CONFOCAL MICROSCOPY IMAGING ANALYSIS

OF PLANT MORPHODYNAMICS

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirement for the Degree

Master of Computer Science

By

JOHN T. FORTMAN

Dr. Ye Duan, Thesis Supervisor

MAY 2010

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

CONFOCAL MICROSCOPY IMAGING ANALYSIS

OF PLANT MORPHODYNAMICS

presented by John T. Fortman,

a candidate for the degree of master of computer science, and hereby certify that, in their opinion, it is worthy of acceptance.

_____

Professor Ye Duan

_____

Professor James Keller

_____

Professor Bruce McClure

…......................................To Koko, for her patience.

ACKNOWLEDGEMENTS

I would like to thank Dr. Ye Duan for a series of interesting classes that provided the basis for the work in this thesis. Through his teachings I have learned more about mathematics and modeling than I had thought possible. I would like to thank Dr. Bruce McClure for providing me with the project for this thesis and for having such awesome initials. I would like to thank Dr. Esteban Fernandez for his work in producing so much data. I hope he is doing well at his new job in California.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

ABSTRACT

Pollen tubes are delivery mechanisms in a plant's reproductive cycle. Morphodynamic analysis of these microscopic structures provide biologists with insight into the inner workings of these structures. This thesis presents three methods of computer-aided analysis of two-dimensional slices of a pollen tube in medium. First, visual comparison of these plant structures can be simplified by straightening and reorienting the pollen tube. The pollen tube must be identified and extracted from its original image. Key features may then be identified and an abstracted version of the image may be generated. Next, motion analysis may be performed on the structures within the pollen tube. Six methods of point feature detection algorithms are discussed. Correspondence tracking is used to track features from frame to frame. Average optical flow identifies general motion within the pollen tube. The results of point feature tracking and optical flow are compared on two data sets, each marking different organelles within the pollen tube, in order to learn more about the nature of the internal structures in the pollen tube.

Introduction

The study of plant morphodynamics examines changes in the shape and structure of the plant over time, such as the shape of leaves or the length of the stem. The microscopic processes that formed these structures are largely ignored. The analysis of one simple structure called a pollen tube has given insight into how the growth process works at a microscopic level.

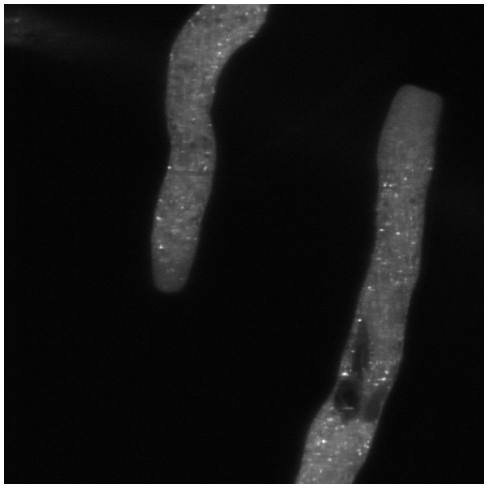

Figure 1: Confocal microscopy image of two pollen tubes marked with Syp21-GFP.

A pollen tube is a hair-like structure that grows outward from a pollen grain. It acts as a microscopic delivery system in the plant's reproductive cycle that injects sperm cells into the pestle of the female plant. Pollen tubes grow only from the tip. Once the internal structure has formed they remain fairly static. Using fluorescent markers, the internal structures of the pollen tube are highlighted so they may be studied individually. Then, a laser scanning microscope is used to create movies of the living pollen over a period of minutes, recording the motion of the highlighted structures as bright dots on a gray background.

The purpose of this paper is to describe automatic methods of extracting motion and shape information from these pollen tube movies. Research for this paper applies existing methods to data that has commonly been analyzed manually. Chapter 1 will detail methods for detecting and straightening the pollen tube images so that pollen tubes

in different movies may be more easily compared visually. Chapter 2 will describe methods of feature detection, feature tracking and motion segmentation. Chapter 3 will introduce techniques for identification and comparison of biological markers. Chapters 4, 5 and 6 will revisit these topics to describe the implementation and results of the algorithms presented.

The Data, from a Computer Scientist's Point of View

The research for this paper used movies generated by confocal microscopy. The experiments involved expression of fluorescent fusion proteins to mark specific biological structures. Laser light and confocal optics have been used to recover fluorescent light from tightly defined regions of a three dimensional target. One or more beams of laser light were scanned across the target in a two dimensional plane. Images are taken at defined intervals. The data produced by each laser is saved as a separate color channel in the resulting image stack. The stack of two-dimensional slices obtained at intervals provides a third, temporal, dimension to the data.

The movies for this paper used a 488nm laser and a 532 nm laser. The 488 nm laser excites a green fluorescent protein (GFP) variant causing it to fluoresce in the 495-525nm range. [Tsien-1998] The 532 nm laser excites a red fluorescent protein (RFP) that was not used for image analysis.

Pollen tubes have been marked by a fluorescent protein fusion, such as Syp21-

GFP, that causes specific structures to glow brightly under laser light. These structures show up as bright spots or milky patches on the final image. Some of the marker dyes used are: Syp21, Syp41 and Ara7. When attached to GFP, they are denoted Syp21-GFP, Syp41-GFP and Ara7-GFP. Specifically, Syp21-GFP labels a class of membranous vesicles called the prevacuolar compartment that is important for movement of soluble proteins to the vacuole. GFP fades quickly when exposed to intense laser light so images taken later in the movie are dimmer. All the dyes used in this paper are attached to GFP.

Differential interference contrast (DIC) is another microscopy method used for taking an image of the target. DIC can be used concurrently with confocal fluorescence. DIC, which does not use laser illumination, produces a false 3-D effect that makes normal image processing very difficult. Color channels containing DIC information are ignored.

The Zeiss line of confocal laser scanning microscopes (LSM) save their movies to an extended, multi-image TIFF format with the default extension, .LSM. This format contains a block of microscope status information under the TIFF tag, 34412. Depending on the number of lasers used to generate the movie, the LSM file can have 1, 2, 3 or more color channels. The movies used in this paper have only 1 or 2 color channels because only two lasers were used.

A complete format for the LSM files [Zeiss-LSM Format] was obtained from the IDL Goodies web page by Karsten Rodenacker of the German Research Center for Environmental Health. The format was provided to him by Zibigniew Iwinski of Carl Zeiss International. While the LSM format is an extension of TIFF, the Silicon Graphics,

Inc. TIFF library [Silicon Graphic-LibTIFF] is incapable of reading the format when two color channels are used. The National Institute of Health software, ImageJ, [NIH-ImageJ] can read the 2-color channel format with a plugin called LSM Reader, [NIH-Image Reader] but ImageJ and LSM Reader are written in Java and inaccessible to C++ code. Fortunately, the LSM movies used in this paper are uncompressed and can be read using the TIFF specification and a little work.

LSM movies used here are potentially 4-dimensional: 3-spatial dimensions and 1-temporal dimension. This paper focuses only on movies with 2-spatial dimensions and 1-temporal dimension. The Z-dimension is limited to 1 for simplicity. Any of the techniques used in this paper may be extended to 3-spatial dimensions.

Making movies in 3-dimensions takes time. When the research for this paper began, a higher frame rate was considered more important. Early movies were taken at 15 – 20 frames per second. During the research it was determined that a frame rate of 3 – 5 frames per second rate was sufficient. All the three-dimensional images taken in the early stages of research showed significant jitter, even at low frame rates. These images were also taken using DIC mode and could not be used. More work must be done to determine the best method of producing three-dimensional movies.

The subject of the LSM movies used in the research for this paper is tobacco pollen tubes. When the pollen grains of flowering plants germinate, they form pollen tubes that transport sperm cells to the egg cell in sexual reproduction. [Lord and Russel-2002] Tobacco was used because the structures are especially large and easily manipulated. The pollen tubes were also grown in a synthetic medium, making the

background of the movies black. With so much separation between foreground and background, the pollen tubes could be easily detected.

This paper is primarily concerned with tracking and comparison of structures within these pollen tubes. The biological origins of the images do not have a direct impact on the workings of the algorithms presented. Only the techniques used in straightening the pollen tube and detecting the tip are a direct result of expectations about the pollen tube's shape and structure. The feature detection and motion capture algorithms presented in Chapter 2 as well as the identification algorithms presented in Chapter 3 work independently of the pollen tube's shape and orientation. More expectations are placed on the microscope operator to produce trackable images then are placed on the pollen tubes themselves.

Well-behaved Pollen Tubes

Reliable image analysis requires well-behaved pollen tubes. A pollen tube is alive. Pollen tubes grow from the tip and some will push their shank around as they grow. This causes movement in the movie that is not easy to handle. The microscope operator also may accidentally bump the microscope, causing a wobble in the movie, or he might move the window being viewed. It is best to simply discard the difficult data where the pollen tube is not well-behaved in these ways.

