A SPLINE FITTING ALGORITHM FOR IDENTIFYING CELL FILAMENTS IN BRIGHT FIELD MICROGRAPHS

by

Jeremy Porter

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2012

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "A SPLINE FITTING ALGORITHM FOR IDENTIFYING CELL FILAMENTS IN BRIGHT FIELD MICROGRAPHS" by Jeremy Porter in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: August 16, 2012

Supervisor:

Dr. Dirk Arnold

Readers:

Dr. Norm Scrimger

Dr. Qigang Gao

DALHOUSIE UNIVERSITY

DATE: August 16, 2012

AUTHOR: Jeremy Porter

TITLE: A SPLINE FITTING ALGORITHM FOR IDENTIFYING CELL FILAMENTS IN BRIGHT FIELD MICROGRAPHS

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: M.C.Sc. CONVOCATION: October YEAR: 2012

Signature of Author

Table of Contents

List of Figures

Abstract

Bright field cellular microscopy offers an image capturing method that is both non-invasive and simple to implement. However, the resulting micrographs pose challenges for image segmentation which are compounded when the subject cells are tightly clustered or overlapping. Filamentous cyanobacteria are a type of organism that grow as linearly arranged cells forming chain-like filaments. Existing methods for bright field cell segmentation perform poorly on micrographs of these bacteria, and are incapable of identifying the filaments. Existing filament tracking methods are rudimentary, and cannot reliably account for overlapping or parallel touching filaments. We propose a new approach for identifying filaments in bright field micrographs by combining information about both filaments and cells. This information is used by an evolutionary strategy to iteratively construct a continuous spline representation that tracks the medial line of the filaments. We demonstrate that overlapping and parallel touching filaments are handled appropriately in many difficult cases.

Acknowledgements

Having discovered a newfound respect for his job as a parent, I dedicate this thesis to my father, who taught me much more than could fit within these pages.

I wish to sincerely thank Dr. Dirk Arnold for his long-suffering support, for insight and advice, and for acting as a cheerful sounding board along the way. This work certainly would not have been possible without his input and direction.

I am very grateful for my parents, siblings, and family at large, who never fail to make me happier for having them around. Thanks also to those good friends who are willing to take a hiatus while work needs doing.

Finally and most importantly, I thank my eternally supportive wife Carolyn, who possibly worked harder for this thesis than I did. And of course Owen, despite his sincere efforts to keep me from ever being finished.

Chapter 1

Introduction

The processing of images produced by optical microscopy is an active subfield of digital image processing. We wish to process images of cells, nuclei, or other microscopic organisms and extract relevant information. Common goals include counting cells, calculating physical characteristics, distinguishing between cell types, and tracking cell lineages across multiple sequential images. These all require some form of *image segmentation*, either as a precursor to image analysis or as an integral component. In general, image segmentation attempts to divide an image into regions where objects of interest are distinguished from irrelevant objects or the image background. In the context of microscopy images then, the task of image segmentation will typically involve

- separating regions of interest (ROI) from the background,
- identifying cell centres or nuclei,
- highlighting cell membranes,
- identifying separate cell areas (their nucleus and cytoplasm),

or some combination of the above.

An image processing system may be fully *automated* or *semi-automated*. An automated system only requires valid input from a human operator, and thereafter operates independently until producing its output. A semi-automated system is one which is human-assisted; it requires periodic input from an expert or operator, usually in the form of feedback on intermediate results. Fully automatic solutions are increasingly popular where the capacity for generating useful images far exceeds the capacity for analyzing and reporting data from these images, particularly in high-throughput situations [5]. For a human operator, the process of image analysis can

be tedious and repetitive, while an automatic system frees the operator for other tasks and is not prone to errors from fatigue. An operator-free approach presents a repeatable process, removing human subjectivity and providing reproducible results which can themselves be analyzed and compared reliably to alternative methods [38]. Automated processes also have the potential to be more sensitive to detail than human interpreters. Semi-automatic tools remain important where human decision-making is still desired.

1.1 Microscope image processing

Microscopy allows us to view otherwise unapproachable detail in magnified samples, but introduces problems beyond those of an ordinary photographic image. Biological samples are prepared by an operator and illuminated, and the magnified results are sampled as a digital image called a *micrograph*. New image capturing technology such as 3D and time-lapse imaging has allowed the creation of huge amounts of image data that has in turn spawned a wide variety of image processing methods and tools [49, 40]. There are numerous methods for illuminating and capturing these images, two of which are *bright field* and *fluorescene* microscopy, discussed below. A general overview of the methods of microscope image processing is presented in [61].

1.1.1 Bright field microscopy

Bright field microscopy refers to one of the simplest procedures for capturing an image: a biological sample is illuminated from below and light passes through. As the sample partially absorbs the light, the received image is one of a dark object on a bright grey or white background. Although the process for image capture is straightforward, processing bright field microscopy images poses a unique set of challenges.

Cells often appear with missing edges or edges with low contrast compared to the background intensity level. A cell's interior may contain pixels of widely varying intensity, often above the maximum (or below the minimum) intensity values outside

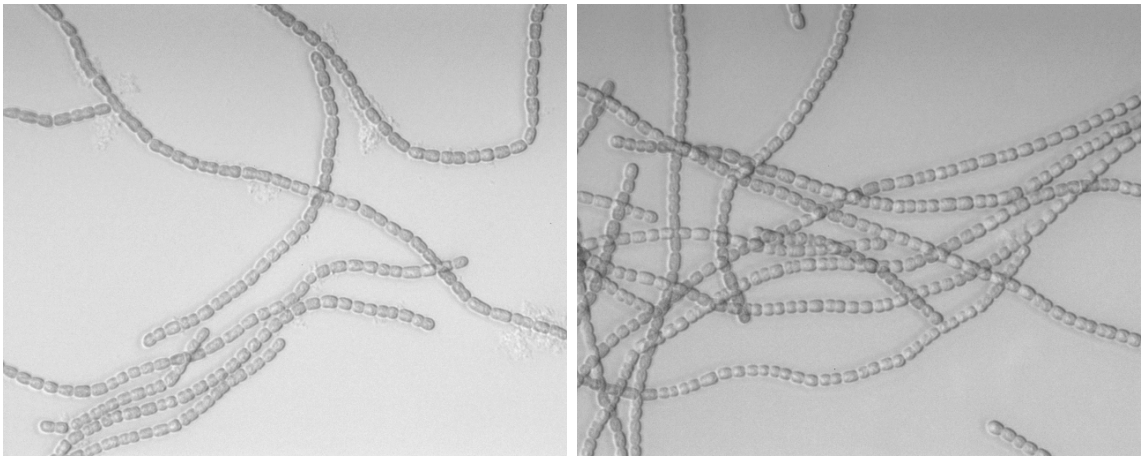


Figure 1.1: Two bright field micrographs displayed in grayscale containing filamentous cyanobacteria. The bacteria are dark objects on a bright background. Notice the uneven illumination levels, intermittent bright halos at the edges of filaments, and inconsistent intensity levels within individual cells. Image data provided courtesy of James W Golden, University of California, San Diego[2].

the cell, but sometimes containing regions with intensities similar to the image background. Due to the diffusion and reflection of light interacting with the cells, they are sometimes surrounded by a bright "halo", a dark outline, or both [25]. The halos in an image may in fact be more intense or salient than the cells themselves [56], and the cytoplasm of neighbouring cells may overlap and cause the cells to become occluded [6].

1.1.2 Fluorescence microscopy

Fluorescence microscopy requires that a sample be partially highlighted with fluorescent stains, or specially bred to genetically express fluorescent proteins for live-cell imagery. This is an extra step in preparation for an operator, and increases the difficulty of imaging live samples. The benefits are tangible however; captured images will typically have high contrast between regions of interest and the non-fluorescent background, with well-defined edges and contours for objects on the cellular scale [61].

[2]Images identified as "TL-4-13-03-adj-2" and "TL-4-6-02-adj-2"
http://biology.ucsd.edu/faculty/goldenj.html

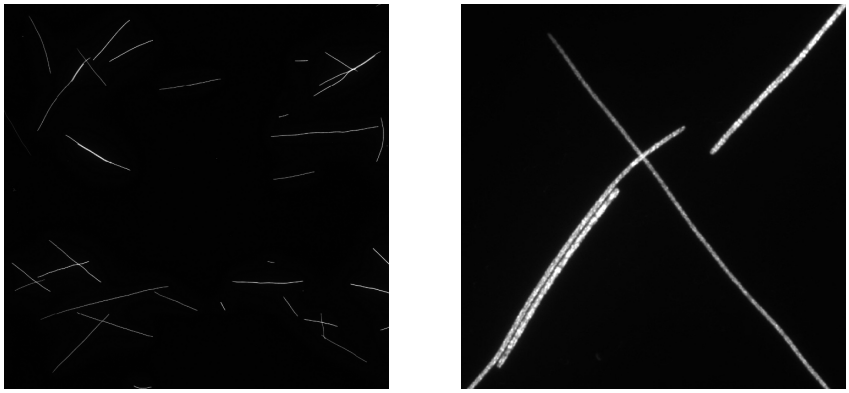
1.2 Filamentous cyanobacteria

Analysis of certain filamentous cyanobacteria is motivated by their early contributions to the evolution of life through photosynthesis [18], and more practically by their toxic impact on drinking water [16]. From an image analysis standpoint, this type of bacteria poses interesting challenges and opportunities for segmentation. To begin with, the growth of individual cells is strictly limited to growing in *filaments* or strands, with each oblong, elliptical cell joined to a cell before and after it in the approximate direction of its medial axis. Figure 1.1 shows two images of filamentous cyanobacteria, with the strands of cell filaments clearly visible.

Once a portion of an image's cells have been correctly identified, this significantly limits the areas which need to be searched for new cells, and also restricts the options for their local orientation. The filaments themselves may be of just as much interest as the individual cells, so identifying the shape and location of these filaments is another goal of image segmentation. This poses an extra challenge to any segmentation process, particularly if the sample contains multiple filaments which may overlap, lie parallel to, and partially occlude one another. In these environments it may be relatively simple to reliably identify most cells in a cluster, but identifying tightly packed filaments can be challenging even for a human interpreter.

Efforts at identifying samples of filamentous cyanobacteria have been limited, and automated solutions have thus far been restricted to filament segmentation in samples with few touching or overlapping filaments [66]. Images captured using brightfield microscopy have been analyzed using semi-automated processes [2], [1] to collect rudimentary statistics, such as cumulative and individual filament lengths. In the case of [1], very little human input was required; however, the sample input was restricted to cases with very sparse filaments, and without overlapping or parallel filaments at all.

Images captured using fluorescence microscopy were analyzed in [66], which defined the notion of a "fixel" (a portmanteau of "filament" and "pixel") to be a



(a) (b)

Figure 1.2: A large fluorescence micrograph containing numerous filaments (a), and a zoomed region of the image displaying the maximum level of detail (b). Images are part of the validation set for *Planktothrix rubescens* available from http://www.technobiology.ch/ as a public download.

fundamental object of which filaments are built. Input images were scanned for areas matching the fixel model, and matching fixels were chained together using local distance and orientation measures. The authors claim their approach accounts for parallel and overlapping filaments, yet their segmentation results demonstrated that significant progress remains to be made: of their sample data, only 86% and 55% of the overlapping and parallel filaments (respectively) were correctly identified. Further, this approach assumes fluorescence micrographs with well-defined edges and a high contrast ratio between foreground and background objects. Figure 1.2 shows sample images used in this process.

Across all of this work, the general problem of overlapping and touching parallel filaments remains unsolved. Lone filaments may be identified, but not when they are tightly clustered with other filaments. This is a common scenario in time-lapse imaging, where filaments may start out evenly separated but grow and move over time so that the filaments become entangled and closely bunched in the later images. In such a crowded image, segmentation methods for individual filaments that cannot account for overlapping and touching filaments will not succeed, so an alternative approach is required.

Our contribution consists of a first step toward automatic segmentation of cells in filamentous organisms. We propose an algorithm that is able to reliably segment filaments in crowded bright field images, and correctly distinguishes between overlapping and touching filaments. Problems with bright field imaging are overcome by combining indicators for both cells and filaments. This mutually supporting information allows segmentation to proceed even when one indicator is unreliable or missing.

Automatically determining the size, shape, and location of each cell is seen as a future goal. Automating the full process of segmentation will increase how quickly large data sets can be analyzed and removes the element of human error. Instead of manually identifying individual cells and their respective filaments as a precursor to biological experimentation or analysis, an automatic process will free human operators for other tasks and potentially make more data available to researchers. In particular, our algorithm forms the basis for an approach to identifying cells and filaments in time-lapse videos, where the amount of data often exceeds the capabilities of humans to process.

The remainder of this thesis is organized as follows. Chapter 2 includes a survey of the literature pertaining to the problem of filament segmentation, and examines the reasons why existing methods are insufficient. In Chapter 3, we present an evolutionary algorithm for matching filaments which establishes a framework for identifying structured clusters of cells. Chapter 4 demonstrates the results of using our approach and contrasts the output with those of existing methods. Finally, in Chapter 5 we draw conclusions from these comparisons and present the success of our method in the context of its alternatives, before discussing avenues for future improvements and lines of inquiry.

Chapter 2

Literature Review and Background

2.1 Image processing basics

Digital image processing is a wide and growing field of study. In keeping with the scope of this work, we will only survey some rudimentary concepts. More details can be found in [19, 61].

An image I is represented by a two-dimensional array of elements called *pixels*, each having an assigned set of values. Image pixels are typically referenced by their row and column coordinates with the origin in the top left corner; however, it simplifies notation to use the Cartesian coordinate set where each pixel has an x and y coordinate and the origin is in the bottom left corner. Each pixel contains data indicating the colour or intensity at that point in the image. In the case of a colour image, the data may be a triad of red, green, and blue components. In the case of a grayscale or "intensity" image, each pixel only has a single value which identifies the intensity or shade of gray. We will be limiting ourselves to grayscale images, in which case the image can be seen as a three-dimensional surface $z = I(x, y)$, where each discrete point (pixel) is assigned a height (intensity). We will generally use the notation $I(x, y)$ to refer to the set of pixel intensities at discrete coordinates x, y.

One of the most fundamental operations for an image is a *spatial convolution*. It is especially popular since its complexity can be greatly reduced through use of the fast Fourier transform (FFT), which is outside the scope of our interest. A discrete convolution is simply a weighted two-dimensional sum centered on a pixel, and carried out across all pixels within an image. Formally, writing $\star$ to denote the convolution operation, the discrete convolution is defined in [19] as

$$f(x, y) \star g(x, y) = \sum_{m=0}^{M-1} \sum_{n=1}^{N-1} f(m, n) \cdot g(x - m, y - n)$$

for $x = 0, 1, \ldots, M-1$ and $y = 0, 1, \ldots, N-1$. Usually, one of the functions $f(x, y)$ is taken to be the intensity map of an image, and the other $g(x, y)$ is designated by a smaller *mask*.

Although simple, spatial convolutions form the basis of numerous important image processing methods. A simple 3×3 smoothing mask, or smoothing *filter*, is given by

$$M(x, y) = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}.$$

Convolving an image with this mask sets each image pixel equal to the average intensity between itself and its eight immediate neighbours. Like all filter masks, the size is not limited to 3×3, though size is usually chosen so that the mask is symmetric about the centre pixel. An alternative smoothing mask is given by the *Gaussian filter* which is an $n \times n$ mask $M(x, y)$ defined by the Gaussian function

$$M(x, y) = G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}. \tag{2.1}$$

Here, the centre pixel of $M(x, y)$ is taken to have coordinates $(0, 0)$, and the value of σ controls the standard deviation of the Gaussian around $(0, 0)$. The effect of this filter is to blend pixel intensities from neighbouring pixels.

Edge detection can also be performed using spatial convolutions. The underlying idea is to detect pixels with a large gradient, or directional first derivative. Again treating an image $I(x, y)$ as a surface in three dimensions, the partial derivatives with respect to x and y are approximated, forming the components of a gradient vector for each pixel. The one-dimensional *gradient magnitude* image is then computed using the magnitude of each pixel's vector. There are several viable approaches for approximating gradients. The Sobel approximation uses two masks

$$M_y(x, y) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad M_x(x, y) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

to compute the partial derivatives, while the Prewitt approximation uses

$$M_y(x,y) = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad M_x(x,y) = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

In either case, the gradient magnitude image is then

$$|\nabla I(x,y)| = \sqrt{(M_x(x,y) \star I(x,y))^2 + (M_y(x,y) \star I(x,y))^2}$$

where the squaring and square root are performed as pixel-wise operations.

Another approach is to use the partial derivatives of the Gaussian function to create partial derivative masks. Since smoothing is generally required anyway as a precursor to reliable edge detection, combining the two operations makes sense. Even better, the Gaussian and derivative operations are commutative and associative when taken as convolution operations. This means that taking the partial derivative of the Gaussian and convolving with the image is equivalent to convolving the Gaussian with the partial derivative of the image. Writing I_x, I_y to denote the first partial derivatives of the image I with respect to x and y, we compute the value of

$$I_x = \frac{\partial}{\partial x} \left(G_\sigma(x,y) \star I(x,y) \right)$$

and take this to be equivalent to computing

$$I_x = \left(\frac{\partial}{\partial x} G_\sigma(x,y) \right) \star I(x,y)$$

and similarly for I_y.

2.2 Approaches to cell segmentation

There is no single approach to segmenting cells or subcellular structures from micrographs. The approach used depends on factors such as the method of image capture, the quality of the image, and the type of image feature needing to be identified. In this section, we summarize several of the more popular methods for cell segmentation focusing on those of particular relevance to our images, but this is necessarily incomplete. For a more thorough treatment of the literature, see the surveys in [49, 38, 44].

2.2.1 Watershed methods

The watershed method is a standard method in digital image processing for segmenting regions of an image based on its intensity profile [19], [61]. Given a greyscale image, pixel intensities are interpreted as height, and the image as a three-dimensional surface with peaks and valleys corresponding respectively to high and low intensity pixels. As the name suggests, a watershed algorithm then finds dividing lines between basins in the image's topography by considering a flood of water rising slowly over the surface. Beginning with its lowest points, the water levels are raised throughout the image. When the water in neighbouring basins touch, this marks a "watershed" line in the segmentation process.

Conceptually, a dam is built at these points to keep the basins' water separate while the water levels continue to rise. The flood proceeds until the entire image is covered, and the resulting set of connected watershed lines define the image's segmentation. In practice, this can be achieved by iteratively labelling pixels with intensity below an increasing threshold. The image is initialized with all pixels unlabelled, and intensity threshold T set to the minimum pixel intensity of the image. With each increment of T, all unlabelled pixels with intensity below T are inspected. Those pixels neighbouring a labelled pixel adopt their neighbour's label, and those without a labelled neighbour are assigned a new and unique label. In this way, groups of labelled pixels spread as the intensity threshold increases, until finally encountering a differently labelled group and forming a watershed line.

If applied to a reliable gradient image, the watershed algorithm will ideally identify basins corresponding to low-intensity objects and watershed lines corresponding to high-intensity edges. However, using this approach directly often leads to over-segmentation due to image noise, or from leaks between basins caused by partially missing or low contrast edges [47]. To correct for this, seeds or markers are often placed in a pre-processing step. These serve as *wells* indicating the exclusive sources of flood waters. Instead of regarding all unlabelled pixels with every increment of the intensity threshold, in this paradigm only pixels neighbouring a labelled pixel are considered as these are the only ones eligible to be flooded.

How these markers are designated is the chief variation among many watershed methods. Markers may be derived from another cell segmentation method, such as a distance transform [9],[4] or level set approach [56] which determines approximate cell centres. Other approaches use specific domain knowledge about the cell structures in the image, as in template matching [12].

2.2.2 Distance transforms

Given a binary image mask which distinguishes between foreground and background, the distance transform assigns to each foreground pixel its minimum distance to a background pixel. Any distance metric may be used, although the standard Euclidean distance is most frequently used. Finding the local maxima within an appropriately sized neighbourhood will give approximate cell centres [38], which can then be used as input for methods of refining the cell centres or finding cell contours [4]. Since it is sensitive to indentations in the foreground mask, this can be useful for cell centre extraction from closely packed or overlapping objects [61].

