSMK/07_ChebiraBJMSMK.html.

[137] "ImageJ: Image processing and analysis in Java," http://rsb.info.nih.gov/ij/.

[138] B. Parvin, Q. Yang, J. Han, H. Chang, B. Rydberg, and M. H. B.-Hoff, "Iterative voting for inference of structural saliency and characteriztion of subcellular events," *IEEE Trans. Image Proc.*, vol. 16, no. 3, pp. 615–623, 2007.

[139] F. Gibou and R. Fedkiw, "A fast hybrid k-means level set algorithm," in *Proc. Int. Conf. on Stat., Math. and Related Fields*, Nov. 2005.

[140] B. Song and T. Chan, "Fast algorithm for level set based optimization," CAM Report 2, 2002. [Online]. Available: http://citeseer.ist.psu.edu/621225.html

[141] G. Papandreou and P. Maragos, "Multigrid geometric active contour models," *IEEE Trans. Image Proc.*, vol. 16, no. 1, pp. 229–240, Jan. 2007.

[142] Y. Jin, E. Angelini, and A. Laine, *State-of-the-Art of Levelset Methods in Segmentation and Registration of Medical Imaging Modalities.* New York: Kluwer Academic Publishers, 2004.

[143] R. H. Webb and K. C. Dorey, *The pixelated image*, 2nd ed., ser. Handbook of Biological Confocal Microscopy, J. B. Pawley, Ed., 1995.

[144] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level set approach to multiphase motion," vol. 127, pp. 179–195, 1996.

[145] L. A. Vese and T. F. Chan, "A multiphase level set framework for image segmentation using the Mumford and Shah model," *Int. Journ. Comp. Vis.*, vol. 50, no. 3, pp. 271–293, 2002.

[146] L. Nirenbarg, "A strong maximum principle for parabolic equations," *Commun. Pure and Appl. Math.*, vol. 6, pp. 167–177, 1953.

[147] J. J. Koenderink, "The structure of images," *Journ. Bio. Cybern.*, vol. 50, no. 5, pp. 363–370, 1984.

[148] R. A. Hummel, *Readings in computer vision: issues, problems, principles, and paradigms*, ser. Morgan Kaufmann Readings Series archive. Morgan Kaufmann Publishers, Inc., CA, USA, 1987, ch. Representations based on zero-crossings in scale-space, pp. 753–758.

[149] V. Berinde, *Iterative Approximation of Fixed Points*, ser. Lecture Notes in Mathematics. Springer, 2007.

[150] K. Kennedy, C. Koelbel, and R. Schreiber, "Defining and measuring the productivity of programming languages," *Int. Journ. High Perf. Comp. Appl.*, vol. 18, no. 4, pp. 441–448, 2004.

[151] L. DeRose and D. Padua, "Techniques for the translation of MATLAB programs into FORTRAN 90," *ACM Trans. Prog. Lang. and Sys.*, vol. 21, no. 2, pp. 286–323, 1999.

[152] K. Li, E. D. Miller, M. Chen, T. Kanade, L. E. Weiss, and P. G. Campbell, "Computer vision tracking of stemness," in *Proc. IEEE Int. Symp. Biomed. Imaging*, May 2008.

[153] H. Thomson, "Bioprocessing of embryonic stem cells for drug discovery," *Trends in Biotechnol.*, vol. 25, pp. 224–230, 2007.