A well-behaved pollen tube will not move laterally within the foreground mask

defined by the first image in the movie. The tip of the pollen tube must also be visible and not touch the edge of the image. The pollen tube may, however, still be well-behaved if the tip grows off the edge of the image. The lateral motion of the pollen tube could be detected and accounted for by searching for global motion of the foreground mask of each image. This introduces its own set of problems. Optical flow would miss any motion that happens in the direction of the pollen tube shank. Even if the tip were present, the foreground mask might not be able to tell the difference between motion of the frame and a growing pollen tube.

One possibility is to find the foreground mask of each image in the movie and straighten the pollen tube based on that mask. If the pollen tube grows at the tip, though, this would introduce motion in the direction of the shank in the straightened image. Also, the tip of the pollen tube could easily grow off the edge of the image, though the original location of the tip could be retained and projected onto the new center line. Motion that occurs when the operator bumps or moves the frame is global and should be detectable with optical flow. These problems have not been solved in this paper.

Methods of Abstraction

Two methods of abstraction are presented in this paper: abstraction of shape and abstraction of motion. Chapters 1 and 4 will break down the shape of the pollen tube and allow them to be compared visually. Pollen tubes grow in a tangled mass, spreading out

in all directions from pollen grains scattered in a medium. The shank of the pollen tube may curve and bend making the motion within one tube difficult to relate to another. More than one pollen tube may be present in a single movie and even those are not easily compared. If straightened, two pollen tubes may be compared side-by-side without the viewer needing to interpret the relative motion. This is shape abstraction.

Shape abstraction is accomplished by first detecting a pollen tube, extracting key features and then generating an idealized, straightened version where the tip is always at the top. Detecting and extracting individual pollen tubes from their movie is a way to eliminate minor differences such as orientation and scale without drastically changing the nature of the movie. The algorithm is broken into six steps:

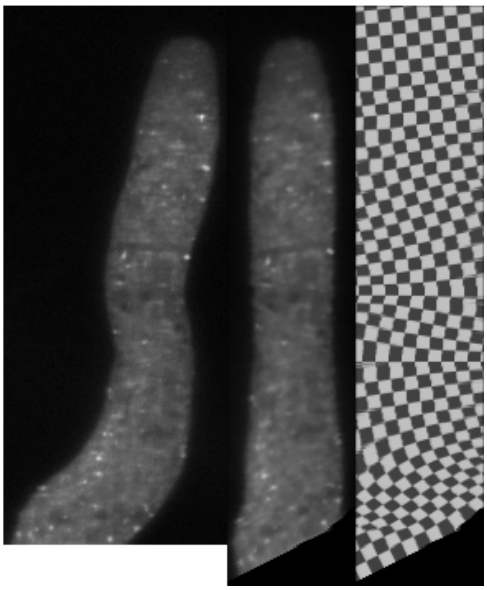

Figure 2: Left to Right: 1) The original pollen tube, 2) the straightened pollen tube generated by the algorithm presented in chapter one and 3) a checkerboard showing the distortion caused by the straightening algorithm.

1. A foreground mask is created that separates the pollen tubes from the background.
2. Connected foreground pixels in the mask are grouped so that the number of pollen tubes in the image can be counted automatically.
3. The closest distance to any edge in the foreground mask is calculated to create a distance field.
4. The ridges in the distance field, which represent the maximum distance from any

edge that falls entirely within the foreground, are then extracted as curved lines.

5. The two end points of this center line are tested to find the tip of the pollen tube, whether it exists on the image.
6. The center, curved line of each pollen tube is mapped to a straight line with the pollen tube tip at the top and texture mapping techniques are used to draw a new image of the straightened pollen tube.

The second abstraction method is motion abstraction. Chapters 2 and 4 will describe two methods of extracting motion data from the pollen tube movies. The first method involves tracking individual features to detect local motion. Six feature detection algorithms will be described and correspondence tracking of point features will be discussed. The second method uses the average optical flow to find general paths in the data to detect global motion. This second method is analogous to finding the lanes of a highway by averaging the motion of the cars.

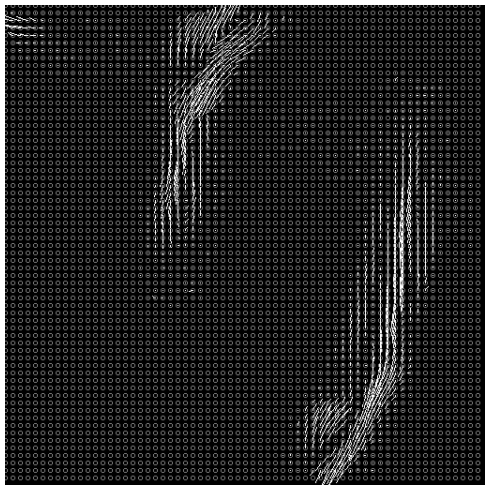

Figure 3: A vector field showing the average direction of motion calculated by Lucas-Kanade Optical Flow.

Chapters 3 and 6 will compare the two methods of motion abstraction. The goal is to differentiate between two markers, Syp21-GFP and Ara7-GFP, to see if they behave differently. To that end, a statistical classifier will be defined that will attempt to identify these markers based solely on the detected motion within the pollen tube. Rather than

creating the classifier as a goal in itself, the statistics used to generate the classifier, the mean and covariance, can give some insight into the function of the structures with the fluorescent markers. A feature vector based on comparisons between tracked features and optical flow will be described.

Finally, a programmer cannot anticipate end user needs. Users need the ability to interact with the code in a way that makes sense to them. Thus, an interface was designed that allows the user to manipulate one or more pollen tubes together and to animate and visually compare them. A more complete explanation of this user interface will be described in Appendix A.

Chapter 1: Detection and Straightening of Pollen Tubes

Foreground Mask

Automatic detection of pollen tubes in the image requires a mask that identifies the pollen tube as a foreground object. A good foreground mask should have smooth edges that accurately represent the shape of the pollen tube. The vacuoles, dark areas within the pollen tube, should be filled in. This foreground mask is a binary image where 0 represents an area of background and 1 represents an area of foreground. This mask may be obtained by applying a threshold to the original image. The threshold value varies by image in part because dyes are fading as the images are taken and in part because the living pollen tubes are moving and changing as time goes on. Dozens of threshold algorithms exist.

Figure 4: A Foreground Mask produced by applying a median filter with a 5-pixel radius window to the original image and then using the expected intensity of the image as the threshold value.

In 1979, Otsu developed a threshold algorithm that minimized the weighted sum of within-class variances between the foreground and background pixels. [Otsu-1979]

$$T_{opt} = arg\,max\left[\frac{P(T)[1-P(T)][m_f(T)-m_b(T)]^2}{P(T)\sigma_f^2(T)+[1-P(T)]\sigma_b(T)}\log\frac{1-P(T)}{P(T)}\right]$$

Here, *P* is the cumulative probability function.

$$P(g)=\sum_{i=0}^{g} p(i)$$

m is the mean of the foreground and background functions.

$$m_f=\sum_{g=0}^{T} g\,p(g) \qquad\qquad m_b=\sum_{g=T+1}^{G} g\,p(g)$$

σ^2 is the variance of the foreground and background functions.

$$\sigma_f^2=\sum_{g=0}^{T} [g-m_f(T)]^2\,p(g) \qquad\qquad \sigma_b^2=\sum_{g=T+1}^{G} [g-m_b(T)]^2\,p(g)$$

For the pollen tube image shown in figure 1, the threshold chosen by the Otsu algorithm includes details, such as the vacuoles, as background. These vacuoles must be labeled as foreground. Otherwise, an internal boundary will be formed inside the pollen tube around the vacuoles or they will appear in the outer boundary as a cavity. Either way, the distance field will find a curve that encircles the vacuole. This curve no longer follows the center of the pollen tube and is not useful.

In 1983, Rosenfeld and De la Torre developed another algorithm that maximized the difference between the histogram value and its convex hull. [Rosenfeld and De la Torre-1983]

$$T_{opt}=arg\,max[Hull(g)-p(g)]$$

Here, *g* is the pixel intensity, *p* is the probability mass function of the pixel intensities and *Hull* is the convex hull of the probability mass function. The threshold value is the pixel intensity where the probability mass function differs most from its convex hull. In the case of confocal images this intensity is usually very close to the background intensity due to a high peak at the background intensity combined with much lower peaks at the foreground intensities. This algorithm does not find the vacuoles unless their intensity is very close to the background intensity.

The mean intensity also works well as a threshold value for confocal images.

$$T_{expected} = \frac{1}{N}\sum_{i=1}^{N} g_i$$

Due to noise in the image, the resulting mask produced by any of the above algorithms has jagged or fuzzy edges. This can be limited by first applying a median filter to the image. The median filter smooths out the image with minimal movement of the edges within the image.