2.2.3 Gradient flow tracking

The gradient flow tracking model was proposed as an improved approach to defining external forces for active contour models [63] of cell segmentation. A gradient vector flow (GVF) field is formed by iteratively minimizing an energy function over the vector field formed from an image's directional gradients. The result is a vector field which matches the gradient vector field where the vector magnitudes are large, and otherwise diffuses and smooths vectors with small magnitude. This attempts to correct for noise and other small variations in local intensity. Under the assumption that an image's gradient vector field only has large vectors near prominent edges and these vectors are approximately normal to the path of the edge, the GVF field will propagate prominent edge vectors through nearby areas of homogeneity.

Although the GVF field was originally intended for finding region boundaries with

active contours, it can also be used to find cell centres. By modifying the underlying energy function to model an elastic deformation where the image gradient landscape is treated as an elastic sheet, the resulting diffused GVF field consists of smoothed vectors pointing toward the nearest cell centre [34]. Each pixel is then identified with a *sink* by tracking the flow of its diffused gradient vector to a terminal point. Pixels which flow to the same sink are grouped together as belonging to the same cell, and the sink is an estimate for the cell centre.

GVF flow tracking is reported to work well, particularly in images with tightly clustered cells, however the iterative construction of the flow field is very expensive in terms of required memory and computational time. Faster approaches have been proposed which change the physical model used to calculate the flow field, including a *vector convolution field* (VCF) [33] which treats edges as objects exerting gravitational forces proportional to their intensity, and a *feature map* [13] which abandons a physical model in favour of propagating "evidence" for edges. This last approach reports a significant improvement in computational complexity over previous VCF and GVF methods.

2.2.4 Ellipse detection

There are several viable methods for extracting ellipses from a binary image. Assuming a robust gradient image with well-defined edges, applying an ellipse detector can potentially identify cell contours and their enclosed regions. These methods assume the shape of the underlying object closely matches a true ellipse and attempt to match this directly. Other methods (such as active contours presented in Section 2.2.5) may locate shapes resembling ellipses, but there is no bias toward this shape in the approach. Two of the more popular methods for direct ellipse fitting are the Hough transform and the Fitzgibbon least-squares approach, and many approaches are based on one of these two. Completely alternate methods have also been used successfully on microscope images, including those motivated by the Mumford-Shah energy functional [65], and modified active contours [54].

Fitzgibbon least-squares method

Ellipses fall into the more general category of conic sections, which also include parabolas and hyperbolas. Any conic section in the Cartesian plane can be expressed using the implicit quadratic

$$f(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0,$$

where $a, b, c, d, e, f \in \mathbb{R}$ are the coefficients. Such a conic can be uniquely written up to a constant factor as the vector $\mathbf{a} = [a, b, c, d, e, f]$, and this vector can be said to represent a specific instance of f. This is a general equation for conics, and it represents an ellipse if and only if the discriminant of the representation $\text{disc}(f) = \text{disc}(\mathbf{a}) = b^2 - 4ac$ is negative.

Before Fitzgibbon [17], ellipse fitting required solving a least-squares problem adapted for general conics and then iteratively searching for a solution with negative discriminant. Given a set of N data points $(x_i, y_i), 1 \leq i \leq N$, the *algebraic distance* was defined as

$$D(\mathbf{a}) = \sum_{i=1}^{N} f(x_i, y_i)^2$$

and minimized using least-squares. To avoid the trivial solution $\mathbf{a} = \mathbf{0}$, extra conditions were required and an algorithm which accounted for these conditions was already known for general conics. Fitzgibbon modified the previously unusable condition

$$b^2 - 4ac < 0 \tag{2.2}$$

into the equality constraint

$$4ac - b^2 = 1 \tag{2.3}$$

which remains valid as solution vectors $\mathbf{a}$ are only unique up to a constant factor. That is, any solution $\mathbf{a} = [a, b, c, d, e, f]$ to Equation (2.2) admits the solution $\frac{1}{\text{disc}(\mathbf{a})} \cdot [a, b, c, d, e, f]$ to Equation (2.3), and similarly in the other direction. This allowed the existing non-iterative least-squares algorithm to be used for fitting exclusively against conics which were ellipses.

The Fitzgibbon method has been used successfully as a component in multi-stage image processing approaches [4, 36, 26] for cell micrographs, typically taking as input a processed binary image representing candidate cell contours. As the least-squares approach requires only location information from data points, no pixel intensity information is used directly and complementary methods are used to highlight cell outlines. The Fitzgibbon method was also improved to account for a significant number of outlier data points and occluded ellipses [11], which are significant concerns for micrographs of tightly packed or overlapping cells.

Hough transform method

The Hough transform was originally proposed as a method for detecting straight lines, but has since been generalized to potentially arbitrary shapes. Given a set of data points (x_i, y_i), the idea of the Hough transform [14] is to find the line $y = mx+b$ which best fits the given data. This is accomplished by parameterizing the line into the polar form

$$r = x\cos(\theta) + y\sin(\theta) \tag{2.4}$$

with $\theta \in [0, \pi)$, and then transforming the line into its *parameter space*. Each instance of the parameter pair (r, θ) specifies a distinct line, and the two-dimensional space of all such parameter pairs corresponds exactly with the set of all possible lines in the image space.

An accumulator array is formed by bounding and discretizing the parameter space, where each cell of the array corresponds to a parameter pair (r, θ) to within some error tolerance. For each data point (x_i, y_i), we iterate over one of the discretized axes and solve Equation (2.4) for the remaining parameter. The accumulator array is incremented in the corresponding cells, with the coordinates indicating parameters for possible lines to which the point (x_i, y_i) might belong. Points will usually cast multiple votes across the accumulator array, depending on how the parameter space is bounded and discretized. After iterating over all data points, the accumulator array is anaylzed and the local maxima found. The cell of the global maximum has received the largest number of votes, and will give the parameters

(r, θ) for the most prominent line among the given data points.

In the case of straight lines, the accumulator array is two-dimensional and it must be iterated over in one dimension. For the more complicated case of ellipses, the accumulator array is five-dimensional and quickly becomes prohibitively expensive to compute. However, by using geometric properties of an ellipse's shape, this can be reduced to a one-dimensional array [62] at the cost of increasing the iteration over N data points to an iteration over all N^2 pairs of data points. This approach was refined in [10], and made more robust for identifying incomplete ellipses.

Separate improvements were made by changing how points were mapped into the parameter space in the Randomized Hough Transform (RHT) [64]. Instead of exhaustively mapping image points into the n-dimensional parameter space, data points are randomly sampled with uniform probability. The single n-dimensional accumulator cell corresponding to the solved parameterization is incremented by one, and any accumulator cells which pass a conservative threshold are blanked and have all corresponding image points removed from the pool of voting data points. In [24], the random sampling is replaced with a genetic algorithm (GA) which evolves individuals consisting of five foreground image pixels, while the issues of voting and thresholding are also re-cast in the context of a genetic algorithm. This GA approach was used successfully for segmenting cell in micrographs, often with small clusters of overlapping cells.

2.2.5 Active contours: snakes

Active contours or "snakes" were proposed [23] as interactive parametric splines with associated energy functionals which could be iteratively minimized (or maximized) based on the underlying features of an image. If initialized close to an image feature by a user, the snake would adjust its contour to match the edges of the feature as closely as possible. Although the original snake was not, modern snakes are often *closed* contours and are used to segment a region of arbitrary shape from the full image. The term *open active contour* is used to distinguish the case where there

are endpoints (see Section 2.3.3), and these are typically used to segment distinct curvilinear features or disconnected edges.

Active contours are initialized as closed contours in the image, and deformed according to an underlying energy field. For instance, using the gradient vectors of an intensity image will attract the snake to edges with large gradient. This constitutes the *external* force on a snake's deformation, but there are also *internal* energy forces which enforce smoothness and continuity constraints on the overall contours. The total energy E of a snake is the sum $E_{\text{int}} + E_{\text{ext}}$ of the internal and external energy along the curve of the contour, so some balance is required between the two forces to achieve bounding contours which appropriately match the image feature while maintaining a desired level of smoothness.

A particular advantage of snakes is their ability to deform *actively* to account for small perturbations in the location or formation of an image's features. In other words, snakes are suitable for segmenting *dynamic* image features. A natural application is in the segmentation of video frames; once a snake has deformed to match a feature in the first frame, it is capable of slightly adjusting itself to maintain its match in the next frame.

Snakes have been used to segment cell contours in images of tightly packed cells [7] where the edges between cells are reasonably well-defined. In [55, 54], the shape of the snake contours were restricted to that of a circle or ellipse, and the energy terms adapted to include the entire covered surface instead of only the edges. This effectively allowed the snake framework to be used in detecting primitive geometric shapes, and ellipse detection (see Section 2.2.4) is of particular importance in the context of cell segmentation.

More complicated morphological topologies can be handled by discretizing the image space into grid points and using these to represent a snake as a simplicial complex [39]. In the case of a two-dimensional image, this means connecting each grid point with its eight neighbours to tessellate the plane with triangles and a snake

is defined by a subset of these triangles. In this way, changes in contour topology can be accommodated, such as events where snakes merge or split. In [67], this method was adapted to simplify computational demands by placing a further restriction on how the snakes may deform, and this was able to segment images of human arterial vessels possessing complex contours.

2.2.6 Active contours: level set methods

Level set methods were introduced [42] to solve problems posed by dynamic contour detection and image segmentation, similar to active contours (see Section 2.2.5). In fact, active contour and level set methodologies have been explicitly combined, and in some cases the terms are almost interchangeable. The most salient aspect of level set methods is that they embed contour representations in a higher-dimensional space. A summary of the modern formulation of level set methods is given in [41], and we follow their standard notation.

Given a two-dimensional area Ω with closed boundary contour $\partial\Omega$, an associated *implicit function* is a three-dimensional surface $\phi(\mathbf{x})$ which satisfies

$$\begin{cases} \phi(\mathbf{x}) > 0 & \text{if } \mathbf{x} \in \Omega^- \\ \phi(\mathbf{x}) < 0 & \text{if } \mathbf{x} \in \Omega^+ \\ \phi(\mathbf{x}) = 0 & \text{if } \mathbf{x} \in \partial\Omega \end{cases}$$

where Ω^-, Ω^+ denote the interior and exterior of Ω, respectively. In other words, the zero level set of $\phi(\mathbf{x})$ is exactly the closed contour $\partial\Omega$, while $\phi(\mathbf{x})$ is positive within the boundary and negative without. Representing the contour in this higher-dimensional setting gives access to more powerful methods for effecting topological changes for dynamic curves. For example, it is a non-trivial problem to compute the union or intersection of two dynamic contours if specified by snakes (as in Section 2.2.5).

Following the introductory example of [41], a simple analogy of this higher-dimensional representation is to consider the points ± 1 on the real number line $\mathbb{R}$.

An explicit representation would be to simply write $\{-1, +1\}$ and have this define the one-dimensional area $[-1, +1]$, with boundary $\partial\Omega = \{x \,\big|\, |x| = 1\}$. An implicit representation would be defining $\phi(x) = -x^2 + 1$, since this satisfies

$$\begin{cases} \phi(x) > 0 & \text{if } x \in (-1, 1) \\ \phi(x) < 0 & \text{if } x(-\infty, 1) \cup (1, \infty) \\ \phi(x) = 0 & \text{if } |x| = 1. \end{cases}$$

We could also define $\phi(x) = -(x^3 + 1)(x^2 - 1)$ as the implicit function, since it too satisfies the above inequalities. The selection of the implicit function is not explicitly determined by the boundary, but is instead chosen to be as smooth as possible, avoiding any very large or very small gradients.

One such function is based on the *signed distance function*. Given a closed boundary $\partial\Omega$, we define

$$d(\mathbf{x}) = \min_{\mathbf{x}_I \in \partial\Omega} (|\mathbf{x} - \mathbf{x}_I|)$$

to be the signed distance function, which is the distance from any point $\mathbf{x}$ to the closest point lying on the boundary $\partial\Omega$. We then define an implicit function $\phi(\mathbf{x})$ for any closed boundary $\partial\Omega$ to be

$$\phi(\mathbf{x}) = \begin{cases} -d(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega^- \\ d(\mathbf{x}) & \text{otherwise} \end{cases}$$

since clearly $d(\mathbf{x}) = 0$ if $\mathbf{x}$ lies on the boundary, and it is positive inside the boundary while being negative outside.

Although this simplifies certain topological operations, deforming the implicit function of a bounded contour still requires an energy field. This can be defined as the gradient field of an image, so that the level set contour will be attracted to strong edges, but there are other possibilities. One popular option involves the Mumford-Shah energy model, which conceptually splits an image into two regions with approximate pixel intensities u_1, u_2. These intensities are given as input, and the Mumford-Shah problem is to find the boundary which splits the image into two regions using a closed boundary $\partial\Omega$ such that the approximate intensity in Ω^+ matches

u_1 and the approximate intensity in Ω^- matches u_2, both as closely as possible.

The Mumford-Shah level set model can be extended to account for more than two regions and also handle more complex topologies [57], and this has been used to segment single cells in bright field images in combination with a watershed [56] or alternative method [6]. Prior knowledge about the features in an image can also be used through a model-based approach to constrain the deformation of level sets [43].

2.2.7 Notch detection for overlapping cells

Notch detection is a straightforward method used for detecting points of overlap between two or more cells [4, 26]. Given an image with reliable contours and orientation information, normal vectors are computed for all edge pixels. Walking along an edge contour, the angle θ of the normal vector at each pixel is recorded in sequence. The angles are smoothed to account for noise and allow subtle changes to be ignored, then all changes in the concavity of the contour above some threshold are recorded as possible notches. Using information about the centre of each cell, notches can be paired off where applicable. If the approximate shape of the cells is also known, as in the case of elliptical cells, then implicit contours can be inferred where occlusion would otherwise prevent an edge contour being detected.

2.3 Approaches to filament segmentation

Line tracing and curve detection are largely synonymous and refer to the segmentation of curvilinear paths from an image. In some cases such as filament segmentation, distinct lines must be individually identified and separated. There are numerous approaches to solving this problem in both 2D and 3D images [32], but we will focus in this section on three particular approaches and highlight their potential applicability to filament segmentation.

2.3.1 Spline fitting to point clouds

Fitting parametric curves to a set of 2D points is a long-standing and popular problem in computer vision. It is assumed that the set of points approximates a shape or curve in the plane, and the problem is to derive a curve which matches the underlying shape. This becomes more difficult when the set of points is unordered [20], as is the case with points given by pixel coordinates in an image. One approach is to use an error fitting term to measure how well a potential curve matches the underlying data, and then optimizing the curve using this measure. A standard error term is the least-squares model, although there are others [59].

In [37], a Euclidean distance minimum spanning tree (EMST) is used to group points locally, and a parametric B-spline curve is then initialized at random within the point cloud. A small set of points is matched by the curve using a least-squares distance error. The local tangent information of the curve is then used to look past the curve's endpoints in a small window, and together with the EMST topology this is used to select new points to be iteratively fitted by the curve. New points continue to be added using the tangent window and EMST until no remaining un-matched points are detected. The resulting B-spline curve is refined slightly, and then output as a matching curve to the shape of the point cloud.

Unfortunately, matching multiple distinct shapes within a single point cloud is not explicitly handled. There is also the strong assumption of homogeneity in the point cloud; if the points are not uniformly distributed across the underlying shape, or if there are numerous outliers, the algorithm's assumptions are violated.

2.3.2 Tracing and detection of curvilinear features

In [52, 53], Steger describes a low-level approach to identifying curvilinear structures which extracts lines with subpixel accuracy and simultaneously estimates their width. It was introduced as a method for identifying roads in aerial photographs [31], and has also been used in identifying coronal loops in solar photographs [15, 3]. An input gray level intensity image $I(x, y)$ is modeled as a three-dimensional continuous

surface $z(x, y)$. Ideally, the perpendicular cross-section of a bright line in $z(x, y)$ is expected to resemble an inverted parabola, with smooth, steep sides and a local maximum where the first derivative vanishes, marking the middle point of the line. In practice, a bright line cross-section may also fit either of the one-dimensional profiles

$$f_b(u) = \begin{cases} h & \text{if } |u| \leq w \\ 0 & \text{if } |u| > w \end{cases}$$

$$f_a(u) = \begin{cases} 0 & \text{if } u < -w \\ h_1 & \text{if } |u| \leq w \\ h_2 & \text{if } u > w \end{cases}$$

where $h_2 < h_1$, and the parameter u is centered on the line and varies in both perpendicular directions. These are the *bar profile* and *asymmetric bar profile*, respectively.

With an inverted parabola model, looking at cross sections $f(u)$ should lead to the ridge of a line being identified as the point u where $f'(u) = 0$ and $f''(u)$ is large. There are no similar conditions for the discontinuous bar profiles; however, convolving the image with Gaussian kernels smooths the responses sufficiently for these conditions on $f'(u), f''(u)$ to hold. The two-dimensional Gaussian kernel $G_\sigma(x, y)$ defined in Equation (2.1) and its partial derivatives are approximated for an image by using discrete values of x and y. Since the Gaussian and differential operators are commutative and associative as convolution operations, we can convolve the above partial derivatives with the Gaussian kernel, and use this as an approximation for the first and second order spatial derivatives of the smoothed image.

Calculating these derivatives for each pixel, useful information can be obtained regarding the location of possible lines. Following the line profile model, we want pixels which have a second directional derivative with large magnitude and a first directional derivative which vanishes along that vector. Writing the partial derivatives of I as $I_x, I_y, I_{xy}, I_{xx}, I_{yy}$, the direction possessing the second directional derivative of maximum absolute value can be found by considering the eigenvectors of the Hessian

matrix

$$H = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{pmatrix}.$$

The eigenvalue e of largest absolute value and its eigenvector $\mathbf{v}_e$ correspond to the magnitude and direction of the second directional derivative, which points in the direction normal to the candidate line. A minimum threshold may be used to discard pixels without a sufficiently large second directional derivative.

A line pixel must also have a vanishing first derivative in the direction of $\mathbf{v}_e$. Since the image is discrete, a Taylor polynomial approximation of the image surface $z(x, y)$ is used to estimate the first derivative with subpixel accuracy. If the first directional derivative vanishes along $\mathbf{v}_e$ within the same pixel, then it is marked as a line pixel.

The result is a binary mask $L(x, y)$ which indicates line pixels, where each line has only a single response. That is, a line in the original image will have a corresponding sequence of line pixels in the binary mask approximating the spine or "ridge" of the curve, and the binary line will be only one pixel wide. This is attractive for any post-processing steps, as multi-pixel responses would require thinning or other corrective operations to be invoked. In particular, this aids in the process of linking line pixels to group them together as belonging to the same curve.

Linking curve pixels

The linking process considers in sequence each non-zero pixel of $L(x, y)$. Let $\mathbf{n}(x, y) = \mathbf{v}_e$ and $\mathbf{n}^{\perp}(x, y)$ respectively denote the normal and tangent directions of the line to be detected. At each pixel, the neighbouring 8-connected region is restricted to three pixels which are approximately in the direction of $\pm\mathbf{n}^{\perp}$. From these, the neighbour which minimizes the sum $d + \beta$ is chosen to continue the line, where $d \in (0, \sqrt{8})$ is the Euclidean distance between the subpixel locations of the line in the adjacent pixels, and $\beta \in [0, \frac{\pi}{2}]$ is the difference between the angles representing $\mathbf{n}^{\perp}$ in each pixel. New pixels continue to be added as long as they exceed a saliency threshold,

which is measured in the magnitude of the vector $\mathbf{n}$.

The linking process was improved by Raghupathy [45, 46] by augmenting the purely local approach of the Steger algorithm with global information. The two important issues specifically addressed were the behaviour of curve following at junctions or intersections, and following of faint curves. Junctions are handled by allowing the linking algorithm to look ahead farther than one pixel at a time in the approximate direction of $\mathbf{n}^{\perp}$ when no suitable matches are found in the immediate neighbourhood. To correct for noise, the average orientation

$$\mathbf{n}^{\perp}_{\text{av}} = \frac{1}{M} \sum_{i=0}^{M} \mathbf{n}^{\perp}_{\mathbf{p}_{k-i}}$$

of the line tangents is computed over the current pixel $\mathbf{p}_k$ and the previous M pixels. A distant pixel $\mathbf{p}_{k+1}$ is linked with the current pixel $\mathbf{p}_k$ if it minimizes the angle difference between $\mathbf{n}^{\perp}_{\mathbf{p}_{k+1}}$ and $\mathbf{n}^{\perp}_{\text{av}}$.