A median filter tests the pixel intensities in a window of radius *n* around a center pixel. These intensities are sorted and the median value is selected.

$$median\,index = \frac{1}{2}((2n+1)^2+1)$$

Figure 4 shows the results of the foreground mask on the pollen tube image in figure 1. Notice that the dark regions of the pollen tube on the right are filled in and the fading edge of a third pollen tube has appeared in the top-left.

Distance Field

The first key feature of a pollen tube is its shape. In its simplest form, the pollen tube is a curved line. This model can be defined by the center line of the pollen tube. The center line is a line equidistant from any edge, where the edge will be defined by the foreground mask. After finding the center line of the pollen tube, the coordinate system of the resulting curve can be used to build a straightened version of the image. In the foreground mask, the boundary between the foreground and background can be considered an edge. A distance field solves for the minimum distance from any pixel to any edge in the image, so the maximum value of the distance field inside the foreground also defines the center line of the pollen tube. This distance field also provides the width of the pollen tube at any position along its length. This can be used to determine the width of the straightened image.

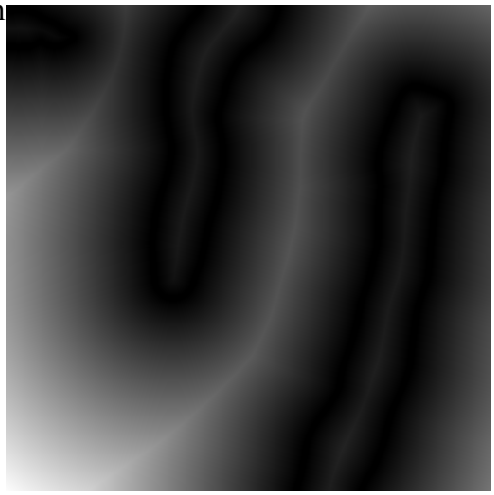

Figure 5: An intensity map of the distance field generated by using the distance value at each pixel as the intensity then scaling this intensity value to a range between 0 and 255.

The edge of the foreground mask for a pollen tube can be considered a closed boundary in two dimensions separating the foreground from the background. The shortest distance from this boundary to any point in space can be found by shrinking or expanding the boundary at a constant speed in a direction normal to the curve. The time it takes this advancing boundary to reach the point in question is directly related to the

distance of that point from any edge in the foreground mask. In this formulation, the boundary is not explicitly defined. Instead, the time when the boundary would cross an intersection on a discrete grid is recorded. Negative time is considered to be inside the original boundary. Positive time is considered to be outside the original boundary. This describes the boundary value formulation of the level set method. [Sethian-1999]

The time it takes for the boundary to cross any point within the surrounding space is a function of the position within that space:

$u(x)$, where $u=0$ at the initial boundary position

A speed function that relates the motion of the boundary to the crossing time can be defined, given that *distance = rate * time*.

$$|\nabla u(x)|=f(x)$$

Information about the boundary position is pushed outward as the boundary expands from its initial, known position to the rest of the space. Taking a note from Sethian, the propagating boundary can be pictured as a wave. [Sethian-1999] Consider the following one-dimension wave equation:

$u_{x+t}(x,t)+u_x(x,t)=0$, where $u(x,0)=f(x)$, or $u(x,t)=f(x-t)$

$$u_{x+t}=-u_x$$

This means that the solution to u at any point x at time t is the same as the initial value of u at the point x – t. The solution is considered dependent.

The solution to u at time t + k can be expanded using a Taylor series as follows:

$$u(x,t+k)=u(x,t)+u_{x+t}(x,t)k+O(k^2)$$

where $O(k^2)$ includes all terms of order k^2 or higher. Rewritten,

$$u_{x+t}(x,t)=\frac{u(x,t+k)-u(x,t)}{k}+O(k)$$

This is known as the forward difference operator because it uses information from the next time step. Instead, this equation can be rewritten using information obtained in the previous time step.

$$u_{x-t}(x,t)=\frac{u(x,t)-u(x,t-k)}{k}+O(k)$$

This is known as the backward difference operator. This operator follows the concept of propagating information from the known to the unknown or moving information *upwind*.

The equations above are for the one dimensional case. These equations extend readily into the spatial domain in any number of dimensions because the derivatives with respect to x, y and z are treated independently. The trick with higher dimensional spaces is knowing which direction the information needs to propagate from. To make things easy, values in the grid will be normalized to [0,1] and the grid will be initialized to 1.

The update equation becomes

$$u(x,y,t+k)=u(x,y,t)-\Delta t[MAX(f(x,y),0)\nabla^{+}+MIN(f(x),0)\nabla^{-}]$$

$$\nabla^{+}=[MAX(D^{-x},0)^2+MIN(D^{+x},0)^2+MAX(D^{-y},0)^2+MIN(D^{+y},0)^2]^{\frac{1}{2}}$$

$$\nabla^{-}=[MAX(D^{+x},0)^2+MIN(D^{-x},0)^2+MAX(D^{+y},0)^2+MIN(D^{-y},0)^2]^{\frac{1}{2}}$$

$$D^{-x}=\frac{u(x,y)-u(x-k,y)}{k} \qquad D^{-y}=\frac{u(x,y)-u(x,y-k)}{k}$$

$$D^{+x}=\frac{u(x+k,y)-u(x,y)}{k} \qquad D^{+y}=\frac{u(x,y+k)-u(x,y)}{k}$$

In order to model the propagation of this boundary wave, Sethian introduced the Fast Marching Method. Fast Marching performs the above equation at each pixel, pushing information outward as the boundary expands. All this happens in one pass using a heap structure.

A heap is a binary tree with all nodes filled from left to right on each level. The nodes on the tree are sorted so that, in this case, the minimum value in the tree is always at the top. Any child node in the tree is required to be larger than its parent. The left child must be smaller than the right child. Otherwise, the data in the tree is unsorted. This structure can be implemented efficiently using an array where the left child position is the twice the parent position and the right child position is one higher than the left. The value of the heap is the distance of a given pixel from the nearest edge. Only pixel positions with calculated distances will be pushed onto the heap.

When initializing this structure, all the boundary pixels are pushed onto the heap. For these positions, the distance from the edge is found explicitly or simply set to 0. In

the program loop, the minimum value is popped off the top of the heap and its eight, surrounding neighbors are tested. All the positions that are either not already in the heap or have not already been checked are pushed onto the heap. As each position is pushed onto the heap, its distance is calculated and saved. When the heap is empty, the distance field has been solved.

In order to determine which pixels have been used, a flag array the size of the image with an additional one-pixel boundary can be kept. As positions are pushed onto the heap, the corresponding flag is set. The one-pixel boundary should initially be set so that no bounds checking is necessary when testing whether a position is available.

Another solution to the distance field is given by Zhao in his paper on Fast Sweeping. [Zhao-2004] This equation is used for convenience because its two-dimensional case is given explicitly in the paper. In the two dimensional case,

$$a = min(u_{i-1,j}, u_{i+1,j})$$
$$b = min(u_{i,j-1}, u_{i,j+1})$$

$$\bar{x} = \begin{cases} min(a,b) + f_{i,j}h & \text{if } |a-b| \geq f_{i,j} \\ \dfrac{a+b+\sqrt{2f_{i,j}^2 h^2 - (a-b)^2}}{2} & \text{if } |a-b| < f_{i,j} \end{cases}$$

The Fast Sweeping algorithm normally applies this solution by making several passes over the data. The passes need not be performed in any order as long the information in the boundary is pushed outward in all directions. While Fast Marching is O(n log n) and Fast Sweeping is O(n), the size of the images used means that the heap does not introduce an unreasonable burden compared to the minimum of five passes across the entire data

for two-dimensional images needed by Fast Sweeping. Therefore, the Fast Sweeping equation may be used in one pass with the heap structure from Fast Marching.

Center Line

The center line is the set of pixels equidistant from any edge in the pollen tube. This can be defined as the set of foreground positions at the maximum distance from any edge. The gradient of the distance field is a smooth ramp that extends outward from the boundary of the foreground mask. Any point that is equidistant from two edges forms a ridge. This ridge follows the exact center of the pollen tube as defined by the foreground mask.

The solution for the center line is a curve defined by the positions on the distance field gradient where the direction of the gradient reverses. A fairly simple method for extracting the center line from the gradient is apparent from an intensity map of the gradient magnitude. The ridges and boundaries in the distance field appear as dark lines in a field of gray as seen in figure 6.