Curves which fade to termination or end then reappear are handled by application of the Radon transform. When the usual linking algorithm signals the termination of a line, the Radon transform is applied to the image to detect any faint lines. The inner product is then computed between the current line and those detected by the Radon transform, and line segments having an endpoint matching that of the terminated curve and maximizing the value of their inner product are selected as faint continuations of the line.

If this fails to detect a fading line segment and the current line does terminate, the possibility still exists of the line re-emerging. This is detected by a combination of the previous two methods. The same Radon transform result is used, but the search considers distant endpoints instead of only the terminating line's current endpoint. Distant endpoints are computed by following the average orientation $\mathbf{n}^{\perp}_{\text{av}}$ from the terminating line's endpoint, and searching in this neighbourhood for line segments from the Radon transform which admit a large inner product. To reduce erroneous line detection, re-emergent line segments are also required to match against the magnitude of their normal vectors.

Scale selection

Selecting the correct value for σ directly impacts the ability of the algorithm to detect lines; larger values are needed for wider lines, but values which are too large will tend to remove critical details. Steger requires that $\sigma \geq \frac{\sqrt{3}w}{6}$, and implies that $\sigma \leq \frac{w}{2}$ should also hold. The selection of this parameter involves analyzing the image in *scale space*, which is the result of smoothing the image using G_σ with varying scale σ. In [60], the authors tackle the issue of scale selection in a restricted case of line tracing. With the assumption that a given line passes through a fixed row or column in the image, a scale-space search is only necessary over this set of entries. A line sweep then proceeds in the directions tangent to the image curve, and if necessary adjusts the value of σ in small increments between successive rows/columns.

This has the benefit of allowing for robust detection of lines with varying width, and significantly decreasing the overhead for determining a single appropriate σ. Unfortunately, the approach is only valid when the line is guaranteed to pass through a known row or column of the image space.

2.3.3 Open active contours: snakes

Open active contours (OAC) or "snakes" were proposed [23] as parametric splines with associated energy functionals which could be iteratively minimized (or maximized) based on the underlying features of an image. Active contours are traditionally defined using a closed contour (see Section 2.2.5) which has no endpoints, and deflates or inflates in a natural way to minimize its energy. To distinguish from these, *open* active contours are those which have specified endpoints. These may be fixed in space or allowed to deform along with the snake, subject to some restrictions [50].

A snake energy functional is composed of both *internal* and *external* forces. Internal forces are responsible for explicitly enforcing the piecewise continuity of the curve, encouraging a smaller size, and discouraging curvature. The external forces

guide the snake toward a desired image feature, allowing it to overwhelm the influence of the internal forces where necessary. That is, an image feature which is both large and curved should be matched with an equivalently large and curved snake, but one that is no larger or more curved than it need be. An important deviation from using closed contours, the snake external forces also encourage stretching the contour's endpoints. Given an intensity image and an appropriate starting position, a snake will be drawn toward the nearest bright (or dark) object in the image and match its full contour.

We focus on one particular implementation of OACs, however there are numerous other approaches in the machine vision literature, surveyed in [50]. The recent formulation in [51, 35] is of most interest, as it is intended specifically for tracking filaments in micrographs and has the added benefit of being implemented in the freely available software package *JFilament* [1]. Here the total energy E of an OAC is written as $E = E_{\text{int}} + E_{\text{ext}}$ where E_{int} is the internal and E_{ext} the external energy. These are defined along the parametric curve $\mathbf{r}(s) = (x(s), y(s)),\ s \in [0, L]$ which defines the OAC. For each value of $s \in [0, L]$, the function $\mathbf{r}(s)$ evaluates to a coordinate pair in an underlying image space.

The internal energy is computed as

$$E_{\text{int}} = \int_0^L \left(\alpha(s) \left\| \mathbf{r}_s(s) \right\|^2 + \beta(s) \left\| \mathbf{r}_{ss}(s) \right\|^2 \right) \mathrm{d}s \tag{2.5}$$

where

$$\mathbf{r}_s(s) = \frac{\mathrm{d}\mathbf{r}}{\mathrm{d}s} = \left(\frac{\mathrm{d}(x(s))}{\mathrm{d}s}, \frac{\mathrm{d}(y(s))}{\mathrm{d}s} \right)$$
$$\mathbf{r}_{ss}(s) = \frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}s^2} = \left(\frac{\mathrm{d}^2(x(s))}{\mathrm{d}s^2}, \frac{\mathrm{d}^2(y(s))}{\mathrm{d}s^2} \right)$$

are the first and second order spatial derivatives, respectively. As we are minimizing the expression for E_{int}, smaller values for these derivatives in the image space are

[1] JFilament software is available as an ImageJ plugin at
http://athena.physics.lehigh.edu/jfilament/

encouraged along the parametric curve. Thus, the term $\|\mathbf{r}_s(s)\|^2$ serves to restrict the stretching and the term $\|\mathbf{r}_{ss}(s)\|^2$ restricts the bending of the snake, while $\alpha(s)$ and $\beta(s)$ weight their relative importance.

The external energy is computed as

$$E_{\text{ext}} = k \cdot \int_0^L \left(E_{\text{img}}(\mathbf{r}(s)) + k_{\text{str}} \cdot E_{\text{str}}(\mathbf{r}(s))\right) \mathrm{d}s \tag{2.6}$$

where E_{img} denotes energy contributed by features in the underlying image, E_{str} encourages stretching, k_{str} is a constant which balances stretching against image energy, and k is a constant which allows balancing E_{ext} against E_{int} in the total energy equation. The energy term for image features is defined as $E_{\text{img}} = G_\sigma \star I$, where G_σ is the Gaussian smoothing kernel with standard deviation σ, I is the underlying image, and $\star$ denotes the convolution operation. This is chosen because the gradient $\nabla G_\sigma \star I$ tends to point toward the middle of filaments, which the energy minimization process exploits. The energy term $E_{\text{img}}(\mathbf{r}(s)) = G_\sigma \star I(\mathbf{r}(s))$ is then intensity of the convolved image under the curve $\mathbf{r}(s)$.

The energy term for stretching E_{str} is based on a re-scaling of the intensity image, but only defined on $I(\mathbf{r}(s))$ where $s = 0, L$, which are the endpoints of the snake. The force

$$F(\mathbf{r}(s)) = \frac{I(\mathbf{r}(s)) - \bar{I}}{I_{\text{f}} - I_{\text{b}}}$$

defines the magnitude, where $\bar{I}$ is the image's mean pixel intensity, I_{f} is the mean foreground intensity, and I_{b} is the mean background intensity. These are constant values, and must be estimated before beginning the snake deformation. This defines the magnitude of the stretching factors, while the direction is tangent to the curve up to a sign, pointing away from the filament, so

$$\nabla E_{\text{str}}(\mathbf{r}(s)) = \begin{cases} \frac{-\mathbf{r}(s)}{\|\mathbf{r}(s)\|} \cdot F(\mathbf{r}(s)) & \text{if } s = 0 \\ \frac{\mathbf{r}(s)}{\|\mathbf{r}(s)\|} \cdot F(\mathbf{r}(s)) & \text{if } s = 1 \\ 0 & \text{otherwise.} \end{cases}$$

As the underlying image is fundamentally discrete, the value of E can be approximated by sampling the pixel intensities along the curve $\mathbf{r}(s)$ and taking a finite sum. Given N unique pixels $s_i = (x_i, y_i)$ under the curve $\mathbf{r}(s)$, we can write

$$\tilde{E}_{\text{int}} = \sum_{i=1}^{N} \left(\alpha(s_i) \left\| \mathbf{r}_s(s_i) \right\|^2 + \beta(s_i) \left\| \mathbf{r}_{ss}(s_i) \right\|^2 \right)$$

$$\tilde{E}_{\text{ext}} = k \cdot \sum_{i=1}^{N} \left(E_{\text{img}}(\mathbf{r}(s_i)) + k_{\text{str}} \cdot E_{\text{str}}(\mathbf{r}(s_i)) \right),$$

and therefore

$$E \approx \tilde{E}_{\text{int}} + \tilde{E}_{\text{ext}}.$$

From this, the spatial derivatives can be computed to guide the deformation of the snake.

This formulation of OAC snakes was used successfully to identify and track filaments in fluorescence micrographs [35, 51], and is amenable to time-lapse image analysis because of the incremental deformation model of snakes. As each frame is expected to have only slight differences from the previous frame, an existing snake can naturally adapt to the new information. However, the single-frame process was only applied to sparse images with few total filaments, there are limited methods for handling overlapping or parallel filaments, and there are no obvious adaptations for bright field images. These snakes use splines only implicitly, by enforcing continuity and differential constraints directly through their internal energy functions. Even though not specified in the general approach, the available *JFilament* implementation uses very short segments by default, effectively assigning one control point for every pixel lying on the spline. This reduces the snake representation to a pixel sequence which does not take advantage of the inherent nature of spline based interpolation.

Chapter 3

An Evolutionary Spline Matching Algorithm

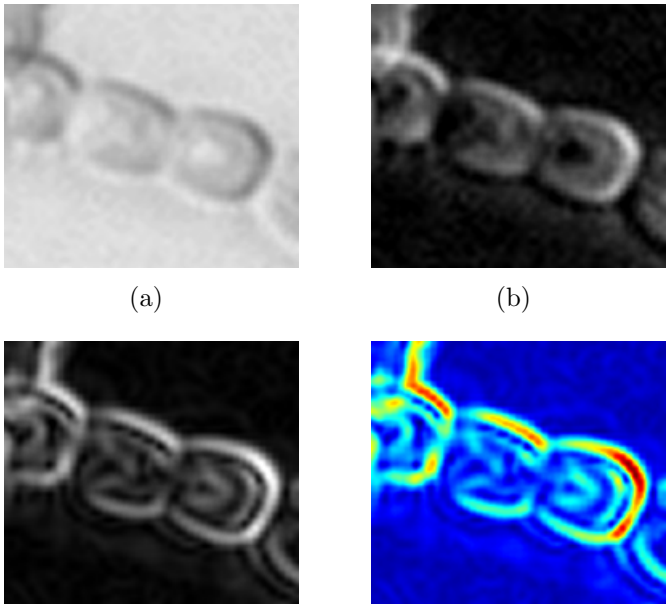
We propose a new approach for segmenting filaments in micrograph images which combines information about both filaments and cells to iteratively construct an outline of the filaments with an evolutionary strategy (ES). Finding an outline implicitly defines a midpoint line for the filament and all its cells, and cell segmentation from these cell centre markers is then left as a task for existing methods. Control points for a Bessel-Overhauser spline are added in sequence along a filament, where each new spline segment is scored using an objective function. Using a simple version of the covariance matrix adaptation evolution strategy (CMA-ES), a candidate spline segment is evolved and appended to the filament's spline.

3.1 Motivation and lead-up to solution

An obvious approach for segmenting images of filamentous cyanobacteria is to rely on existing methods (see Section 2.2) for segmenting cells in bright field micrographs. Methods which are able to segment tightly clustered cells should be of particular applicability in the context of crowded filament cells. However, for our purposes there are several potential drawbacks to this approach.

Methods for the segmentation of clustered cells are best used on images with clearly defined cell contours. This will generally be true for images captured using fluorescence microscopy [34, 8, 58] although bright field images with consistently strong edges may also be suitable [7, 30]. In our image sets edges are often faint or missing entirely, which means any purely edge-based approach will undersegment the image. Figures 3.1 and 3.2 show examples of cells with partially and fully missing edges. Individual cells with missing or irregular edges have been reliably extracted [6, 56] from bright field images; however, such edge-free approaches have not yet

been used on images of multiple or tightly clustered cells.



(a) (b)

(c) (d)

Figure 3.1: Four views of a cell with a partially missing edge: (a) original input image, (b) negativized image after illumination correction, (c) gradient magnitude image, (d) false colour of gradient magnitude image.

There is no accepted method for reliably segmenting cells in bright field images with poor contrast, non-homogeneous intensity levels between foreground and background, and missing edges. Further, such a method would still not be able to communicate any topological information in the case of filamentous organisms. Not only would we like to know the location of each cell's membrane and approximate centre, but also its parent filament, neighbouring cells, and orientation within its filament. Additionally, the overall path, orientation, and length of each filament should be identified within the image. This information is not available purely through cell segmentation.

The alternative approach is then to consider segmenting filaments first. With reliable filament information, the relative positions and orientations of cells are strictly limited within the image space. This constrains any search to a very narrow window,

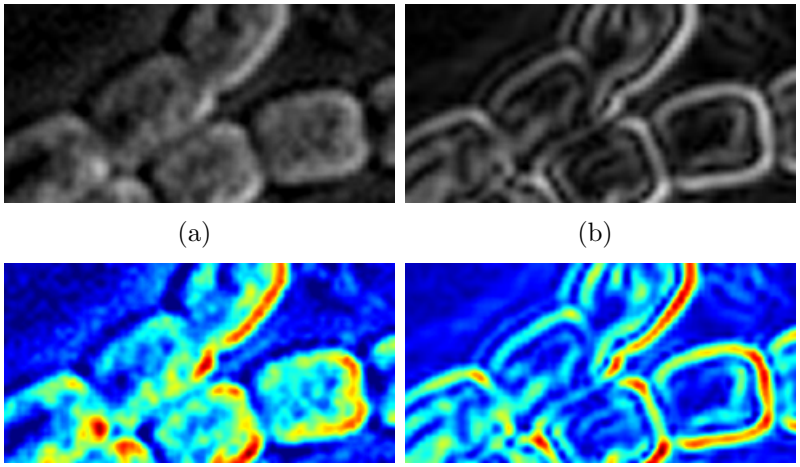


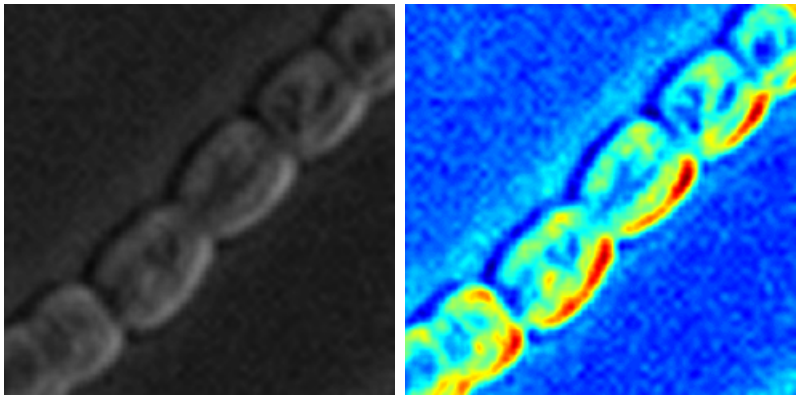
Figure 3.2: Four views of touching cells with missing edge: (a) negativized image after illumination correction, (b) gradient magnitude image, (c) false colour of negativized image in (a), (d) false colour of gradient magnitude image in (b).

greatly reducing the impact of missing and inconsistent cell features, and possibly making existing cell segmentation methods viable.

With proper thresholding, the Steger-Raghupathy algorithm (see Section 2.3.2) is reasonably robust and theoretically attractive. The differential geometric approach has been used successfully in segmenting micrographs [48], and the Steger-Raghupathy code has been applied to processing astronomical images [15, 3]. However, several factors make it unsuitable for direct application to the problem of segmenting filaments in bright field micrographs.

The contrast between filaments and the image background is generally low, so that the overall intensity of a filament is unreliable. In Figure 3.3, a typical "coffee bean" intensity profile is shown for several cells. Each cell has internal regions of intensity comparable to the image background. As the Steger-Raghupathy approach relies on the relative intensity to determine the proper width of each line, this kind of intensity pattern will mean incorrect reporting of line ridges and widths. In spite of this, the approach does give reasonable approximations to filament ridges for some isolated filaments after sufficient smoothing. Intersections between nearly orthogonal

filaments are handled adequately, though a robust linking method would be required. However, intersections between filaments at sharp angles, non-intersections between parallel touching filaments, and groups of multiple tightly clustered filaments are all instances in which the Steger-Raghupathy method fails.



(a) (b)

Figure 3.3: Two views of cells showing low contrast between cell centres and background: (a) negativized original image, (b) false colour of negativized image.

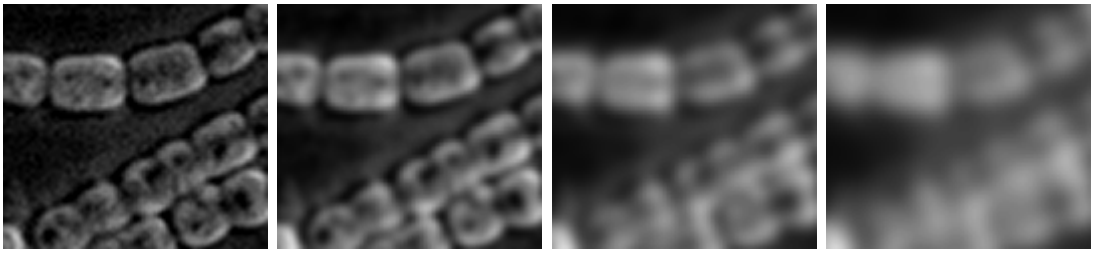
More accurately, it is the input images which fail the assumptions of the Steger-Raghupathy algorithm. The filaments in our images have inconsistent intensities and appear to have inconsistent widths. Where two filaments lie in parallel, the smoothing factor blurs together the intensities of neighbouring cells so that the reported line ridge is actually closer to the edge dividing the two filaments. This is an expected and even correct response from the algorithm, but it is not the response we desire. This effect is compounded in areas of multiple clustered filaments, and it is not clear whether Steger's pixel responses could be correctly linked in these cases, even if they could be accurately determined.

Open active contours which conform to the ridge of a line (see Section 2.3.3) suffer from similar problems, and may be even more sensitive to the uneven intensities within a filament. In particular, cells with low intensity interiors will disrupt the progress of a snake as it deforms along a filament. If instead of ridges the snake is designed to be attracted to the large gradients of edges, then faded and missing

edges as well as the bright field halos still prevent open active contours from being an attractive approach. Snakes are not inherently able to move past difficult areas of low or uneven intensity, since by definition they seek the path of least resistance. As they rely on strictly local information from the image, they cannot explore distant options when extending along a filament which might require overlooking small areas of difficulty.

Open snakes also lack any method for distinguishing between filaments that present equally viable paths of deformation, as in the case of parallel or tightly clustered filaments, and filaments intersecting at a sharp angle. Whether a filament should be followed in one direction or another can depend on non-local indicators or varying local features that may or may not be present. Part of this problem lies in the smoothing pre-processing step which minimizes the impact of noise and allows for a local search. In our image sets, the granularity of certain features like edges between cells on parallel touching filaments, is small enough that any effective noise-removal will also degrade the features. In these regions, the blurring effect will render any search over the local region unreliable. In Figure 3.4, the effect of smoothing is presented on a section of image with touching cells. Even with a conservative smoothing parameter of $\sigma = 1.0$, some details begin to disappear. More aggressive smoothing renders the borders between cells completely unrecognizable. For detecting these filaments, the Steger algorithm advocates using a smoothing parameter in the approximate range $3.75 \leq \sigma \leq 6.5$. Snakes also perform very poorly with larger values of σ, yet without this level of smoothing they invariably become stuck in local energy minima due to intensity inconsistencies in the interior of cells. Results for both approaches are presented in more detail in Section 4.1.

These problems suggest a third alternative which will form the core of our approach. Rather than segmenting cells and combing them to segment filaments, or segmenting filaments and splitting them to segment cells, we will simultaneously combine image features for both filaments and cells to generate filament outlines. These will be used in a complementary way, and accommodate weak or incomplete



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

Figure 3.4: Visualizing the effects of smoothing on tightly packed cells. Three filaments are seen lying in close proximity, with two lying in parallel and touching. In order left to right, (a) the original image is displayed, followed by smoothing results with increasing values of (b) $\sigma = 1.0$, (c) $\sigma = 2.0$, and (d) $\sigma = 3.0$.

features in the underlying image. This moves away from the exclusively local approaches of snakes and line tracing, and incorporates more global information into the objective. Especially in areas where features are indistinguishable on the level of individual cells, global information about the parent filaments can be used to infer a correct segmentation. Splines are used to represent segmented filaments, which implicitly provide constraints on continuity and rigidity of the representation, rather than needing to provide this balance explicitly through an internal energy function. To avoid becoming stuck on local optima we employ an evolutionary strategy rather than a gradient descent method, to effect the growth and deformation of filament outlines. This has the added advantage of allowing for dynamic distance between spline control points. Rather than specifying a constant length, the distance between successive control points is permitted to grow or shrink according to the complexity of the image features.

3.2 Evolving a single filament outline

The core function of our algorithm is to locate a ridge which traces the midpoint line of an entire filament, as well as parallel edge lines tracing the contour of the filament. A bright field image and the approximate width w of the filament cells are taken as input, and a starting point for a filament can be provided manually or initialized automatically. The output for each filament is a single spline curve which indicates

the median ridge of the filament. Two parallel edge splines are implicitly defined at distance $\frac{w}{2}$ from the ridge spline, and these indicate the exterior filament edges.

The placement of a new control point is decided based on maximizing the objective function

$$O(S) = E_{\text{edge}} + E_{\text{ridge}} + E_{\text{length}} \tag{3.1}$$

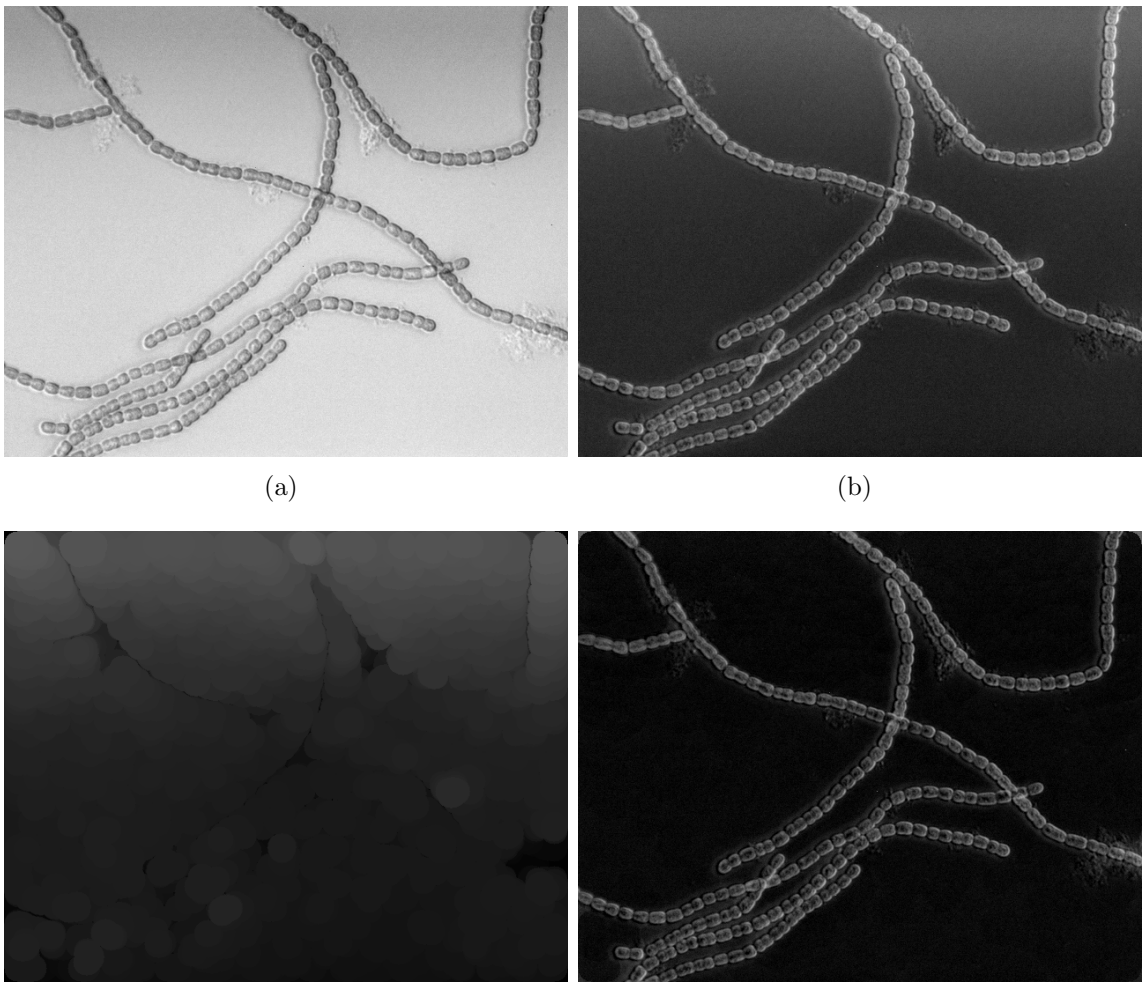
which is the sum of three components that reward the spline for highlighting edges of cells, the ridge of a filament, and overall length. The edge and ridge scores are both calculated by using processed versions of the input image. The length score depends only on the spline, and will be discussed in Section 3.2.3. The edge image I_e and ridge image I_r can be any images which highlight cell edges and filament ridges respectively, and different pre-processing methods may be more appropriate for different images. Our original image set contains dark filaments on a bright background. This is reversed for ease of visualization by taking the image negative.

The optimization of this score is handled by an evolutionary strategy. A new control point is initialized at a short distance in the direction of the approximate end point tangent of the previous segment. The placement of the control point is then explored stochastically, and scored by examining the spline segments which result. The best options are combined at each generation, and gradually the process converges to a solution. We have chosen an evolutionary strategy with an available implementation which automatically scales its step size between generations.

3.2.1 Image pre-processing

The illumination levels of our image sets were observed to be very uneven, so rudimentary illumination correction was applied. The morphological top-hat operator computes the difference between an input grayscale image and its morphological opening, which serves as an estimate for the illumination level in a local area. A structural element is required which is larger than the details of interest in the image; we used a disc element of radius 15. Figure 3.5 demonstrates the progress,

moving from an input image with dark features on a bright background, to a negative image with more even illumination.



(a) (b)

(c) (d)

Figure 3.5: Visualizing the illumination correction process. The original image (a) is first converted to its negative (b), then the morphological opening is computed (c). The difference between the negative image and its opening are the result (d).

Each edge and ridge image is generated by using this illumination corrected image as input. We present several options for computing the edge and ridge images here, and choose the best result for our image set.

A naive method for generating the edge image I_e is to convolve the input image

I with the Gaussian derivative and take the magnitude of the gradient, as in

$$I_e = \|\nabla G_\sigma \star I(x, y)\| .$$

This will likely give unsatisfactory results, as bright field images can have very large intensity changes within individual cells as well as at their edges, producing considerable noise in a gradient magnitude image. The Steger line tracing algorithm can be used with small smoothing parameter to extract prominent curvilinear edges from a gradient magnitude image, creating a binary mask outlining the filament edges. An alternative to the Gaussian derivative method is to use a phase congruency edge detector [28, 29], which is notably robust against poor image contrast and varying levels of illumination. Feature detection based on phase congruency seeks to highlight areas of an image where the Fourier waveforms are in phase [1]. As a simple analogy, Figure 3.6 contains several sine waves of varying amplitudes, wavelengths, and phase offsets which are completely in phase at several points. At these locations, the sum of the sine waves experiences a significant change, which would likely be perceived as a salient edge in an associated image [27]. The response from a phase congruency edge detector is always in the range $[0, 1]$ so no further scaling is required.

Based on our image set, the Gaussian derivative method on its own gave only adequate results for generating useful edge data. The large gradients within most cells resulted in significant noise in the edge data. By contrast, the phase congruency edge detector produced much cleaner images with very little noise. In both cases, uneven edges were still apparent in areas where a filament had high contrast on one side and low contrast on the other. This is unattractive as we would ideally like all true edges to be weighted evenly.

The Steger approach discussed in Section 2.3.2 can be used without pixel linking to highlight edges that are part of a curvilinear structure. This algorithm can be run on the phase congruency edge image, using suitable parameters for locating lines one to two pixels wide. The result is a binary mask of all prominent curvilinear edges. If the phase congruency image's pixel intensities are mapped to real values

[1] The implementation used for the phase congruency edge detector is available as a software download from http://www.csse.uwa.edu.au/~pk/

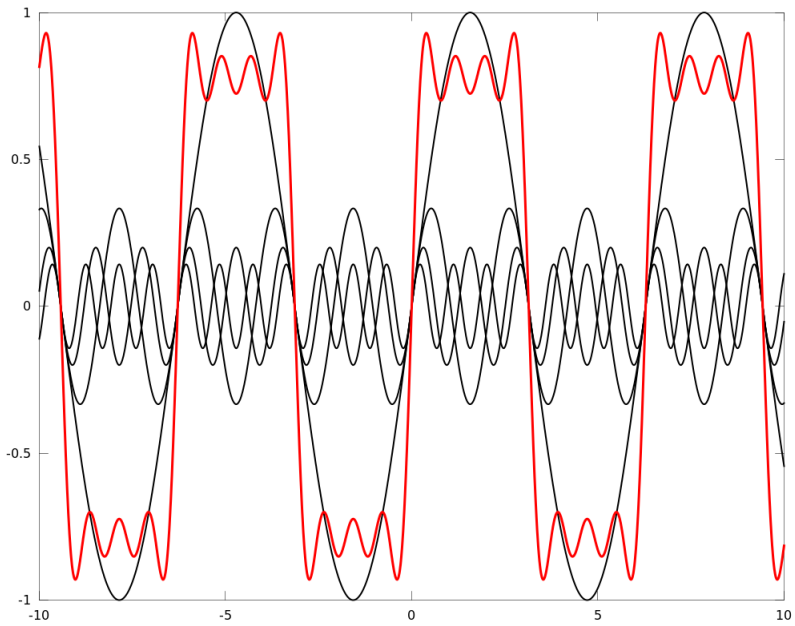
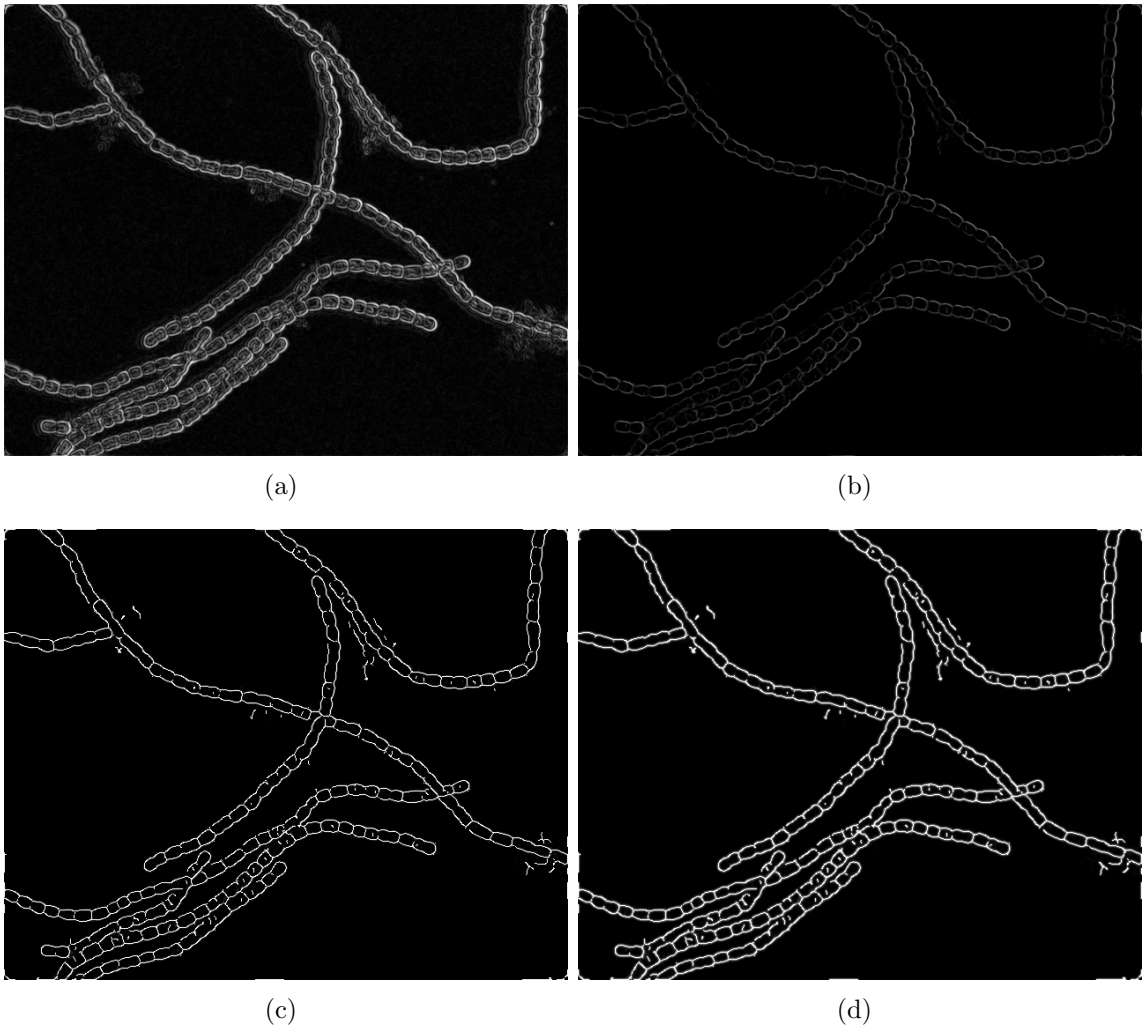


Figure 3.6: Sine waves of various amplitudes, wavelengths, and phase offsets (black) with their sum superimposed (red). There are several points with maximum *phase congruence*, where all the waves are in phase with each other. The black waves correspond to the first four terms of the Fourier series $\sum_{n=0}^{N} \frac{\sin(x \cdot (2n+1))}{2n+1}$.

in $[0, 1]$, then taking the pixel-wise maximum between this and the Steger mask will give an image with all prominent edges assigned the maximum intensity. A visual comparison of these edge-detection methods is presented in Figure 3.7. To decrease the resulting sharp slopes in preparation for an optimization process, the last step is to smooth the image with a Gaussian filter using $\sigma = 1.0$, and take the pixel-wise maximum between the two.

Generating the ridge image I_r is done by convolving the input image with various sized elliptical structural elements that are rotated through twelve evenly spaced orientations, as depicted in Figure 3.8. The maximum response across these orientations is taken to be the final result. This aims to highlight bright areas of the image which contain elliptical features such as cells. It is not expected that this will reliably detect many of the cells, but rather that it will indicate approximate regions



(a) (b)

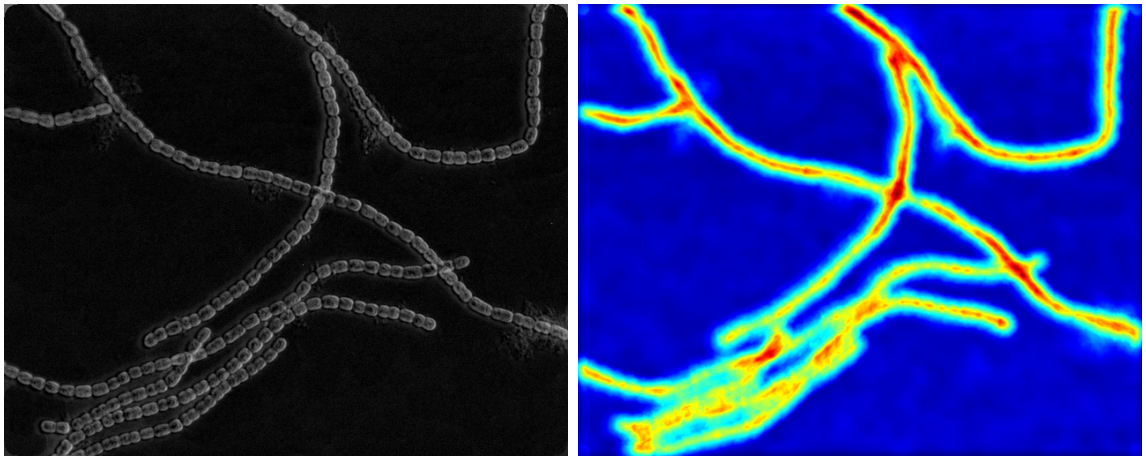
(c) (d)

Figure 3.7: Comparison of edge data images: (a) gradient magnitude using Gaussian derivative, (b) phase congruency with noise removal, (c) pixel-wise maximum of image (b) and Steger response of image (b), (d) pixel-wise maximum of image (c) and image (c) smoothed with Gaussian $\sigma = 1.0$.

Figure 3.8: All twelve orientations of elliptical structural elements used for generating ridge data. Each ellipse is no more than 25 pixels wide in its largest dimension.

of interest for each filament. Where the cells are more clearly defined, the result will be moderate responses near the edges of cells, gradually building to a local maximum near the cells' middle points. Taken across the entire filament, these local maxima

will form a median ridge through the centre of the filament. A sample of the ridge data for one input image is given in Figure 3.9.



(a) (b)

Figure 3.9: Visualization of ridge data image: (a) input image, (b) ridge data displayed in false colour.

3.2.2 Spline model for tracing filaments

The Bessel-Overhauser model is used to represent filament splines, and these are created by iteratively placing control points in the image. A spline is therefore defined entirely by its sequence of control points: writing $\mathbf{p}_i = (x_i, y_i)$ to indicate the i-th control point having coordinates (x_i, y_i), a spline S is the ordered set of n control points $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$. The i-th spline segment lies between control points $\mathbf{p}_i, \mathbf{p}_{i+1}$, and is given by a parameterized curve $\mathbf{r}^i(u) = (x(u), y(u))$ with $u \in [u_i, u_{i+1}]$ over this segment. The values of u_i, u_{i+1} are the left and right parameter *knots* of the i-th segment. Written without a superscript, the curve $\mathbf{r}(u)$ is the result of joining each of the segment curves $\mathbf{r}^i(u)$.

Bessel-Overhauser splines are C^1-continuous cubic splines which interpolate their control points. They allow using *chord length parameterization* to account for unevenly spaced control points. For each segment, a distance metric evaluates the

distance between the segment's end points, and this is used to parameterize the corresponding curve $\mathbf{r}^i(u)$. Thus, for the i-th spline segment connecting control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, the parameter knots satisfy

$$u_{i+1} = u_i + \|\mathbf{p}_{i+1} - \mathbf{p}_i\|.$$

The difference $\mathbf{p}_{i+1} - \mathbf{p}_i$ is a vector, and the operator $\|\cdot\|$ denotes its magnitude. We are using the Euclidean distance, so this is equivalent to

$$\|\mathbf{p}_{i+1} - \mathbf{p}_i\| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}.$$

Each control point $\mathbf{p}_i$ has a single associated tangent vector $\mathbf{t}_i$ satisfying the C^1 continuity constraint between its two neighbouring curves $\mathbf{r}^{i-1}(u), \mathbf{r}^i(u)$. These tangents are computed by first assigning each segment a *half tangent* vector

$$\mathbf{h}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{u_{i+1} - u_i}$$

of unit length, and then letting

$$\mathbf{t}_i = \frac{(u_{i+1} - u_i) \cdot \mathbf{h}_{i-1} + (u_i - u_{i-1}) \cdot \mathbf{h}_i}{u_{i+1} - u_{i-1}} \tag{3.2}$$

be the tangent vector for control point $\mathbf{p}_i$.

This definition requires a control point to have two neighbouring control points, which raises a difficulty with defining tangents for the end points of a spline. There are two solutions; one is to use the end control points solely to determine tangent information, so spline segments are only computed between control points $\mathbf{p}_2, \ldots, \mathbf{p}_{n-1}$. The other solution is to estimate the missing half tangent by setting it equal to the existing half tangent. Since we want our spline to interpolate all of its control points, we employ the latter strategy and define $\mathbf{t}_1 = \mathbf{h}_1$ for the first control point, and $\mathbf{t}_n = \mathbf{h}_{n-1}$ for the final control point.