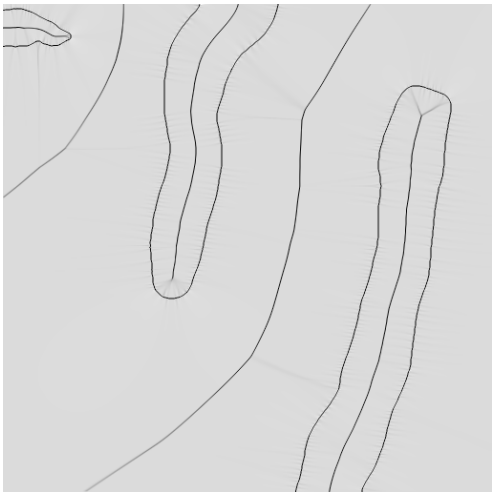

Figure 6: An intensity map of the gradient magnitude of the distance field.

When this gradient intensity map is inverted and a threshold is applied, the resulting image shows a series of white lines on black. If any of these white pixels or any

of their surrounding eight neighbors correspond to a background pixel in the foreground mask as defined above, that pixel is not a center line and can be ignored. A connected component algorithm will then walk along the line in both directions by choosing one of the eight neighbors of the center pixel that meets two criteria; it is white and has not already been examined. As this algorithm progresses, the end points of a curved line are pushed to their farthest distance along the line. As each pixel is tested, it is flagged so the algorithm does not try to use that pixel again. Setting the tested pixel to non-white allows it to be flagged without requiring extra memory. Many pairs of end points may result once all the pixels in the image have been tested.

On this first pass, the endpoints of the center lines are more interesting than the lines themselves. When the threshold was applied to the gradient magnitude image, gaps were created. If only a pixel or two separates the line segments, the closest pair of end points are matched. The line segments are then merged by drawing the missing pixels between these matched end points and the process is repeated until all line segments are too far away from each other. If the line segments are too short they may be difficult to merge because both end points may be considered close. This is the case with one-pixel line segments.

The center lines must be only one pixel wide at any point, otherwise the algorithm that walks the eight-connected path will get lost, breaking the center line. If the logical line passes between two pixels in the gradient magnitude image, both pixels will be white. Therefore, the lines must be artificially thinned. This is done by removing any pixel with two or more four-connected neighbors. Once all the end points have been

matched, the center lines may be walked again to build one or more arrays of points that are the center lines.

Pollen Tube Tip

A pollen tube has a nearly constant width along its length, ending in a semi-circular region at the tip. The extracted center line has two end points, either of which may be a pollen tube tip. The distance field requires the ridge that formed the center line is equidistant from all edges, placing one of the end points near the center of the pollen tube tip. Therefore, the end point in the center of the pollen tube tip will never touch the edge of the image. Half the end points in an image can usually be eliminated by testing whether the value of the distance field is greater than the distance to the edge of the image.

Figure 7: Hough Circle Transform parameter space. The radius of the tip of the center pollen tube was selected.

The generalized Hough transform [Ballard-1980] can be used to detect the pollen tube tip. The original Hough transform detects colinear points by finding the parameters – slope and intercept – of a line that passes through those points. A vote is saved for each set of parameters that would produce a given line. The line with the most votes wins. An accumulator grid in parameter space, where the X-axis is the slope and the Y-axis is the

intercept, can define all possible votes. Because the intercept is unbounded, a small range of possible values must be selected to search. The slope is an angle and may be restricted to some range of π radians. The grid may be as coarse or as fine as needed, but a fine grid with a very short step, while more accurate, will take longer to process and require more memory.

The generalized Hough Transform searches for the center point of some predefined structure based on the edges of that structure. The minimum number of edges needed to describe a structure is called a primal sketch. The goal is to find a semicircle oriented in any direction, so the primal sketch used is a circle of some radius. The generalized Hough transform can also find the radius of the circle, but the distance field provides some prior knowledge. The value of the distance field at the end point being tested determines the expected radius.

The generalized Hough transform uses orientation information from the image gradient to determine the best direction from the edge to find the center point. The radius, or a good approximation of the radius, is already known and the gradient of the foreground mask provides a well-defined edge. Without taking the direction of the gradient into account, the center of a circle defined by a known edge may be in any direction the expected radius away. A Bresenham circle with the expected radius may be drawn in parameter space to add the vote for that position to the accumulator grid. The magnitude of the gradient at that position will be the value of the vote. This means that only pixel positions that are on the edge get a vote. The magnitude of the gradient at any pixel position inside the foreground or background is zero. This voting method is similar

to the original Hough transform in which every possible position in parameter space was voted on rather than voting a few times based on the orientation of the gradient. Figure 5 shows the result of this method.

When drawn as an image, parameter space looks like a series of ghostly curves with a few bright spots sprinkled throughout. These bright spots are probable locations for the correct center point of the circle. The end point being tested should have a large number of votes compared to the rest of parameter space. However, the pollen tube tip is not exactly a semicircle. Some pollen tubes have square tips with rounded edges. Others have completely rounded tips. The effect this will have on the Hough transform is that the center point will not be at a single point. The maximum value that is some radius from the end point must be found. If this value is greater than some minimum, the end point may be considered the tip of the pollen tube.

Texture Mapping

Texture mapping is the process of mapping pixel intensities from a source, u-v image space to a destination, x-y image space. [Heckbert-1989] This mapping is a function of the source coordinate system. The positions of the pixel intensities to be written to the destination image are found by first knowing the position in the destination image to be written and how that position relates to the destination coordinate system then finding a pixel intensity the same distance away in the source coordinate system. The method of

using the destination position to determine the intensity to read from the source is known as *screen order* or *screen scanning*.

The mapping between source and destination is generally described using an affine transform. An affine transform is a three dimensional matrix, one more dimension than the source image, that combines rotation, translation, scale and shear in one operation. This transform affects the entire image. However, the center line is a curve that will be mapped to a straight line. The transformation is determined by the tangent and normal vectors of the curve, so the transformation at each position along the curve must be calculated.

The first step is to define the target, the dimensions of the pollen tube to be drawn. The distance field represents the minimum distance from a given point to any edge. This value can used to determine the dimensions of the target image because the center line is, by definition, centered between the two long edges of the pollen tube. The width of the target image shall be at least double the maximum value of the distance field at any point along the center line. This will put the boundary of the pollen tube on the exact edge of the target image. The length of the center line must be determined. Finding the length of the center line is trivial as it is the length of the array used to keep the points. The value of the distance field at each end point must be added to the length in order to include enough space for the tip and any slant present in the base of the pollen tube. If desired, more pixels may be added to these dimensions in

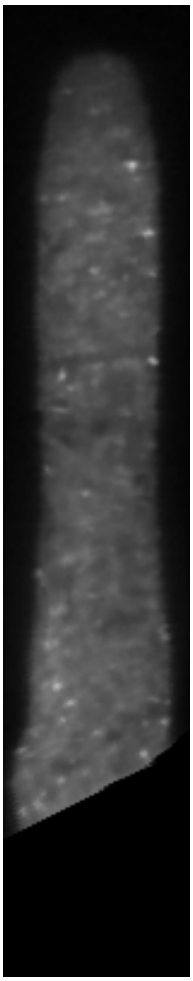

Figure 8: Straightened pollen tube.

order to pad the target image.

Now that the dimensions for the target image have been determined, the internal features must be defined. The destination center line will be a straight line along the center of the target image with the end point representing the tip at the top. This end point shall be a distance equal to the value of the distance field at that point below the top of the image so the tip of the pollen tube is visible. This straight line represents the Y-axis of the destination image. The Y-axis can be represented by the vector <0,1>. Therefore, the X-axis can be represented by the vector <1,0>. These vectors may be scaled to change the size of the destination image as described below.

The u-v coordinate system for the source image is not so simple. At any point along the center line, the tangent to the curve will be the new V-axis with the end point representing the pollen tube tip representing "north" in the coordinate system. The U-axis can be found by taking the cross product of the V-axis with the normal of the image plane. This creates a new vector:

$$\vec{U} = [V_y, -V_x]$$

Using these two vectors, the center line may be sampled at intervals along the curve and multiples of the tangent and normal vectors may be used to find the position within the source image.

Not every pollen tube will have the same resolution. In order to compare pollen tubes, scaling the destination image up or down is an important step. When scaling the image, the exact voxel size in meters may be extracted from the LSM information

contained in the file. This value is divided from the desired scale to create a real scale for the image.

Finding the local tangent vector is problematic because the center line is kept as an array of points. The method used here is to take two points on the center line that are ten pixels apart and draw a line between them. Ten pixels is an arbitrary number, but one that seems to work well. This line will be used as the tangent line. At any point outside the known area around the center line itself, such as the very tip of the pollen tube, the center line is assumed to move straight along the last known tangent line. The last ten pixels in the center line may, therefore, be used as the tangent vector no matter how far away from the end point the texture is drawn.

When sampling the data from the source image, a pixel is chosen and its position is compared to the defined axis. If the pixel Y-position is above the top end point, a multiple of the V-axis is used to determine the position from the pollen tube tip end point in the source. Otherwise, the index of the pixel on the center line is used to determine the position on the V-axis. A multiple of the U-axis will always be used to relate the U-axis with the X-axis. The intensity must be interpolated from the fractional position in the source.

Using a single sample from the found position can cause an undue amount of aliasing in the destination image. Instead of sampling each pixel in the source image once, they may be sampled many times to get a more accurate intensity. Using a 3x3 window around the original pixel the intensities found in source may be added in a 3x3 window around the position in the destination image. This process is repeated for the

entire destination image so that every pixel in the image (except the border pixels) is sampled nine times from nine different locations in the source. After the destination image has been constructed, the real intensities may be found by averaging the accumulated intensities. These sample points may also be given a weight to give more emphasis to the center sampled point. The result is either divided by 9 for the average pixel intensity or by the total weight. The weight used may be a Gaussian mask so that the sum of the weights is 16. This turns the division into a bit shift, speeding up the average.

Chapter 2: Feature Detection and Tracking

Feature Detection Algorithms

The goal of feature detection is to find interesting regions of an image that would be invariant to simple changes in the structure or orientation of the image. A feature in one frame of animation should ideally be detectable in the next frame. Many feature detection algorithms exist and there are many types of features. This paper will focus on a set of algorithms that detect point features.

Detecting point features requires several assumptions. First, the image must contain point features. Second, noise will not be a significant factor. Third, the entire movie will have constant illumination so that point features in early frames will also appear the same in later frames. Fortunately, the movies produced using confocal microscopy, and including either the Syp21-GFP, Syp41-GFP or Ara7-GFP marker dyes do contain small, globular structures that qualify as point features. Finally, the illumination of the image does change very gradually over the movie. This last is due to the dye fading out. The level of illumination is similar between any two consecutive frames, unless the microscope operator changes the laser intensity.

Small changes in illumination equate to large changes in the numbers of features being detected. As contrast between the features and background flatten out, the threshold used to determine the featureness of the pixel can remove too many potential features. No automatic method of detecting the proper contrast threshold has been

provided to keep the number of detected features consistent between frames. The GFP markers fade over time due to the bleaching effects of the high intensity laser used to illuminate the pollen tube.

In feature detection, three types of areas exist: walls, edges and corners. A wall is an area of high self-similarity, meaning all the pixels surrounding the center are very close in intensity to the center. Walls are not very interesting because little information can be obtained from them. Edges show a distinct difference between pixel intensities to one side of the center and pixel intensities to the opposite side. Corners show a distinct difference in pixel intensities in more than one direction.

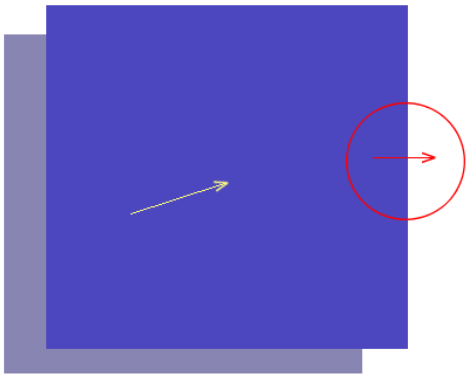

Figure 9: The aperture problem. The blue square has moved in the direction of the yellow array, but only the motion in the direction of the red area can be detected inside the red circle.

Corners are important for detecting motion because the direction of motion in a corner can be followed in two dimensions. The direction of motion in an edge may only be followed in the direction of the image gradient. This is known as the aperture problem as seen in figure 9.

Local Extrema

The simplest and most naïve type of point features that can be detected are bright spots on the image. The algorithm itself is very simple. Within a small window centered around a given pixel, all the pixel intensities must be less than or equal to the intensity of the center pixel – white features. This is the local maximum. The algorithm can also be reversed and accept features that are the local minimum – black features. The algorithm may be made more robust by requiring a minimum intensity difference between the center pixel and the average of the surrounding pixels. The results can further be improved by ignoring any features detected in the background.

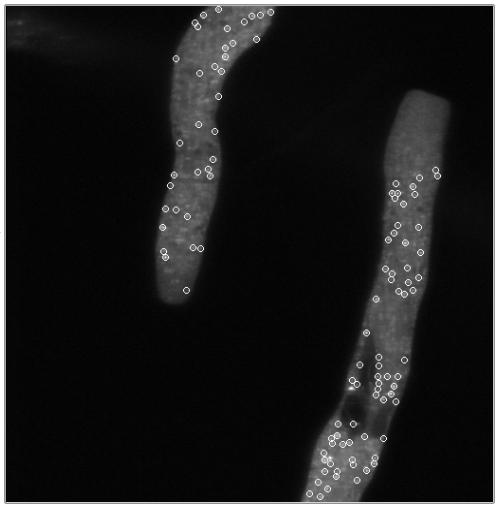

Figure 10: Features detected by Local Extrema.

Placing the worth of a feature solely on the difference in intensity between a center pixel and its neighbors makes this algorithm highly susceptible to noise. It also requires features to be points and will not detect physical features that are too large. If all the neighbors of a pixel are bright, the algorithm cannot tell the difference between a feature and a blank wall. More well-considered algorithms exist.

Moravec's Interest Operator

In 1979, Hans Moravec developed a visually guided, roving robot. The interest operator was intended to find unmoving obstacles in the robot's path. The video camera this robot used as an eye was simple and prone to noise. So combining multiple images would increase the quality of the image used for feature detection. The image resolution was only 280 x 240 and up to 30 consecutive images needed to be averaged to achieve reasonable quality. [Moravec-1979]

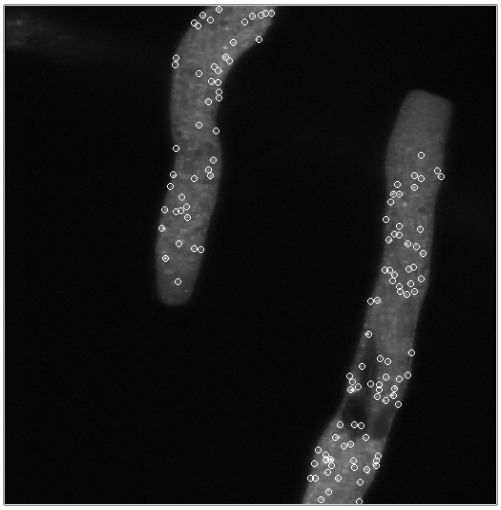

Figure 11: Features detected by Moravec's Interest Operator.

The interest operator measures areas of low self-similarity, defined as directional variance over small windows in the image. If all the pixels in the local area around the center pixel are the same, the area is considered to have high self-similarity. If a large change in intensity is apparent between the center pixel and its neighbors, the area is considered to have low self-similarity.

The self-similarity measure is given by the following difference, where I is the intensity of the image at each point in a small window around the center pixel.

$$E_{x,y} = \sum_{u,v} [I_{x+u,y+v} - I_{u,v}]^2$$

The window around the center pixel is compared to the window around each adjacent

pixel. The value of the operator is the minimum variance between windows. The orientation of the operator is the direction from the center pixel to the adjacent pixel with maximum variance. Features are the local maximum of the operator values.

Testing adjacent pixels means that only 45º angles are handled, making the interest operator anisotropic or directionally dependent. The operator is also very sensitive to noise in the image. The operator does not perform well under affine transforms and responds too readily to edges. The Moravec operator is interesting for historical purposes and should be expected to perform better than Local Extrema but not as well as the algorithms that follow below.

Harris/Plessey Corner Detector

The auto-correlation detector developed by Chris Harris in 1988 at Plessey Research Roke Manor was an idea formed from research into 3-D motion in natural scenes. [Harris and Stephens-1988] Borrowing from the ideas of Morevec, Harris set out to find corners in an image. Whenever Moravec's detector found an edge, the value of the operator would be a small parallel to the edge but large in a perpendicular direction. A corner would

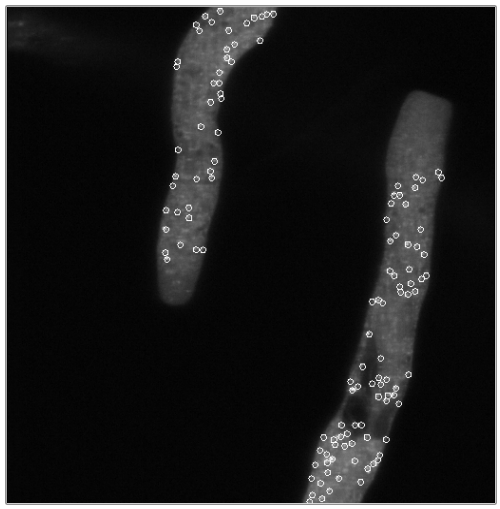

Figure 12: Features detected by Harris/Plessey Corner Detector.

show a large response in any direction.

Instead of testing the variance at 45º increments, Harris found the first order derivative of the image intensities or the gradient of the image. The gradient is capable of detecting small changes in the orientation of image structures and less likely to be anisotropic. Noise reduction is achieved by convolving the image with a Gaussian kernel. Edge responses can be detected and ignored.