A filament's spline is initialized as n control points, and the placement of subsequent control points is optimized by an evolutionary strategy. For now, we assume that $n > 1$. The result of running the ES is a *ridge spline* which traces the middle ridge of a single filament, and is defined by a set of evolved control points. This

is accompanied by two curves parallel to the ridge spline at a distance of $\frac{w}{2}$ each, where w is again the approximate width of a filament cell. These two *edge splines* are defined implicitly from the ridge spline and trace an approximation of the filament's edges. Since the ridge spline is the only explicitly defined tracing element, we will sometimes simply refer to it as *the* spline for a filament.

3.2.3 Evolutionary strategy: objective function

A full filament segmentation is represented by an ordered sequence of control points. These explicitly define a ridge spline for the filament, which in turn implicitly defines the two edge splines. However, only a subset of these control points is used in evolving the position of a new control point, under the assumption that well-established control points do not require any further improvements.

The search space is defined by individuals consisting of an N-dimensional vector containing the x and y coordinates of $\frac{N}{2}$ control points spanning $\frac{N}{2}-1$ spline segments. For a spline S with n existing control points, when adding the $(n+1)$-th control point the dimension of the search space is

$$N = 2 \cdot \min(n+1, 3).$$

This means that in evolving a new control point, we calculate its score based on the new spline segment and up to two previous segments, rather than scoring across the entire spline. Additionally, older control points are assigned much smaller step sizes to further limit the impact of higher dimensionality.

The scoring of individuals is based on the objective function in Equation (3.1). Edge splines and ridge splines are scored separately by using the images I_e and I_r generated in the pre-processing step (Section 3.2.1), and the length score is independent of the image. The ridge score for the $(n+1)$-th control point is determined by sampling M points u_j lying under the curve segments $\mathbf{r}^i(u)$ for $i = \frac{N}{2} - 1, \ldots, n$ and

taking the generalized mean of their pixel intensities

$$E_{\text{ridge}} = \left(\frac{1}{M} \cdot \sum_{j=1}^{M} I_r(\mathbf{r}^i(u_j))^q \right)^{\frac{1}{q}}.$$

In practice, the points along the spline are interpolated to pixel coordinates using the nearest neighbour approach; however, other methods should produce similar results.

The edge score for the $(n+1)$-th control point is determined in a similar manner, by sampling M points u_j lying under the two curves running parallel to $\mathbf{r}^i(u)$ at a distance $h = \frac{w}{2}$ of half the estimated width of cells in the image. As before, $i = \frac{N}{2} - 1, \ldots, n$ and we take the generalized mean of the pixel intensities at these points, then sum the two to form the edge score

$$E_{\text{edge}} = \left(\frac{1}{M} \cdot \sum_{j=1}^{M} I_e(\mathbf{r}^i(u_j) + h \cdot \mathbf{r}^i_{uu}(u_j))^q \right)^{\frac{1}{q}} + \left(\frac{1}{M} \cdot \sum_{j=1}^{M} I_e(\mathbf{r}^i(u_j) - h \cdot \mathbf{r}^i_{uu}(u_j))^q \right)^{\frac{1}{q}}.$$

Here, $\mathbf{r}_{uu}$ is the second derivative of the curve $\mathbf{r}$ with respect to the variable u, and points along the spline are again interpolated using the nearest neighbour method.

In both E_{edge} and E_{ridge}, the value of q in the generalized mean is a tunable parameter. When $q = 1$ the generalized mean reduces to the arithmetic mean

$$\frac{1}{M} \sum_{j=1}^{M} a_j,$$

and the limit of the sum as $q \to 0$ is the geometric mean

$$\left(\prod_{j=1}^{M} a_j \right)^{\frac{1}{M}}.$$

If we limit q to real values in the range $[0, 1]$, then smaller values will increase the sensitivity to low-intensity pixels when computing the scores. As we typically want edge and ridge scores to be sensitive to pixels with near-zero intensity, our default value is $q = 0.5$.

The length score for the i-th spline segment is determined separately from any image data, and is intended to encourage control points to be as sparsely distributed

as possible. A serious drawback of open snakes as implemented in *JFilament* is that energy is only evaluated at individual control points. This is a reasonable approximation if the control points are placed at each pixel along the curve, but then this negates the advantages of using a spline representation. If the control points are more widely spaced, the energy approximation terms become poor, and are much more sensitive to local noise. By encouraging spline segments to be as long as possible, our spline approach moves toward a more compact representation while still computing a score over the entire spline segment. The length score is computed as a function of the Euclidean distance between a new control point $\mathbf{p}_{n+1}$ and the existing control point $\mathbf{p}_{n-1}$

$$\|\mathbf{p}_{n+1} - \mathbf{p}_{n-1}\| = \sqrt{(x_{n+1} - x_{n-1})^2 + (y_{n+1} - y_{n-1})^2}.$$

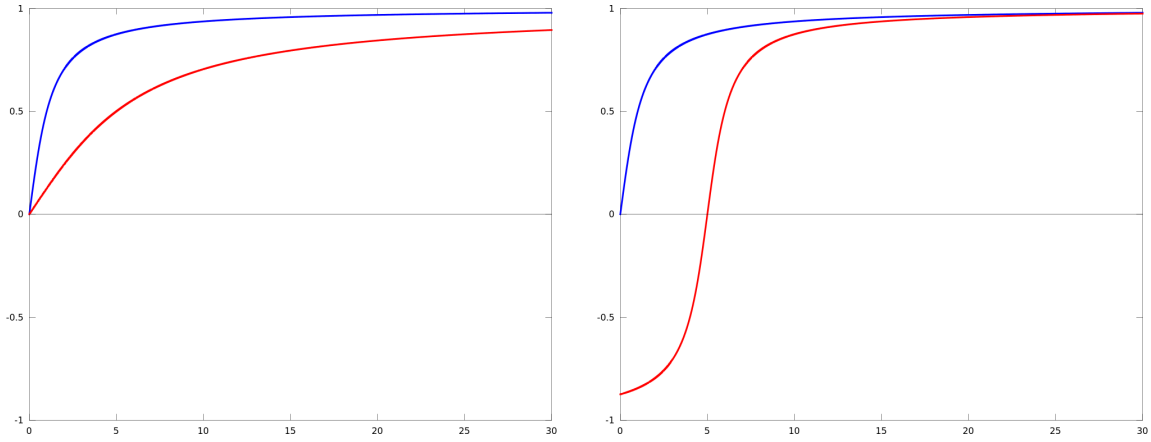
By using this distance, we favour splines which increase the overall length of the spline, rather than those which only increase the latest segment. This also allows for a more relaxed constraint on short segments, as very short segments will be generally permitted as required in regions of high curvature, so long as they are neighboured by a longer segment. Finally, measuring distance from the second last control point discourages splines which wrap immediately back on themselves.

We would like to scale the reward value so that we can control how quickly length increases are rewarded, and shift the reward value in order to penalize distances below a threshold. Through a combination of both, we also implicitly control the point of maximum slope in the reward function, so that distances near this point will be most encouraged to increase. To do so, the Euclidean distance is used as input to the sigmoid function

$$E_{\text{length}} = \frac{2}{\pi} \cdot \arctan\left(\frac{1}{b}\left(\|\mathbf{p}_{n+1} - \mathbf{p}_{n-1}\| - a\right)\right).$$

Here, the inverse tangent function is used; however, any sigmoid function scaled to the range $[-1, 1]$ would also be sufficient. The values of a and b are parameters used to change the impact of the length score on the overall value of the objective function. The value of $a \in \mathbb{R}$ shifts the function to control the x-intercept, and the value of $b \in (0, \infty)$ scales the function to determine its steepness. Interpreted in the

context of spline segments, if $\|\mathbf{p}_{n+1} - \mathbf{p}_{n-1}\| < a$ then the control point $\mathbf{p}_{n+1}$ will have a negative length score, and a length score of 0 if $\|\mathbf{p}_{n+1} - \mathbf{p}_{n-1}\| = a$. The value of b determines the value of $\|\mathbf{p}_{n+1} - \mathbf{p}_{n-1}\| + a$ which will give a score of 0.5, so smaller values of b will increase the overall steepness. The effects of parameters a and b are shown in Figure 3.10.



(a) (b)

Figure 3.10: Comparison of arctangent sigmoid functions, re-scaled to have range $[-1, 1]$. The function $f(x) = \frac{2}{\pi} \cdot \arctan(x)$ is in blue with $a = 0, b = 1$. In image (a), the function $g(x) = \frac{2}{\pi} \cdot \arctan\left(\frac{1}{5}(x)\right)$ is in red with $a = 0, b = 5$. In image (b), the function $g(x) = \frac{2}{\pi} \cdot \arctan((x - 5))$ is in red with $a = 5, b = 1$.

3.2.4 Evolutionary strategy: parameters

The covariance matrix adaptation evolution strategy [2] (CMA-ES) is a method for stochastically solving real-valued optimization problems [22, 21]. The CMA-ES algorithm works by combining parent individuals with a normally distributed mutation to form children, and at each generation it updates a covariance matrix C. This affects the shape of the normal distribution, favouring areas of the search space which will tend to lead to an optimal solution. This adaptation is handled transparently by the CMA-ES algorithm, and does not require any further tuning beyond providing

[2]The implementation of CMA-ES used is available as a software download from http://www.lri.fr/~hansen/

initial parameters.

In our instance, individuals are N-dimensional vectors corresponding to x- and y-coordinates for a set of $\frac{N}{2}$ spline control points. The CMA-ES algorithm is initialized with an N-dimensional vector $\mathbf{m}_0$, and from this λ children are generated for the first generation by sampling from a normal distribution $\sigma \cdot \mathcal{N}(0, C)$ with mean 0 and using covariance matrix C. This distribution is also scaled by the scaling factor σ which is a global step size parameter. The covariance matrix can also be written as a matrix eigendecomposition

$$C = BD^2B^{\mathrm{T}}$$

where B is an orthonormal matrix satisfying $BB^{\mathrm{T}} = I$, and D is a diagonal matrix with diagonal entries corresponding to the square roots of the eigenvalues. In practical terms, diag(D) together with σ specifies the step size for elements of each individual; specifically, for one of the $\frac{N}{2}$ control points.

The λ generated children form a single generation, and from these the best μ parents are chosen to generate the next vector mean $\mathbf{m}$. Assuming the $\mathbf{x}_i$ are written in order of descending objective function values, so that $\mathbf{x}_1$ is the fittest individual and $\mathbf{x}_\lambda$ the least fit, then the vector mean $\mathbf{m}^{(g)}$ for generation (g) is the weighted mean

$$\mathbf{m}^{(g)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_i^{(g)}$$

where the weights w_i are assigned by the CMA-ES algorithm based on the ordered ranking of individuals $\mathbf{x}_i$. The covariance matrix C and global step size σ are also updated for each generation, so that the general update rule for creating the next generation's λ individuals is

$$\mathbf{x}_k^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}\left(0, C^{(g)}\right)$$

for $k = 1, \ldots, \lambda$.

To initialize the CMA-ES, the half tangent $\mathbf{h}_{n-1}$ of the last spline segment is used as an approximation for the direction of the spline and a new control point is created

as

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \frac{w \cdot \mathbf{h}_{n-1}}{\|\mathbf{h}_{n-1}\|}$$

with magnitude w equal to the estimated cell width. Step size parameters are initialized as $\sigma = 0.6$ and the diagonal of matrix D is $\text{diag}(D) = [0.01, 0.25, 1]$. This gives a step size (standard deviation) of $1 \cdot \sigma = 0.6$ for the newest control point $\mathbf{p}_{n+1}$, 0.15 for the penultimate control point p_n, and 0.006 for $\mathbf{p}_{n-1}$. The control point $\mathbf{p}_{n-2}$ is used to evaluate the $(n-2)$-th spline segment, but its position is considered to be fixed. If only two control points are involved in the evolution process, then D is trimmed to remove the corresponding value.

Initializing the step size matrix this way gives the most freedom to the newest control point $\mathbf{p}_{n+1}$, as it is assumed this will require the most movement to optimize its score. The penultimate control point $\mathbf{p}_n$ is still allowed some leeway, giving the ES the opportunity to improve this placement using the updated information provided by attempting to place $\mathbf{p}_{n+1}$. The previous control point $\mathbf{p}_{n-1}$ is given an even smaller window in which to change, with the assumption that it has already been improved at least once. The CMA-ES will automatically change these step sizes as the algorithm runs by updating its covariance matrix, though it is expected that control point $\mathbf{p}_{n-1}$ will not have its value change significantly. Established control points $\mathbf{p}_i, i \leq (n-2)$, are taken as fixed and not involved in the evolution process.

Algorithm 1 Single filament segmentation algorithm

procedure SEGMENTFILAMENT($I_r, I_e, \mathbf{p}_1, \mathbf{p}_2$)
    stop := *false*
    $n := 2$
    while !stop **do**
        $\mathbf{p}_{n+1} = \mathbf{p}_n + \frac{w \cdot (\mathbf{p}_n - \mathbf{p}_{n-1})}{\|\mathbf{p}_n - \mathbf{p}_{n-1}\|}$
        $s = \min(n + 1, 3)$
        $[\mathbf{p}_n, \ldots, \mathbf{p}_{n+1-s}]$ = CMA-ES($[\mathbf{p}_n, \ldots, \mathbf{p}_{n+1-s}]$)
        $n := n + 1$
        stop = CheckStopConditions($[\mathbf{p}_n, \ldots, \mathbf{p}_1]$)
    end while
end procedure

Chapter 4

Results and Comparisons

Following the single filament segmentation algorithm, filament traces were mostly successful for manually selected starting points. To ensure that all filaments were segmented, each filament was assigned three control points by hand in close proximity, and from there the filament spline was evolved in both directions. Filament contours were universally followed closely, even in areas of very high curvature, and areas of image feature ambiguity were typically handled well. The number and spacing of control points appeared reasonable, with more control points being placed in areas of uncertainty or high curvature, and fewer in areas of relative homogeneity. As with any stochastic process, the results may change between runs of the algorithm.

Full evaluation of this algorithm would ideally be carried out over a large data set, but we are presently limited by images made available to us. These image sets are single frames drawn from two time-lapse videos showing the growth and movement of the cyanobacteria. Each subsequent frame tends to be more cluttered and more difficult to segment than the last, so we initially only consider the two starting frames. These frames still exhibit a wide variety of filament orientations and feature characteristics, so they are suitable candidates for an initial evaluation of the algorithm.

Two exemplary results are given here in Figure 4.1, while the full set of results is presented in Appendix A. Each segment was allowed to evolve for 125 generations and was scored using the objective function detailed in Section 3.2.3. It can be observed (see Figures 4.2, 4.3) that the edge and ridge splines conform to the contours of the filaments even when certain feature information is missing.

Overall, although performing far better than either alternative, the problem areas for our algorithm are still those which made the Steger and OAC approaches unusable. Parallel touching filaments present two viable paths for a segmentation algorithm to follow, and inconsistent image features can confuse the search for an optimal control point. The performance of the algorithm is then dependent on the quality of the underlying image features. Missing edges and uneven intensity can be accommodated, but only up to a point. In extremely cluttered images with multiple overlapping, intersecting, and touching filaments, the performance of the algorithm may be understandably poor.

For the rest of this chapter, we first give a context for these results by presenting equivalent results for both the OAC and Steger methods. We then discuss the successes and failures of our own approach.

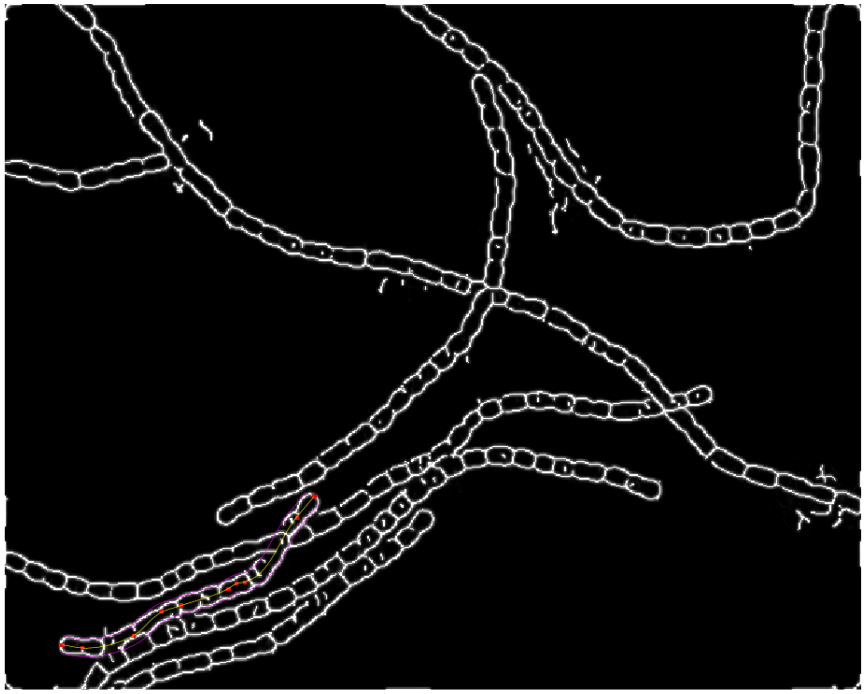
4.1 Presentation of alternatives

Implementations of both the Steger and open snake methods were used to process the simplest of our two primary sample images. As expected, their results were not sufficient for the task of identifying and segmenting individual filaments among clusters of multiple touching or overlapping filaments. We present results from both methods here.

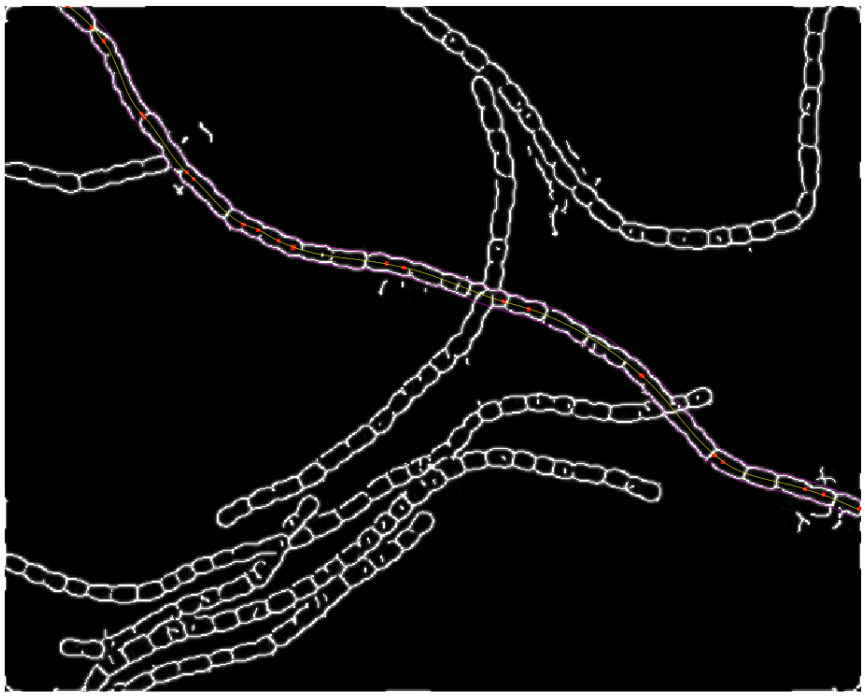
4.1.1 Steger curvilinear feature detection

To test the Steger method, we modified the implementation authored by Steger available through the GRASP project[1]. We ignored the labels assigned by pixel linking, and instead treated the Steger output as a binary mask indicating line ridges in the image. The resulting masks typically contained numerous false lines in regions of very low intensity; however, these could easily be thresholded away, so we focused instead on the lines given for regions containing filaments. Smoothing plays a crucial

[1]GRASP project details and GFPL code available online from
http://www.lsc-group.phys.uwm.edu/~ballen/grasp-distribution/



(a)



(b)

Figure 4.1: Two identified filaments with ridge spline (yellow) and edge splines (magenta). In image (a), the matched filament is tightly clustered with several other filaments, and is missing significant portions of edge data. In image (b), the matched filament is relatively isolated but very long, almost spanning the width of the image.

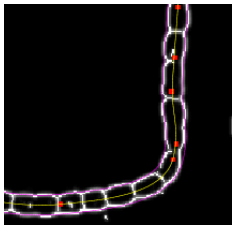


Figure 4.2: A filament spline (yellow) with matching edge splines (magenta), all specified by a set of control points (red). The edge splines do not match the filament edges at all points, particularly near the large bend in the filament, but the segmentation is still successful.

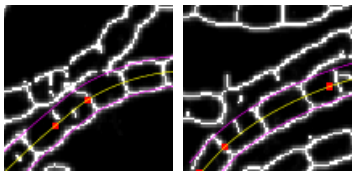
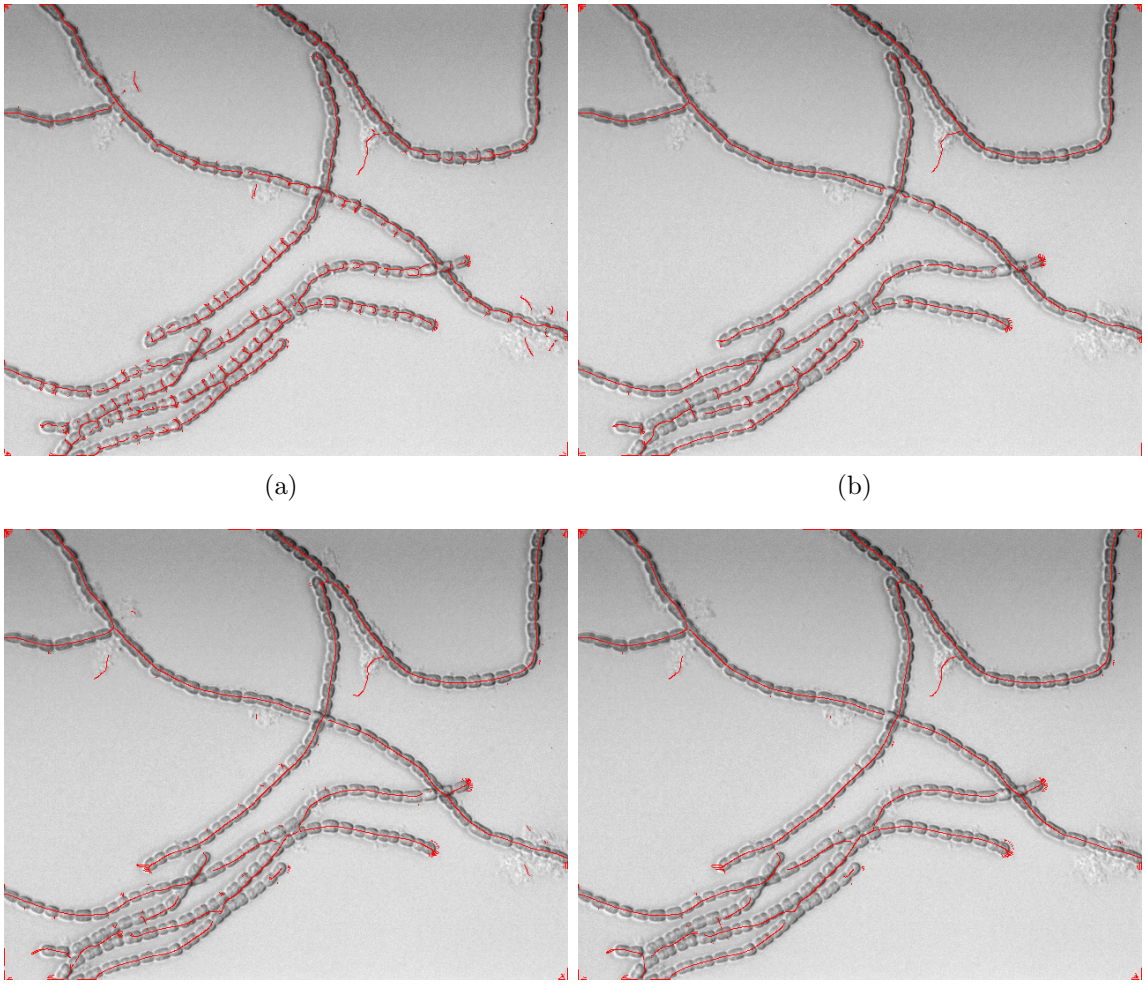


Figure 4.3: Two separate areas on a filament with patchy or missing edge information. In spite of this, the edge splines (magenta) still give a reasonable approximation to the filament contours.

role in the Steger algorithm, so the value of the smoothing parameter σ was varied to observe the range of possible output. Since the filament width is approximately $w = 13$ by manual inspection, the range of the smoothing parameter should be

$$3.75 \approx \frac{\sqrt{3}w}{6} \leq \sigma \leq \frac{w}{2} \approx 6.5.$$

The results of applying the Steger algorithm this way are given in Figure 4.4. In each case, the Steger mask is shown in red overlaid on the original (non-negative) image. Small smoothing values of σ permit the Steger approach to detect at least one edge of a filament with reasonable accuracy, but the resulting line does not approximate the filament's median. These images are also filled with smaller, insignificant lines which are detected between neighbouring cells. Larger smoothing values of σ remove these extraneous details and smooth the isolated filaments enough that the response very closely approximates the filaments' median. However, this amount of smoothing blurs the distinction between parallel touching filaments and results in



(a) (b)

(c) (d)

Figure 4.4: Results of running Steger algorithm (red) overlaid on original non-negativized image for easier viewing, where all images have had extraneous lines removed using thresholding. In image (a) with $\sigma = 3.75$, numerous orthogonal lines were detected between neighbouring cells on the filament, suggesting this smoothing parameter is too small. In image (b) with $\sigma = 4.75$, fewer such lines were detected, but the median lines are not very close to the middle of the filaments. In image (c) with $\sigma = 5.75$, orthogonal lines between neighbouring cells were rarely detected, and the median line position is improved. Parallel touching filaments are incorrectly treated as one line, and intersecting filaments have large gaps in their median lines. In image (d) with $\sigma = 6.25$, lines for parallel touching filaments were still incorrectly detected, and otherwise very few differences are visible from using $\sigma = 5.75$ in image (c).

these being treated as a single line. Intersections and orthogonal overlaps are generally handled well, although a robust pixel linking algorithm would still be required

to separate the lines for each filament.

4.1.2 Snake method

To test the viability of open snakes, we used the *JFilament* software referenced in Section 2.3.3. As with the Steger approach, smoothing is an important feature of the *JFilament* implementation; however, it has a different effect. Small smoothing parameters apparently have little impact on the algorithm's ability to segment entire filaments. Small areas of noise are reduced or removed, but others still remain as impediments to full segmentation. Moderately increasing the smoothing parameter results in erratic behaviour of the snakes, making segmentation impossible.

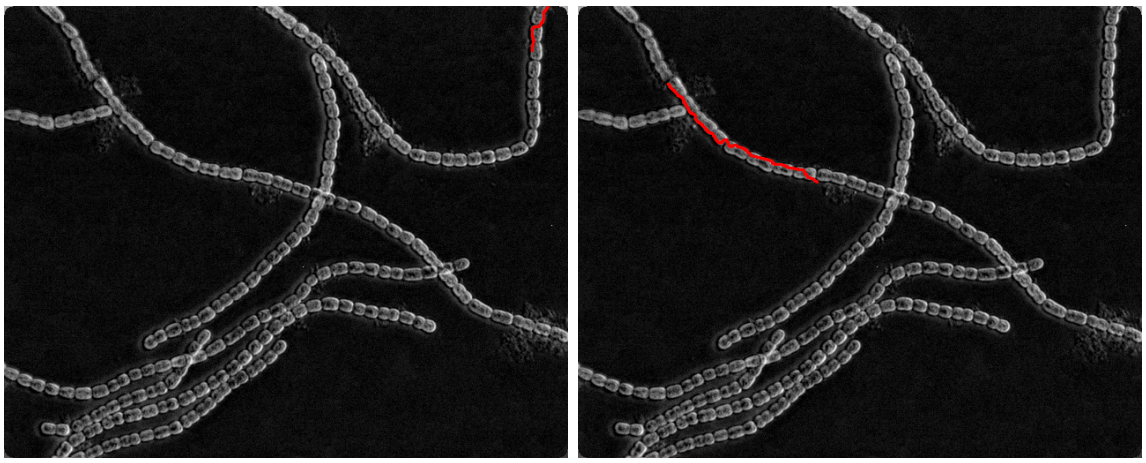
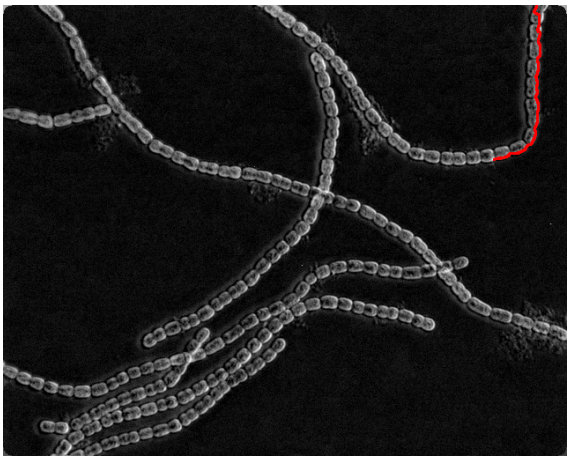


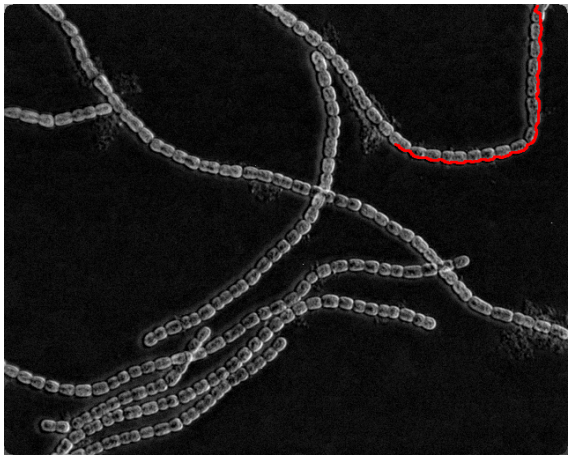
Figure 4.5: Two images showing open snakes stuck in local minima. Both snakes have minimized their energy functionals as much as possible in their local regions, and this has resulted in an incomplete segmentation. Both images also show the snakes attaching to edges of the filament rather than the median ridge lines, as well as tracking dividing lines between cells which leads to switching filament sides.

The software is interactive and requires the user to input starting points for the segmentation process to start. In our tests, we always chose starting points lying on or very near the median line of filaments. *JFilament* snakes also require several rounds of deformation, controlled by the user. When two subsequent deformations produce no noticeable differences in the snake, this is taken to be the converged upon

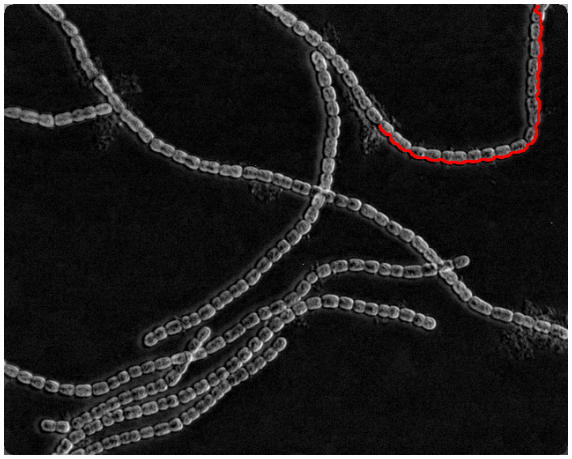
solution.



(a)



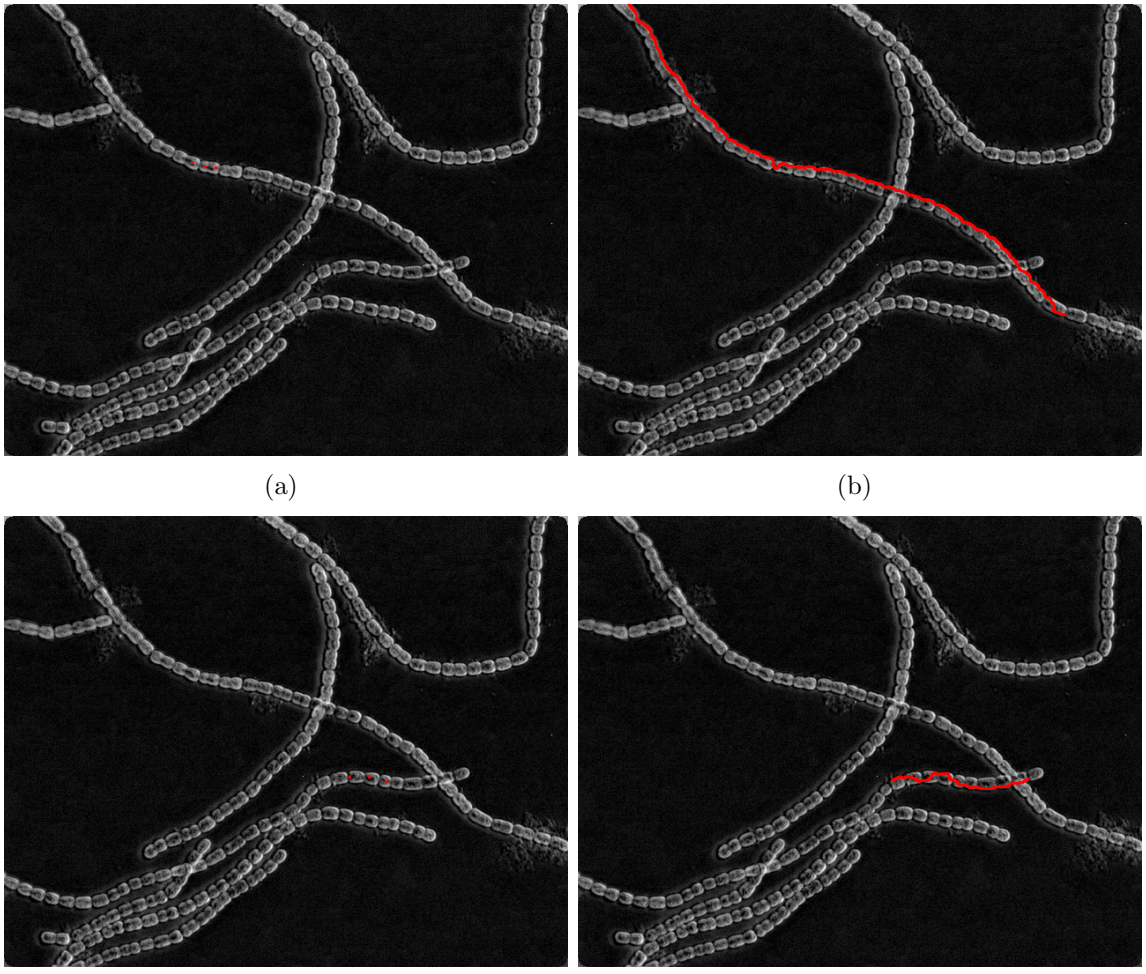
(b)



(c)

Figure 4.6: Progression of a more successful snake through several deformations. The first deformation (a) gives an open snake tracking one edge of the filament very closely, with a single switch between the filament sides near the top of the image. In images (b) and (c) respectively, the next two deformations increase the length of the snake along the filament's edge. The final snake in image (c) only identifies the edge along approximately half of the filament's length.

The results of attempting to track various filaments using open snakes are given in Figures 4.5, 4.6, and 4.7. The snakes are shown in bold red overlaid on the negative image used as input. No smoothing was used on these images, as no different



(a) (b)

(c) (d)

Figure 4.7: Two images (a) and (c) showing the starting points given by the user alongside the resulting open snakes in images (b) and (d), respectively. Despite both sets of starting points being located very close to their filaments' median lines, both snakes are attracted to the filament edges. Further, both snakes fail to track the full filament, and both switch between the two filament edges multiple times.

behaviour was observed with small smoothing parameter $\sigma = 1.0$. With smoothing parameter $\sigma > 2.0$, the convergence speed of most snakes was greatly reduced, their average length decreased, and many simply shrank from their starting points to smaller segments or even a single point. The two most salient defects in the non-smoothed results are the open snakes' inability to trace the entire length of a filament, and their attraction to the filament edges instead of the filament ridges. Local energy minima seemed to eventually prevent each snake from segmenting the

full filament, and smoothing was unable to correct for this.

4.2 Discussion of results

Both the Steger and OAC approaches were unable to accurately track the median ridge lines in our sample image, particularly in the case of overlapping and parallel touching filaments. By contrast, our method is able to distinguish between neighbouring filaments by searching beyond local optima and overlooking temporary disruptions in image features, such as edges. Full results are presented in Appendix A.

The most salient problem with the results is the capacity for over-segmentation. Several filaments were matched very well by their spline outlines, but one or more control points were added at the ends of the splines, beyond what was needed for a correct segmentation. This is to be expected: the very nature of the segmentation algorithm is to be biased toward encouraging exploration beyond a small number of poor scoring segments. Without this tendency, many of the more difficult regions would be impossible to correctly segment. The trade-off then is that stopping conditions are difficult to accurately and safely parameterize without compromising the algorithm's ability to segment accurately in difficult areas.

In all of the above images, several stopping criteria are used to indicate when a filament spline is finished. These are all handled at a level above the evolutionary strategy, which locates optimal control points without interpreting their context. The filament tracing process is halted if any of the following criteria are met:

(1) If the last control point was placed in a region with neighbouring pixels in the ridge image I_r all below a fixed threshold

(2) If the last control point was placed within a fixed number of pixels from the edge of the image I

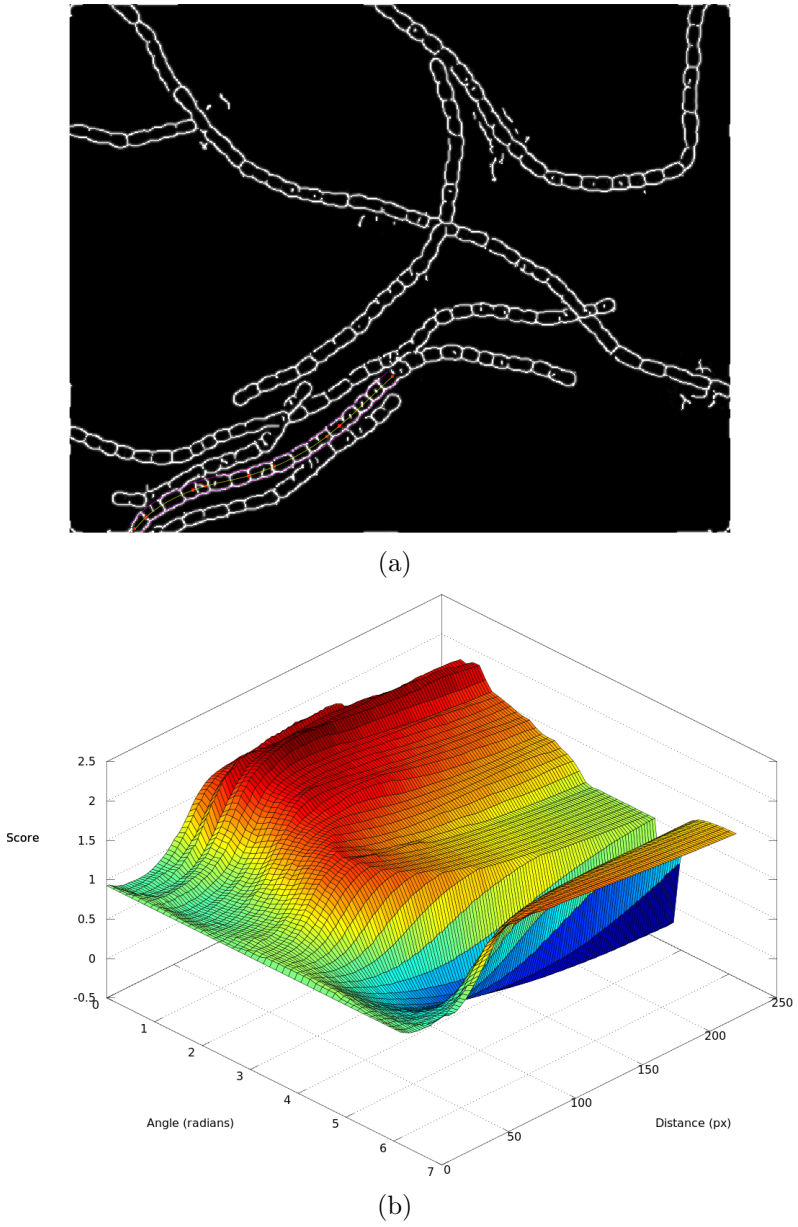
(3) If the last control point receives an edge score *or* ridge score which is less than a fixed percentage of a previous control point

(4) If the last control point was placed such that the resulting ridge spline passes through the previous segment's region

Stopping condition (1) is triggered by a filament spline stepping past the end of a filament and into an empty region of the image which can be confidently classified as a background region. The threshold used must be sufficiently strict to prevent false positives from occurring in regions of the foreground with relatively low intensities in I_r. Stopping condition (2) is triggered by a filament spline reaching the end of a filament which continues out of the current frame entirely. Since no more information is available outside the image, we stop the segmentation process. Stopping condition (3) occurs when a spline uses two spline segments to cover an area with very low score. This may indicate that the spline has reached the end of the filament and, pushed onward by the penalizing length score, has jumped to a second filament. The percentage drop-off required to trigger this condition must be sufficiently low to ensure it still allows the algorithm to explore areas with weak image features. Stopping condition (4) is triggered by a filament spline reaching the end of a filament and, rather than stepping off into an empty region with near-zero ridge and edge pixels, looping back onto the same filament. This is naturally discouraged by the continuity requirements of the spline representation and by our definition of the length score (see Section 3.2.3). However, when faced with the end of a filament the penalty for moving back along the filament may be outweighed by the penalty for moving forward into an empty region. This behaviour can be recognized easily, and the offending control point omitted from the final segmentation result.