The first order derivative of the image intensities or the gradient operator produces a vector field describing the slope from the darkest regions to the brightest regions in the image. Edges and corners have a large response, represented by long vectors. Flat areas are represented by short vectors.

The gradient operator can also be found using finite difference approximation. This representation is convenient when dealing with discrete pixels within an image.

$$\nabla I = \left\langle \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right\rangle = \left\langle I_{x+1,y} - I_{x-1,y}, I_{x,y+1} - I_{x,y-1} \right\rangle$$

An error function describing the weighted sum of squared differences over a small window on the image is defined to detect corners within the gradient. Notice the similarity to the equation in Moravec's Interest Operator.

$$E_{x,y} = \sum_{u,v} w_{u,v} [I_{x+u,y+v} - I_{u,v}]^2$$

Written in terms of the gradient,

$$E_{x,y} = \sum_{u,v} w_{u,v} [x \nabla I_x + y \nabla I_y + O(x^2 + y^2)]^2$$

$$E_{x,y} = A\,x^2 + 2\mathrm{C}\,xy + B\,y^2$$

where

$$A = \nabla I_x^2 \otimes w$$

$$B = \nabla I_y^2 \otimes w$$

$$C = \nabla I_x \nabla I_y \otimes w$$

$w = e^{\frac{-(u^2+v^2)}{2\sigma^2}}$, the Gaussian kernel.

In matrix form,

$$E_{x,y} = (x, y)\, M\, (x, y)^T$$

where

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

M is the sum of gradient values over a small window of pixels around a center point. The cornerness response, *R*, for this window is

$$R = Det(M) - k\, Tr(M)^2$$

where

$$Det(M) = AB - C^2$$

$$Tr(M) = A + B$$

and *k* is some constant.

If k is small, then A and B must both be large enough to pass some threshold value. A value of $k = 0.04$ is common.

The Harris operator usually results in blobs of detected features in an area around the actual corner. Several things can be done to reduce the number of redundant features detected. The corner may be selected as the local maximum in an area of high cornerness. The local maximum may either be determined from the image intensity or the operator response value. The position of the feature may be further refined by finding the subpixel position of the highest response. This is done by solving for the peak of a three-dimensional parabola with the X and Y directions defined by the local window around the feature and the Z-direction defined by the responses at those positions within the window.

Smallest Univalue Segment Assimilating Nucleus (SUSAN)

In 1997, Smith and Brady of Oxford University developed a method for detecting corners based on the area of similarity within a circular mask. The center pixel being tested for featureness is called the nucleus. The area within the circular mask of similar intensity to the nucleus is called the Univalue Segment Assimilating Nucleus or USAN.

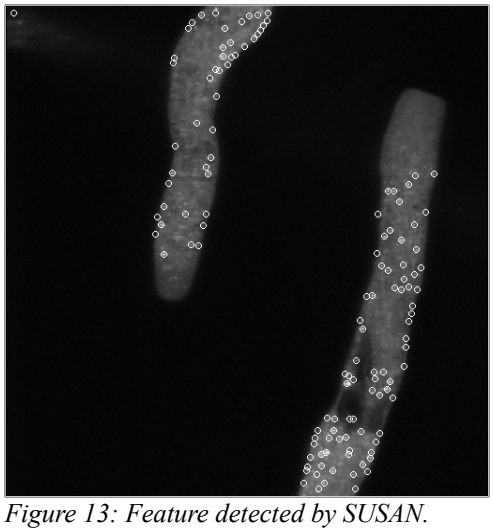

Figure 13: Feature detected by SUSAN.

Features are defined as the local minimum USAN value, the Smallest USAN. SUSAN is both an edge detector and a corner detector. As such, the algorithm exhibits a strong edge response. [Smith and Brady-1997]

The circular mask typically has a radius of 3.4 pixels requiring 37 pixels to be tested. The USAN value is the number of pixels, *r*, that have similar contrast with the nucleus, r_0. The similarity measure, where τ is the contrast threshold, is

$$c(\vec{r},\vec{r}_0)=\begin{Bmatrix}1, \text{ if } |I_{\vec{r}}-I_{\vec{r}_0}|\le\tau\\0, \text{ if } |I_{\vec{r}}-I_{\vec{r}_0}|>\tau\end{Bmatrix}$$

or for smoother results,

$$c(\vec{r},\vec{r}_0)=e^{-\left(\frac{I_{\vec{r}}-I_{\vec{r}_0}}{t}\right)^6}$$

$$n(\vec{r}_0)=\sum_{\vec{r}} c(\vec{r},\vec{r}_0)$$

The final edge response is based on a maximum number of pixels in the mask. Increasing or decreasing the value of τ will change the number of returned features.

$$R(\vec{r})=\begin{Bmatrix}g-n(\vec{r}), \text{ if } n(\vec{r})<g\\0, \text{ otherwise}\end{Bmatrix}$$

where

$g=\frac{3}{4}n_{max}$ and n_{max} is the maximum value n may take.

The algorithm described above is for the SUSAN edge detector. The corner detector adds a non-maxima suppression step that eliminates, if possible, any edge responses and uses a

smaller value of *g*.

SUSAN responds consistently to noisy image or large differences in the value of τ. This consistency is important and makes SUSAN a good candidate for the feature tracking algorithm to be described later.

Features from Accelerated Segment Test (FAST)

In 2006, Rosten and Drummond of Cambridge University developed a quick corner detection algorithm for use in real time applications. [Rosten and Drummond-2006] The method is similar to SUSAN in that it does not calculate the gradient of the image. Instead of testing all the pixels in a circular mask, only the pixels around the edge of the circle are tested. If the pixel intensities along a contiguous arc of the circle are all greater than or less than the center pixel intensity, that pixel is considered a corner. The number of contiguous pixels needed must be some significant percentage of the total number of pixels in the circle edge.

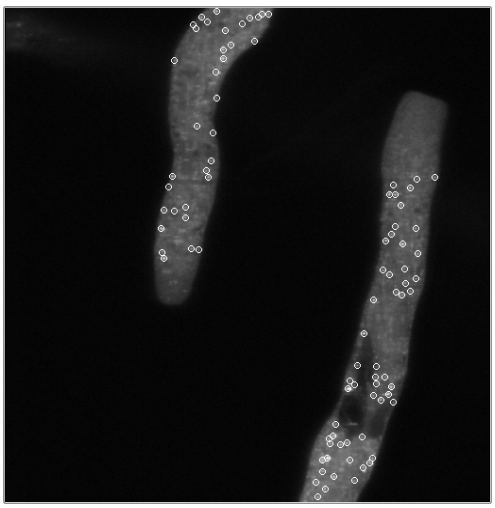

Figure 14: Features detected by FAST.

In a Bresenham circle of radius 3, there are 16 pixels to test at known locations relative to the center. If at least nine

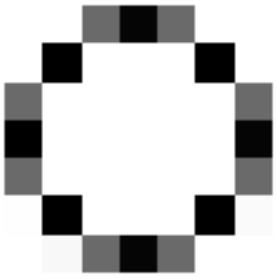

Figure 15: A Bresenham Circle of radius 3.

contiguous pixels have intensities greater than or less than the center, a feature has been detected. The algorithm allows for 9, 10, 11 or 12 contiguous pixels depending on the severity of the corner that must be detected.

The goal of the FAST project was to create a fast corner detector. The code for this algorithm was generated using an evolutionary algorithm with a fitness function based on the speed of the resulting feature detector. Regardless of how the code is generated, the resulting algorithm is the same.

The FAST algorithm does not perform well in the presence of noise as it is designed to use the fewest number of pixels possible. FAST also tends to return multiple, overlapping features if the physical features are small. To eliminate the overlap, a minimum distance function must be run against the list of detected features. Any feature that is too close to an accepted feature is ignored. This minimum distance function is not part of the original algorithm.

Scale Invariant Feature Transform (SIFT)

In 1999, David Lowe developed a scale invariant feature detection algorithm designed for object recognition rather than speed. [Lowe-1999] The algorithm can logically be broken into two parts: feature detection and feature description. This

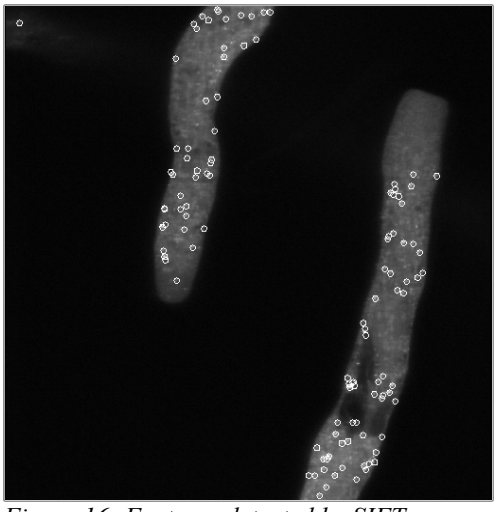

Figure 16: Features detected by SIFT.

paper focuses solely on feature detection.

The feature detection algorithm builds off work by Lindberg using Laplacian of Gaussians (LoG) for scale invariance. [Lindberg-1998] Lindberg used the Harris corner detector for feature detection. Mikolajczyk showed that the 26-connected maxima and minima of LoG produces very stable feature points. [Mikolajczyk and Schmidt-2001] Lowe introduced a finite difference approximation of LoG known as the Difference of Gaussians (DoG).

Laplacian of Gaussians is a method of determining the scale of a particular location in an image. This method involves applying the Laplacian operator to a Gaussian blurred image. The value of σ that produces the maximum Laplacian value at a given location in the image is said to be the scale of the image. The LoG of a feature found with the Harris Corner Detector results in a three dimensional feature position where the Z-coordinate corresponds to the scale, or σ, of the feature. This feature is said to be scale invariant.

As an image shrinks it loses detail in the same way that a Gaussian blurred image blends away detail. It can be shown that a Gaussian blurred image with $\sigma = 2$ has the same information as an image scaled down by half. Any value of σ between 1 and 2 is, therefore, an intermediate scale. Scale space refers to a series of Gaussian blurred images covering a range of values for σ.

An octave of scale space is described as one doubling of σ. For example, σ between 1 and 2 is one octave and σ between 2 and 4 is the next octave. The number of scales in between determines the resolution of scale space. More intermediate scales may

mean a more accurate result but the memory required goes up quickly.

$$\text{scale normalized LoG} = \sigma^2 \nabla^2 G$$

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

DoG differs from the LoG by a constant factor, *k*. The error factor goes to zero as *k* goes to 1. In practice, a good value of *k* relates to the number of scales in scale space. So, $k = 2^{1/s}$ where *s + 2* is the number of scales needed for one octave of scale space.

The maximum number of octaves possible may be determined by reducing the smallest dimension of the image by half until the dimension is only 1-pixel. The number of octaves equals the number of divisions required. In order to handle scales between 0.5 and 1, the image should first be upsampled – scaled up by two – and another octave added to the mix. Upsampling can increase the accuracy of the algorithm.

First, the Gaussian blurred image at each scale in each octave must be found. The value of sigma at each scale may be found iteratively by started with $\sigma = 1$ – or 0.5 if upsampling was used – and multiplying by *k* at each scale. Next, the difference of each consecutive pair of Gaussians in each octave must be found. Finally, for each group of three consecutive differences, the 26-connected local minima and maxima must be found in the center image of the three.

The accuracy of the detected feature location may be further increased by solving for the subpixel position of the feature in scale space. The following linear equation may be solved using Gaussian elimination.

$$M = \begin{pmatrix} D_{xx} & D_{xy} & D_{xs} & -D_x \\ D_{xy} & D_{yy} & D_{ys} & -D_y \\ D_{xs} & D_{ys} & D_{ss} & -D_s \end{pmatrix}$$

where D is the first and second partial derivatives of the three consecutive difference images around the detected feature point. These partial derivatives may be found using finite difference approximation.

$$D_x = \frac{1}{2}(I_{x+1,y,s} - I_{x-1,y,s})$$

$$D_y = \frac{1}{2}(I_{x,y+1,s} - I_{x,y-1,s})$$

$$D_s = \frac{1}{2}(I_{x,y,s+1} - I_{x,y,s-1})$$

$$D_{xx} = I_{x+1,y,s} + I_{x-1,y,s} - 2I_{x,y,s}$$

$$D_{yy} = I_{x,y+1,s} + I_{x,y-1,s} - 2I_{x,y,s}$$

$$D_{ss} = I_{x,y,s+1} + I_{x,y,s-1} - 2I_{x,y,s}$$

$$D_{xy} = \frac{1}{4}(I_{x+1,y+1,s} + I_{x-1,y-1,s} - I_{x+1,y-1,s} - I_{x-1,y+1,s})$$

$$D_{xs} = \frac{1}{4}(I_{x+1,y,s+1} + I_{x-1,y,s-1} - I_{x+1,y,s-1} - I_{x-1,y,s+1})$$

$$D_{ys} = \frac{1}{4}(I_{x,y+1,s+1} + I_{x,y-1,s-1} - I_{x,y+1,s-1} - I_{x,y-1,s+1})$$

The resulting values in the fourth column describe how far along each axis to move the detected feature point. The movement in the X and Y directions are simple additions because the step is constant, but the scale step is not constant. The subpixel scale must be interpolated from the known scale at each difference image.

This paper is interested in tracking point features, so the feature descriptor portion of the SIFT algorithm is ignored. SIFT features are normally tracked by comparing feature descriptors. For more information on finding the orientation and descriptor of the feature, see [Lowe-2004].

Feature Tracking using Nearest Neighbor Correspondence

The correspondence problem is one of assignment. Given two independently produced sets of points, a point from the second image must be found that closely matches a given point in the first image. Assuming the same features will be detected in each image this problem has a solution.

The Hungarian Method is based on the work of two Hungarian mathematicians, D. Kőnig and E. Egerváry, later named and published by Harold Kuhn in 1955. [Kuhn-1955] The original method assigned individuals to jobs based on which jobs they could perform. The result is the minimum cost match between a group of jobs and a group of individuals where each job or individual may only be matched once.

The two groups form a bipartite graph. For purposes of feature tracking, the

vertices of the graph are features found in each of two consecutive frames, forming two disjoint sets. The edges of the graph are assigned a cost associated with matching a feature from the first frame to a feature in the second. The cost function will be discussed later. For now, assume that edges with a low cost value match better.

1. Choose the lowest cost in each row then subtract this value from all other costs in the row.
2. Choose the lowest remaining cost in each column and subtract this value from all the other costs in the column. Any zeros indicate a potential match, though multiple rows may have a zero the same column.
3. Next, choose a lone, unflagged zero. A lone zero is any zero that appears once in a column. A zero may become lone if another zero in the column is flagged.
4. Flag all the edges in the row if the selected zero is the only one – flagged or unflagged – in the column, otherwise flag the edges in the column.
5. Continue to select zeroes until all have been flagged.
6. If some rows do not have selected zeroes, choose the smallest unflagged cost and subtract it from all the other unflagged costs. Then, unflag everything and repeat steps 3-6 until an edge is chosen in each row.

A special case exists when unflagged zeroes exist, but none of the unflagged rows has a lone zero. In this case, a random zero is flagged and the process continues.

The algorithm still works if there are more columns than rows. Some columns will simply not get assigned. However, if more rows exist than columns or if certain rows should not be matched, the algorithm will still attempt to find a match. If no

columns are left to match, the algorithm could loop infinitely. If certain rows should not be matched, the algorithm will attempt to match them anyway. This can lead to unpredictable results.

Imagine that the first frame has a feature that does not appear in the second and the second frame has a feature that does not appear in the first. Now imagine that both frames have the same number of features. The Hungarian Method will match the two extraneous features together unless the features are very far apart. In this case, the Hungarian Method will mitigate the extra cost by mismatching other features until the total cost is at a minimum. This causes the track to jump between features.

Instead a simpler solution is proposed, but ultimately one better suited to the problem at hand. Using the bipartite graph from the Hungarian Method, select the best possible match in each row. If the best possible match is worse than some minimum value, ignore the row as unmatchable, otherwise there is a potential match. Given that match, there are three possible cases:

1. No previous match for this row was found.
    - In this case, assign the match and continue.
2. A previous match exists, but the old cost is lower.
    - Ignore the new match as unmatchable.
3. A previous match exists, but the new cost is lower.
    - Remove the previous match and assign the new match.

This method does not handle occlusion, missing features or any of the other problems that plague point-tracking algorithms. It does, however, work if the only information

available is the position of the feature. The only cost function necessary is the distance between points.

Many more intelligent approaches for tracking of point features exist. Galvin, McCane and Novins described a method for corner tracking that used two trackers then compared their results. [Galvin, McCane and Novins-1999] Their correspondence tracking method included position, velocity, color and gradient at the feature position as part of a confidence function. This confidence function matches very closely with the cost function mentioned above.

A previously tracked vector is represented by

$$x = [\, p_x, p_y, v_x, v_y, g_x, g_y, c \,]^T$$

The candidate vector is represented by

$$z = [\, p_x, p_y, g_x, g_y, c \,]^T$$

The cost function for matching x and z is

$$d(x, z) = w_g \frac{|z_g - x_g|}{(|z_g| + |x_g| + 1)\sigma_g} + w_c \frac{|z_c - x_c|}{|x_g|\sigma_c} + w_p \frac{|z_p - x_p|}{\sigma_p}$$

Here, p represents position, v represents velocity, g is the smoothed gradient at position p and c is the color. σ represents the estimated standard deviation of the differences and w represents the weight applied to each the gradient, color and position.