These stopping conditions on their own are not enough to catch all stopping scenarios. Indeed, it may only be after one or more segments that it becomes clear that the segmentation should have terminated at some previous control point. Rather than compromising the efficiency of the algorithm with more *ad hoc* stopping criteria, we propose instead to use a simple *pruning* stage to eliminate extraneous spline segments, and otherwise rely on a human operator to trim splines by selecting which control points should serve as end points.

The automatic pruning process first checks if a filament spline is stopped under

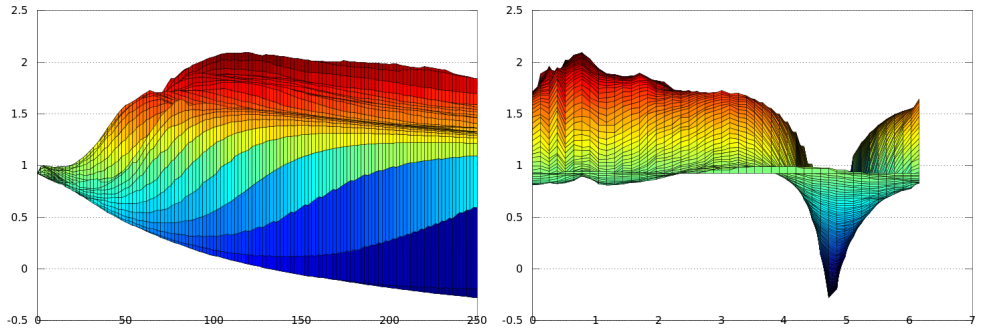


(a)

(b)

Figure 4.8: A simplified visualization of the objective function in the search space. The 3D surface (b) is the score of a new control point as a function of angle and distance relative to the last control point. The spline and control points used are also pictured (a).

condition (4) then the last segment is automatically removed, and no further pruning is required. In the case of stopping condition (2), there are also no extraneous spline segments needing to be removed. However, if stopping condition (1) is met, there may be one or more segments at the ends of the filament which are superfluous.



(a) (b)

Figure 4.9: The same objective function as Figure 4.8 as seen from the sides. Control point score is displayed: (a) as a function of pixel distance, (b) as a function of angle relative to previous control point.

To detect these, we look at the ridge and edge scores for each individual segment, ignoring the contribution from neighbouring spline segments. This uses the same underlying data as the objective function outlined in Section 3.2.3, but we are now only interested in scoring one segment at a time, and we ignore the length score entirely. Stepping backwards through the spline segments, those first few which have an individual segment score below a fixed threshold are discarded, and the first segment exceeding the threshold becomes the new terminating segment for the filament spline.

In some cases, no stopping conditions are ever met because a filament spline hops from one filament to another within a single spline segment. This is an unavoidable consequence of having an increased tolerance for missing edges and inconsistent intensity. A simple solution is to present the segmentation to an end user, who can easily identify a pruning point for splines that segment more than one filament. One alternative is to use more global information about the segmentation of other splines in the image. If we can consider two alternative segmentations of a single area and choose the one with higher segment scores, this would give a method for exhaustively identifying the "best possible" filament segmentations throughout the image. For the segmentation results given in Appendix A, any splines which hopped between two filaments had extraneous control points removed by a human operator, but were otherwise left unchanged.

The search space for optimizing new control points is 4- or 6-dimensional, as there are three control points, each with two degrees of freedom. It is understandably difficult to visualize the shape the objective function takes on within this space; however, we can approximate its appearance with a 2-dimensional slice of the objective function. In Figures 4.8 and 4.9, the evolution process of a single filament is halted near an area of some ambiguity. The usual approach at this stage is for the algorithm to initialize a new control point and vary its position along with the position of the two previous control points, searching for an improving score. Instead, these surface plots were generated by fixing the previous control points and only allowing the new control point to vary in its position. The offset is measured in relative angle θ (in radians) and relative distance r (in pixels) from the previous control point . There is obviously a global maximum lying approximately 100 pixels away, with several local maxima elsewhere on the surface. There are also faint ridges in the surface lying perpendicular with the θ-axis, showing the rise and fall of a new control point's score as it is moved away in a fixed direction.

Chapter 5

Conclusions and Directions for Future Work

Our goal was to create a fully automatic system for the segmentation and identification of filamentous cyanobacteria in bright field micrographs. As demonstrated, existing methods are either unable to perform robust segmentation in bright field images, or fail to reliably track filaments in cluttered images, or both. By contrast, our proposed method is able to use information from a bright field image in a complementary way that allows for missing or incomplete image features. Using a spline representation allows for implicit constraints on continuity and gives a compact data structure for storing the segmentation results. Using an evolutionary strategy allows us to break free from local optima in the search space, while also allowing for a dynamic length of spline segments. Our proposed method is the first step toward a fully automatic system, but it still requires the intervention of a human operator to process its output.

The resulting filament splines can be observed to very accurately trace the median line of their associated filament, in some cases even changing orientation for very slight curves between cells. The edge splines, which are constructed implicitly and mirror the path of the filament spline, also track the filament edges accurately. Since the distance between edge splines and their mirrored ridge spline is a matter of user input, the estimated cell width parameter is clearly a critical component for reliable results.

While our method is robust against missing edges and apparent gaps in filaments, these strengths become liabilities in the context of a global segmentation problem. Although able to appropriately handle intersecting filaments and filaments in parallel, there remains the issue of appropriate stopping conditions which signal that no further spline segments should be added to a filament spline. If the filament runs

to the edge of the image or ends in an uncrowded area, then it is a simple matter of checking for a sudden drop-off in spline score in a single step. Things are more complicated with filaments that terminate in a 'T-shaped' collision with another filament, or in an area with other nearby filaments. In these cases, the gap between filaments is indistinguishable from the gaps in features which normally occur within a filament. These situations can be adequately handled by a human operator when segmenting one filament at a time, but the ultimate goal is to have a fully automated system that operates with a minimum of user input. To correctly identify meaningful stopping criteria, we then need to apply the filament tracing approach at a more global level.

5.1 Future work

Stopping conditions are the most pressing concern, and perhaps the most straightforward to resolve. Since we cannot automatically distinguish accurate stopping conditions within a single filament, the next stage is to compare information globally across all filaments in an image. Spline representations for multiple filaments can be maintained at the same time, where a new spline segment will be added to only one filament at a time. When two filament splines overlap, we must make a decision on which spline is a better match. Looking at the segments near where the two filament splines initially crossed, the relative spline segment scores can be compared and the spline with a better score will be taken to be correct. The losing spline will be truncated to the control point before the spline intersection, all of the winning spline's pixels will be temporarily set to zero, and the losing spline will be allowed to attempt evolving again.

In segmenting a single filament, we might rely on user input to initialize the first set of control points to serve as a root for evolving the spline representation. In a global setting, this is no longer an option. We will therefore require a method for automatically generating starting control points. A minimum of two control points lying near the middle of a filament are required in order to designate both position and local orientation. The control points need not be separated by any significant

distance, yet there is no minimum distance required between the control points. The ridge image generated in Section 3.2.1 serves as a reasonable starting point. Dividing the image into smaller regions and then finding points of local maxima gives a single starting control point. To find a second, we look at pixels lying at a fixed small distance and take the global maximum from the circle's perimeter.

Another natural extension is to use the filament segmentation results to permit individual cell segmentation, especially in highly cluttered images. Having a median ridge spline for a filament designates a very narrow window where cell centres can appear, and the edge splines limit the position of cell membranes. It may be possible to modify traditional cell segmentation methods to use this as a starting point to improve their performance in bright field images with numerous cells.

Finally, we should investigate embedding this approach in the framework of time-lapse image analysis. If we assume that the first frame of a video can be reliably segmented and each filament assigned an accurate spline representation, then how would these splines be adjusted to accommodate sequential frames with slight offsets in the filament positions and orientations? Work has been done on registering micrographs from time-lapse imagery [58, 67, 51, 2], and this literature should be investigated for usable results. An even more interesting question is to ask: if we assume that no single frame of a video can be completely reliably segmented, can we combine mutual information between neighbouring frames to infer a total segmentation? This kind of result would have immediate applications in time-lapse imaging microscopy.

Appendix A

Figures of Segmented Filaments

Figures A.1, A.2, A.3, A.4, A.5, A.6, A.7, and A.8 all contain individual segmentation results from a single image. All filaments in this image were successfully segmented. Figures A.9, A.10, A.11, and A.12 contain selected filaments from a second sample image with much more crowded filaments. In the second image, only a selection of filaments were attempted. For reference, both original images are given in Figure 1.1.

Control points for each filament were manually initialized and the full spline evolved autonomously. Where necessary and in accordance with the stopping conditions described in Section 4.2, splines were automatically or manually trimmed by removing extraneous control points.

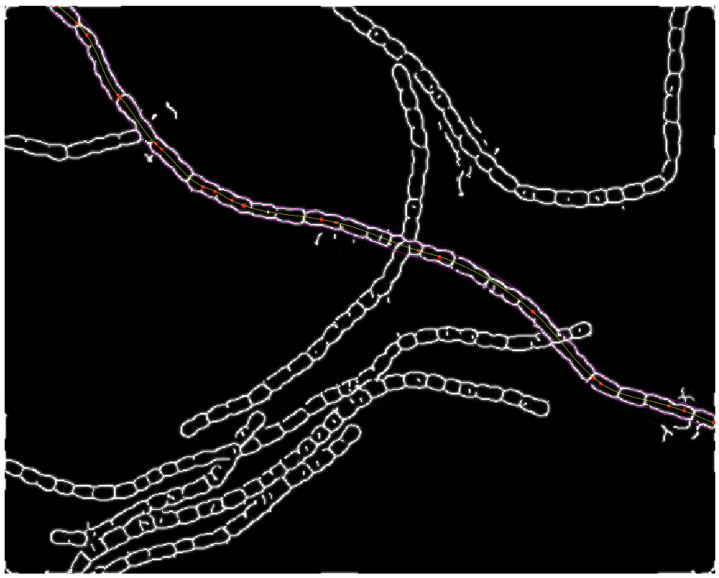


Figure A.1: Filament 1-1, correct segmentation with 22 control points total.

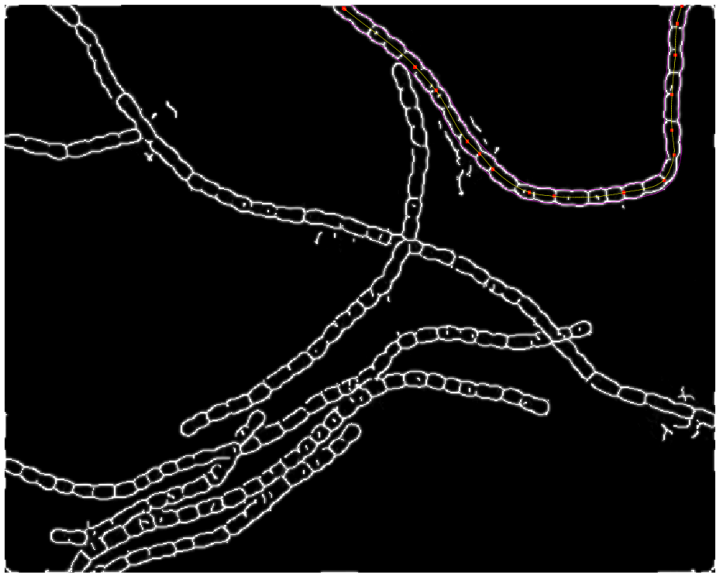


Figure A.2: Filament 1-2, correct segmentation with 17 control points total.

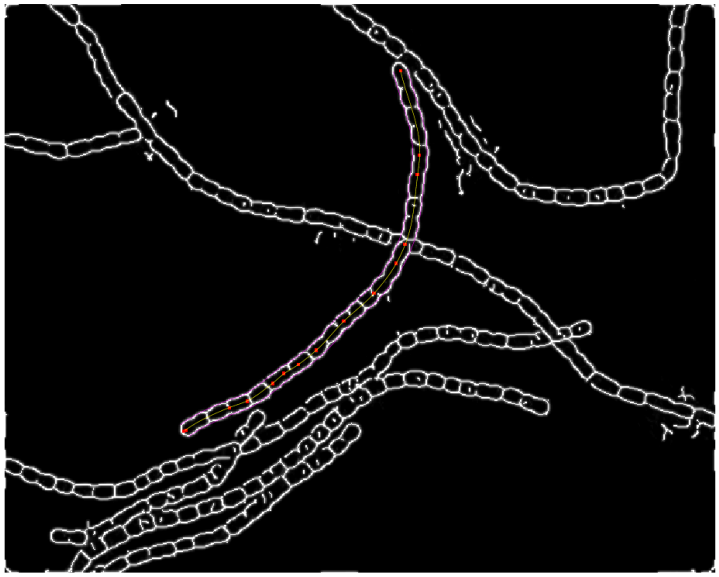


Figure A.3: Filament 1-3, correct segmentation with 15 control points total.

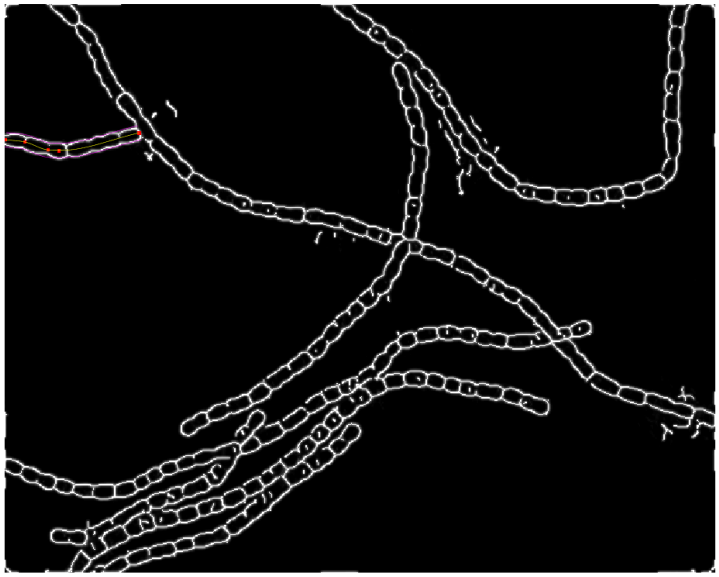


Figure A.4: Filament 1-4, correct segmentation with 5 control points total.

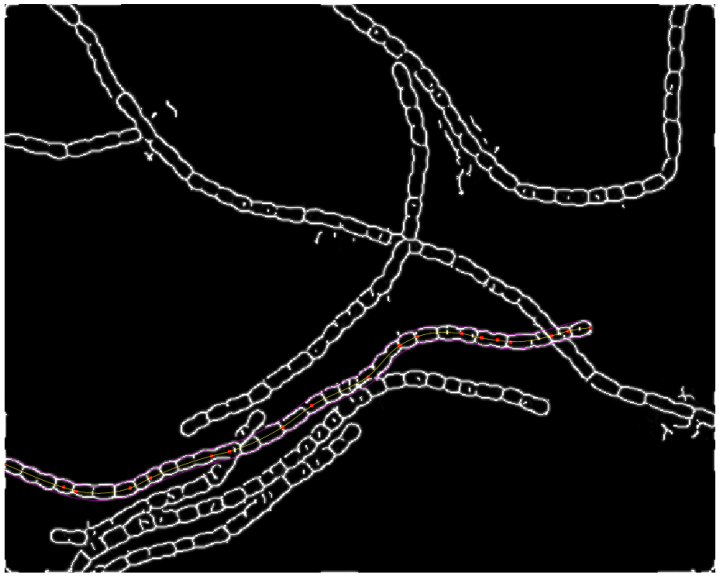


Figure A.5: Filament 1-5, correct segmentation with 19 control points total.

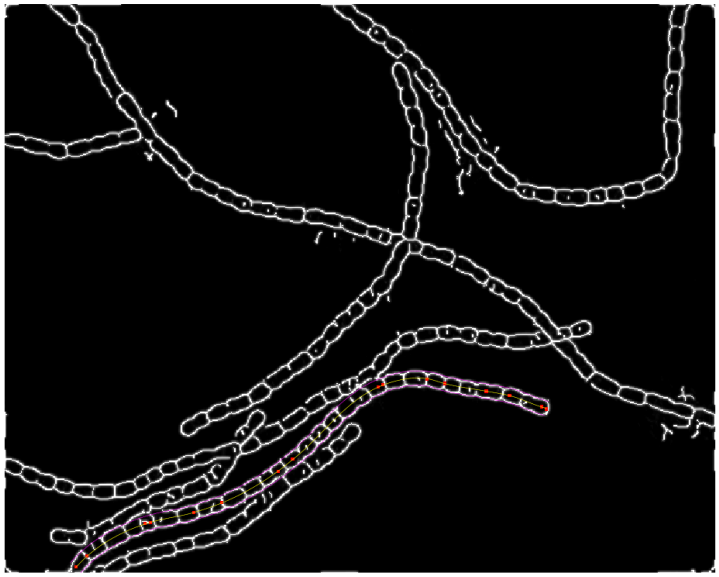


Figure A.6: Filament 1-6, correct segmentation with 16 control points total.

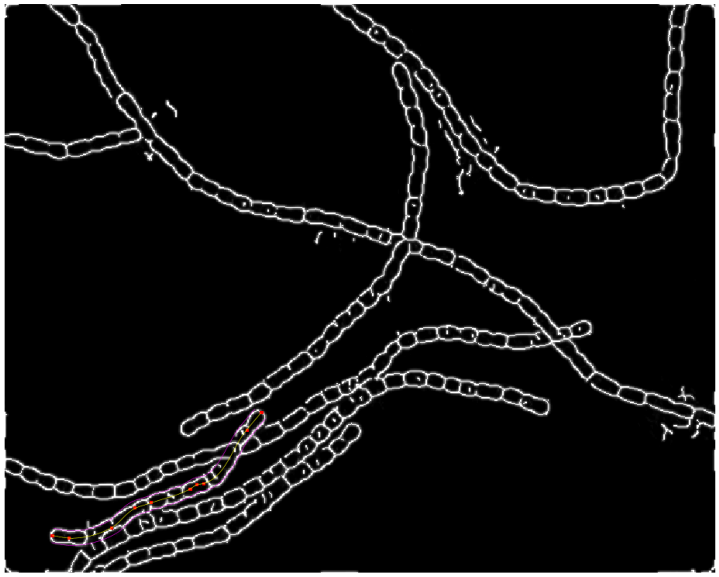


Figure A.7: Filament 1-7, correct segmentation with 10 control points total.

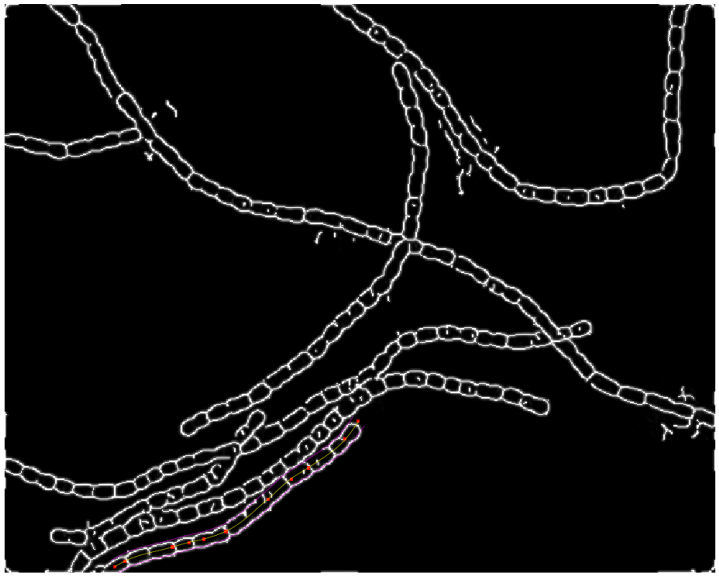


Figure A.8: Filament 1-8, correct segmentation with 11 control points total.

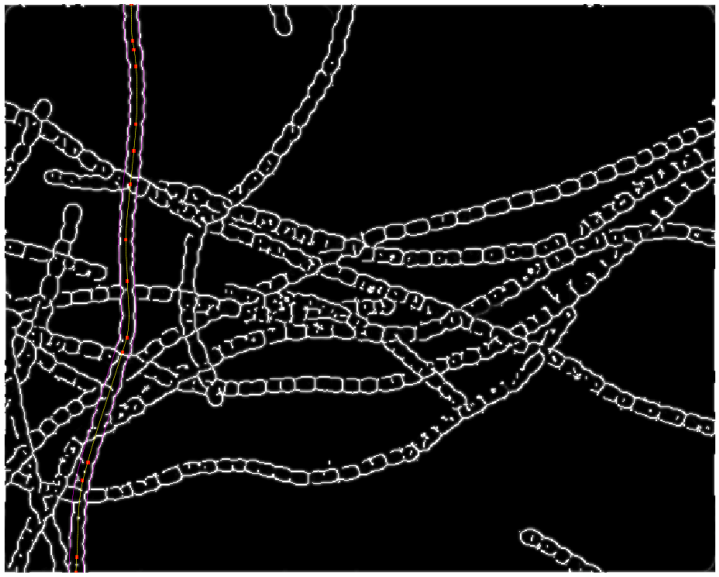


Figure A.9: Filament 2-1, correct segmentation with 15 control points total.

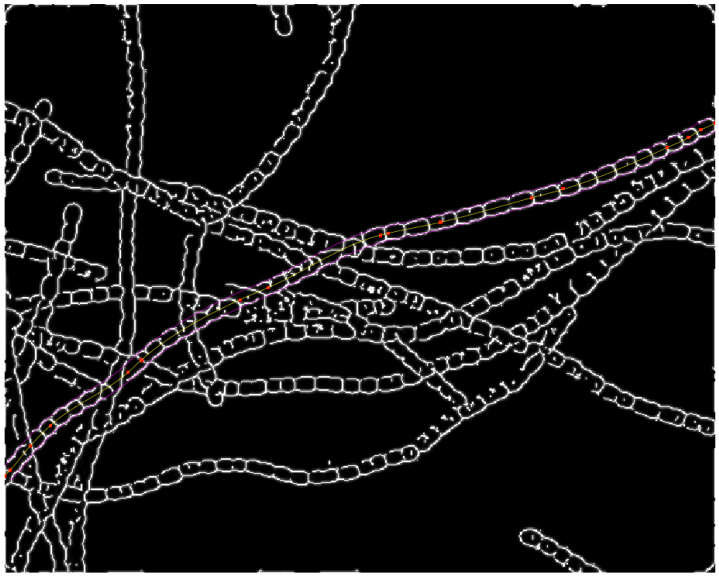


Figure A.10: Filament 2-2, correct segmentation with 17 control points total.

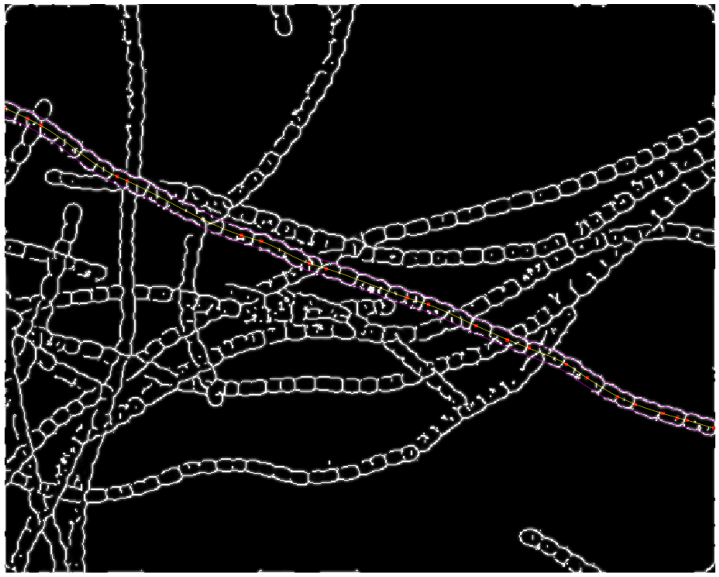


Figure A.11: Filament 2-3, correct segmentation with 22 control points total.

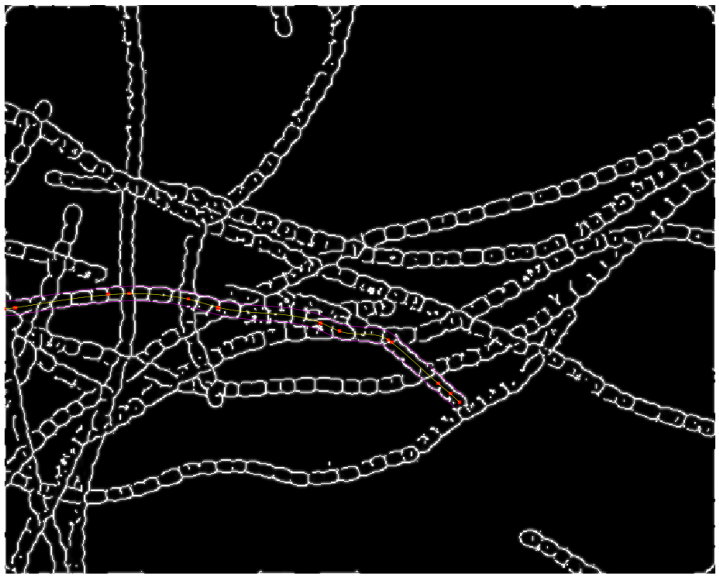


Figure A.12: Filament 2-4, incorrect segmentation with 22 control points total.

Bibliography

[1] L. Almesjö and C. Rolff, *Automated measurements of filamentous cyanobacteria by digital image analysis*, Limnology and Oceanography: Methods **5** (2007), 217–224.

[2] H. Asai, S. Iwamori, K. Kawai, S. Ehira, J. Ishihara, K. Aihara, S. Shoji, and H. Iwasaki, *Cyanobacterial cell lineage analysis of the spatiotemporal hetR expression profile during heterocyst pattern formation in anabaena sp. pcc 7120*, PloS ONE **4** (2009), no. 10, e7371.

[3] M.J. Aschwanden, J.K. Lee, G.A. Gary, M. Smith, and B. Inhester, *Comparison of five numerical codes for automated tracing of coronal loops*, Solar Physics **248** (2008), no. 2, 359–377.

[4] X. Bai, C. Sun, and F. Zhou, *Splitting touching cells based on concave points and ellipse fitting*, Pattern recognition **42** (2009), no. 11, 2434–2446.

[5] E. Bengtsson, C. Wählby, and J. Lindblad, *Robust cell image segmentation methods*, Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications **14** (2004), no. 2, 157–167.

[6] L. Bradbury and J.W.L. Wan, *A spectral k-means approach to bright-field cell image segmentation*, 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2010 (EMBC 2010), IEEE, 2010, pp. 4748–4751.

[7] K. Bredies and H. Wolinski, *An active-contour based algorithm for the automated segmentation of dense yeast populations on transmission microscopy images*, Tech. Report 2011-028, Medical University of Graz, 2011.

[8] D.D. Brüllmann, A. Pabst, K.M. Lehmann, T. Ziebart, M.O. Klein, and B. d'Hoedt, *Counting touching cell nuclei using fast ellipse detection to assess in vitro cell characteristics: a feasibility study*, Clinical Oral Investigations **16** (2012), no. 1, 33–38.

[9] E.D. Cheng, S. Challa, and R. Chakravorty, *Microscopic cell segmentation and dead cell detection based on cfse and pi images by using distance and watershed transforms*, Digital Image Computing: Techniques and Applications, 2009 (DICTA 2009), IEEE, 2009, pp. 32–39.

[10] A.Y.S. Chia, M.K.H. Leung, H.L. Eng, and S. Rahardja, *Ellipse detection with Hough transform in one dimensional parametric space*, IEEE International Conference on Image Processing, 2007 (ICIP 2007), vol. 5, IEEE, 2007, pp. 333–336.

[11] A.Y.S. Chia, D. Rajan, M.K.H. Leung, and S. Rahardja, *A split and merge based ellipse detector*, 15th IEEE International Conference on Image Processing, 2008 (ICIP 2008), IEEE, 2008, pp. 3212–3215.

[12] F. Cloppet and A. Boucher, *Segmentation of complex nucleus configurations in biological images*, Pattern Recognition Letters **31** (2010), no. 8, 755–761.

[13] J. De Vylder and W. Philips, *Computational efficient segmentation of cell nuclei in 2d and 3d fluorescent micrographs*, Proceedings of SPIE, the International Society for Optical Engineering, vol. 7902, 2011.

[14] R.O. Duda and P.E. Hart, *Use of the Hough transformation to detect lines and curves in pictures*, Communications of the ACM **15** (1972), no. 1, 11–15.

[15] N. Durak, O. Nasraoui, and J. Schmelz, *Coronal loop detection from solar images*, Pattern Recognition **42** (2009), no. 11, 2481–2491.

[16] B. Ernst, S. Neser, E. OBrien, S.J. Hoeger, and D.R. Dietrich, *Determination of the filamentous cyanobacteria planktothrix rubescens in environmental water samples using an image processing system*, Harmful Algae **5** (2006), no. 3, 281–289.

[17] A. Fitzgibbon, M. Pilu, and R.B. Fisher, *Direct least square fitting of ellipses*, IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999), no. 5, 476–480.

[18] E. Flores and A. Herrero, *Compartmentalized function through cell differentiation in filamentous cyanobacteria*, Nature Reviews Microbiology **8** (2009), no. 1, 39–50.

[19] R.C. González and R.E. Woods, *Digital image processing*, Pearson/Prentice Hall, 2008.

[20] A.A. Goshtasby, *Grouping and parameterizing irregularly spaced points for curve fitting*, ACM Transactions on Graphics **19** (2000), no. 3, 185–203.

[21] N. Hansen, *The CMA evolution strategy: a comparing review*, Towards a New Evolutionary Computation, Studies in Fuzziness and Soft Computing, vol. 192, Springer, 2006, pp. 75–102.

[22] ———, *The CMA evolution strategy: A tutorial*, `http://www.lri.fr/~hansen/cmatutorial.pdf`, 2011.

[23] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active contour models*, International Journal of Computer Vision **1** (1988), no. 4, 321–331.

[24] N. Kharma, H. Moghnieh, J. Yao, Y.P. Guo, A. Abu-Baker, J. Laganiere, G. Rouleau, and M. Cheriet, *Automatic segmentation of cells from microscopic imagery using ellipse detection*, Image Processing, IET **1** (2007late publisher=IET), no. 1, 39–47.

[25] A. Korzynska, W. Strojny, A. Hoppe, D. Wertheim, and P. Hoser, *Segmentation of microscope images of living cells*, Pattern Analysis & Applications **10** (2007), no. 4, 301–319.

[26] S. Kothari, Q. Chaudry, and M.D. Wang, *Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques*, IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2009 (ISBI 2009), IEEE, 2009, pp. 795–798.

[27] P. Kovesi, *Phase congruency: A low-level image invariant*, Psychological Research **64** (2000), no. 2, 136–148.

[28] _____, *Edges are not just steps*, Proceedings of the Fifth Asian Conference on Computer Vision (ACCV 2002), 2002, pp. 822–827.

[29] _____, *Phase congruency detects corners and edges*, The Australian Pattern Recognition Society Conference: DICTA 2003, vol. 1, DICTA, 2003, pp. 309–318.

[30] M. Kvarnström, K. Logg, A. Diez, K. Bodvard, and M. Käll, *Image analysis algorithms for cell contour recognition in budding yeast*, Optics Express **16** (2008), no. 17, 12943–12957.

[31] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, and A. Baumgartner, *Automatic extraction of roads from aerial images based on scale space and snakes*, Machine Vision and Applications **12** (2000), no. 1, 23–31.

[32] D. Lesage, E.D. Angelini, I. Bloch, and G. Funka-Lea, *A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes*, Medical Image Analysis **13** (2009), no. 6, 819–845.

[33] B. Li and S.T. Acton, *Active contour external force using vector field convolution for image segmentation*, IEEE Transactions on Image Processing **16** (2007), no. 8, 2096–2106.

[34] G. Li, T. Liu, J. Nie, L. Guo, J. Chen, J. Zhu, W. Xia, A. Mara, S. Holley, and STC Wong, *Segmentation of touching cell nuclei using gradient flow tracking*, Journal of Microscopy **231** (2008), no. 1, 47–58.

[35] H. Li, T. Shen, M.B. Smith, I. Fujiwara, D. Vavylonis, and X. Huang, *Automated actin filament segmentation, tracking and tip elongation measurements based on open active contour models*, IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2009 (ISBI 2009), IEEE, 2009, pp. 1302–1305.

[36] X. Li, Y. Wang, Y. Deng, and J. Yu, *Cell segmentation using ellipse curve segmentation and classification*, 1st International Conference on Information Science and Engineering, 2009 (ICISE 2009), IEEE, 2009, pp. 1187–1190.

[37] Y. Liu, H. Yang, and W. Wang, *Reconstructing B-spline curves from point clouds - a tangential flow approach using least squares minimization*, International Conference on Shape Modeling and Applications, 2005, IEEE, 2005, pp. 4–12.

[38] V. Ljosa and A.E. Carpenter, *Introduction to the quantitative analysis of two-dimensional fluorescence microscopy images for cell-based screening*, PLoS Computational Biology **5** (2009), no. 12, e1000603.

[39] T. McInerney and D. Terzopoulos, *Topologically adaptable snakes*, Proceedings of the Fifth International Conference on Computer Vision, 1995, IEEE, 1995, pp. 840–845.

[40] E. Meijering, I. Smal, O. Dzyubachyk, and J.C. Olivo-Marin, *Time-lapse imaging*, Microscope Image Processing (2008), 401–440.

[41] S. Osher and R.P. Fedkiw, *Level set methods and dynamic implicit surfaces*, Applied Mathematical Sciences, vol. 153, Springer Verlag, 2003.

[42] S. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics **79** (1988), no. 1, 12–49.

[43] D. Padfield, J. Rittscher, N. Thomas, and B. Roysam, *Spatio-temporal cell cycle phase analysis using level sets and fast marching methods*, Medical Image Analysis **13** (2009), no. 1, 143–155.

[44] H. Peng, *Bioimage informatics: a new area of engineering biology*, Bioinformatics **24** (2008), no. 17, 1827–1836.

[45] K. Raghupathy, *Curve tracing and curve detection in images*, Master's thesis, Cornell University, 2004.

[46] K. Raghupathy and T.W. Parks, *Improved curve tracing in images*, IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004 (ICASSP 2004), vol. 3, IEEE, 2004, pp. 581–584.

[47] C. Restif, *Segmentation and evaluation of fluorescence microscopy images*, Ph.D. thesis, Oxford Brookes University, 2006.

[48] M.E. Sargin, A. Alttnok, K. Rose, and B.S. Manjunath, *Tracing curvilinear structures in live cell images*, IEEE International Conference on Image Processing, 2007 (ICIP 2007), vol. 6, IEEE, 2007, pp. 285–288.

[49] L. Shamir, J.D. Delaney, N. Orlov, D.M. Eckley, and I.G. Goldberg, *Pattern recognition software and techniques for biological image analysis*, PLoS Computational Biology **6** (2010), no. 11, e1000974.

[50] M. Shemesh and O. Ben-Shahar, *Free boundary conditions active contours with applications for vision*, Advances in Visual Computing **6938** (2011), 180–191.

[51] M.B. Smith, H. Li, T. Shen, X. Huang, E. Yusuf, and D. Vavylonis, *Segmentation and tracking of cytoskeletal filaments using open active contours*, Cytoskeleton **67** (2010), no. 11, 693–705.

[52] C. Steger, *Extracting curvilinear structures: A differential geometric approach*, 4th European Conference on Computer Vision, 1996 (ECCV 1996) **1064** (1996), 630–641.

[53] ———, *An unbiased detector of curvilinear structures*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 2, 113–125.

[54] P. Thevenaz, R. Delgado-Gonzalo, and M. Unser, *The ovuscule*, IEEE Transactions on Pattern Analysis and Machine Intelligence **33** (2011), no. 2, 382–393.

[55] P. Thevenaz and M. Unser, *Snakuscules*, IEEE Transactions on Image Processing **17** (2008), no. 4, 585–593.

[56] S. Tse, L. Bradbury, J.W.L. Wan, H. Djambazian, R. Sladek, and T. Hudson, *A combined watershed and level set method for segmentation of bright field cell images*, Proceedings of SPIE Symposium on Medical Imaging: Image Processing **7259** (2009).

[57] L.A. Vese and T.F. Chan, *A multiphase level set framework for image segmentation using the Mumford and Shah model*, International Journal of Computer Vision **50** (2002), no. 3, 271–293.

[58] Q. Wang, J. Niemi, C.M. Tan, L. You, and M. West, *Image segmentation and dynamic lineage analysis in single-cell fluorescence microscopy*, Cytometry Part A **77** (2010), no. 1, 101–110.

[59] W. Wang, H. Pottmann, and Y. Liu, *Fitting b-spline curves to point clouds by curvature-based squared distance minimization*, ACM Transactions on Graphics (TOG) **25** (2006), no. 2, 214–238.

[60] R.D. Wedowski, A.R. Farooq, L.N. Smith, and M.L. Smith, *High speed, multiscale tracing of curvilinear features with automated scale selection and enhanced orientation computation*, International Conference on High Performance Computing and Simulation, 2010 (HPCS 2010), IEEE, 2010, pp. 410–417.

[61] Q. Wu, F.A. Merchant, and K.R. Castleman, *Microscope image processing*, Academic Press, 2008.

[62] Y. Xie and Q. Ji, *A new efficient ellipse detection method*, 16th International Conference on Pattern Recognition, 2002, vol. 2, IEEE, 2002, pp. 957–960.

[63] C. Xu and J.L. Prince, *Snakes, shapes, and gradient vector flow*, IEEE Transactions on Image Processing **7** (1998), no. 3, 359–369.

[64] L. Xu, E. Oja, and P. Kultanen, *A new curve detection method: randomized Hough transform (RHT)*, Pattern Recognition Letters **11** (1990), no. 5, 331–338.

[65] C. Yap and H. Lee, *Identification of cell nucleus using a Mumford-Shah ellipse detector*, Springer, 2008, pp. 582–593.

[66] M. Zeder, S. Van den Wyngaert, O. Köster, K.M. Felder, and J. Pernthaler, *Automated quantification and sizing of unbranched filamentous cyanobacteria by model-based object-oriented image analysis*, Applied and environmental microbiology **76** (2010), no. 5, 1615–1622.

[67] S. Zheng, *An intensive restraint topology adaptive snake model and its application in tracking dynamic image sequence*, Information Sciences **180** (2010), no. 16, 2940–2959.