Adding the gradient and color elements to the cost function can improve the tracking results by pushing closer the features that may be spatially distant but still close

in appearance. A threshold for the maximum possible difference in cost is still required to prevent the tracker from jumping between similar features when the first disappears in the first frame and another appears in the next frame. This threshold needs to be more lax than the simpler position tracker because of the added dimensions in the feature space. A good mnemonic for this threshold can be found by subtracting the expected value of the cost function from its standard deviation.

Initially, the gains made from the added work of finding the gradient and color values in the Galvin-McCane-Novins equation for each feature as well as the standard deviation for each appear quite small. The position-only tracker is very similar to Galvin-McCane-Novins. By weighting both the gradient and color to 0 in the Galvin-McCane-Novins equation, we are left with a simple position tracker. Tracks that jump between physical features are also more likely in the Galvin-McCane-Novins tracker because the maximum cost requirement is more relaxed. However, tracks made by fast moving features are more likely to remain unbroken. This can create more, longer tracks.

No matter which tracking method is used, the quality of the track depends on the features being detected. If the detected features loop around the physical feature as it moves, the track is very jagged. This can be mitigated by averaging the position over several tracked frames. The research for this paper averaged the position of the tracked features over five frames when smoothing was applied.

Lucas-Kanade Optical Flow

Optical flow is a method of capturing the motion between two frames of video. This motion can be represented as a transformation between the two frames of animation either as a simple translation or as a more complex affine transformation. The algorithm described here was first developed by Bruce Lucas and Takeo Kanade in 1981 as an image registration technique. [Lucas and Kanade-1981]

In order to use this technique to find local motion, we must restrict our tests to a small window on the image. Within this window, we will make several assumptions that make our calculations easier. First, all the pixels in the window will be moving in the same direction. Second, the base line of the motion is small. Third, the source and destination images have the same illumination. We can, therefore, describe the motion as follows:

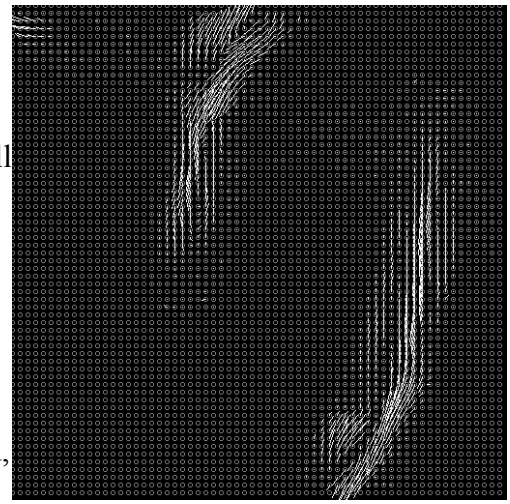

Figure 17: A vector field showing the average optical flow. The circle represents the original of the vector and the line represents the magnitude and direction of the motion.

$$h(\tilde{x}) = \tilde{x} + \Delta x \,,\ \forall\, \tilde{x} \in W(x), \text{ where } \Delta x \in \mathbb{R}^2$$

In terms of image intensities,

$$I_1(x) = I_2(h(x)) = I_2(x + \Delta x)$$

If x is a function of time, this becomes

$$I(x(t),t) = I(x(t) + u\,dt, t + dt), \text{ where } \Delta x = u\,dt$$

and when expanded using Taylor series expansion,

$$\nabla I(x(t),t)^T u + I_t(x(t),t) = 0$$

or more simply,

$$\nabla I^T u + I_t = 0$$

Here, u represents a motion vector in the direction of the image gradient. This is the vector we wish to find within our local window. One observation to make here is that the value of u is strictly dependent upon the image gradient in the region of the image being tested. Motion perpendicular to the image gradient is ignored. The true, global motion may not be detectable if the window does not contain sufficient texture. This is known as the aperture problem.

The least squares estimate of the motion is often formulated as a minimization of the following error function:

$$E_b(u) = \sum_{W(x)} [\nabla I^T(\tilde{x},t)u(x) + I_t(\tilde{x},t)]^2$$

Breaking down the error function for linear least squares,

$$\nabla E_b(u) = 2\sum_{W(x)} \nabla I(\nabla I^T u + I_t) = 2\sum_{W(x)} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} u + \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \right)$$

Least squares requires this function to be minimized, so it is set equal to 0.

$$\nabla E_b = 0$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} u + \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} = 0$$

rewritten,

$$Gu + b = 0$$

If G is invertible, the solution is

$$u = -G^{-1} b$$

G is singular if the window is centered over an edge or over an area of constant intensity. In these locations, G is not invertible and the detectable motion is 0. The best locations for finding motion also happen to be feature points. You may notice that the matrix G can also be found in the derivation for the Harris Corner Detector, shown above as the matrix M.

Motion Segmentation using K-means Clustering

Both point tracking and optical flow only give us information about two consecutive frames. To get a bigger picture view of the motion within the pollen tube, the motion over time must be visualized. A simple average of the optical flow vector fields will either show whether the motion is truly chaotic or whether the motion follows a consistent path.

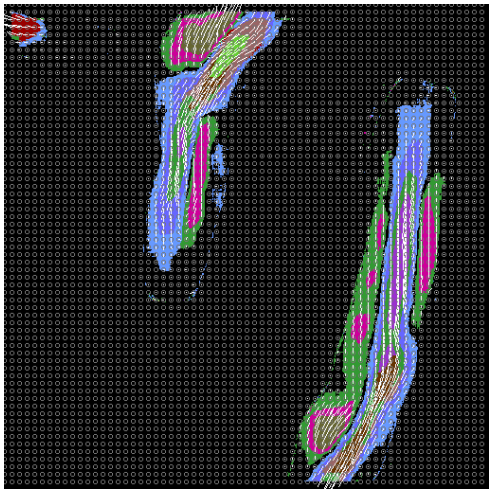

Figure 18: Motion clusters overlaid on top of the average optical flow vector field using random colors to denote cluster regions.

Pollen tubes do exhibit consistent motion. The more frames of animation available, the more consistent the average vector field. By applying a clustering algorithm to the averaged vector field, areas of motion can be grouped by direction and magnitude.

The k-means clustering algorithm is one of a group of hard clustering algorithms. [Theodoridis and Koutroumbas-2006] The algorithm groups data vectors, in this case vectors in a vector field, into *k* groups based on a representative, mean vector. The similarity measure is the squared, Euclidean distance from the test vector to the representative vector.

At initialization, the number of representative vectors must be chosen. This is the value of *k*. These vectors may be assigned randomly. Their initial value need only break up the test vectors. After all the test vectors have been assigned to their closest

representative vector, updating the representative vector can be done by taking the mean of the test vectors in that group.

$$\vec{v}_{representative} = \frac{1}{N} \sum_{i=1}^{N} \vec{v}_{test}$$

This process repeats until some convergence condition is met. This convergence condition may be any or all of the following:

- A specified number of iterations have passed.
- The representative vectors move less that a given threshold of distance.
- The number of members in a cluster does not change.

Chapter 3: Methods for Identification

Microscope Data

Up to now, the pollen tube movies have been viewed as a series of images. Motion has been measured in units of pixels per frame. For real comparisons between pollen tubes, more exact units must be used. Fortunately, the LSM file format retains the settings used on the microscope as a TIFF tag. [Zeiss-LSM Format] The scale in meters per pixel and the time between frames in seconds may be extracted from the data structure stored in this tag. For comparison, a pollen tube can be measured in nanometers and an entire 600-frame movie gives roughly a five minute window on the life of a pollen tube.

Feature Selection

Feature tracking finds the distance and direction moved by a feature between two frames. Given the more exacting information found in the microscope settings the speed, measured in meters per second, may be calculated.

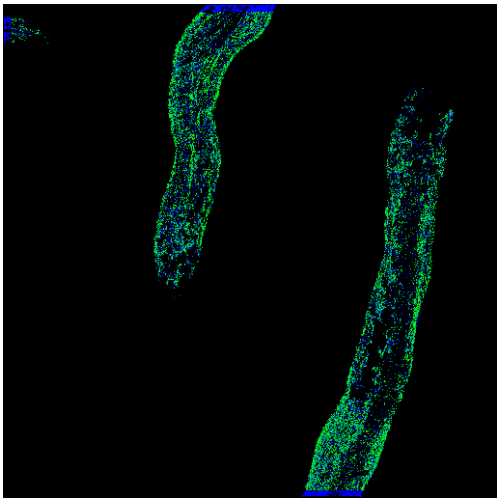

Figure 19: Strange motion map. Green dots represent features tracked in the opposite direction to the optical flow. Blue points represent features tracked perpendicular to the optical flow. The dark areas either have no tracked features or all the features are moving with the average flow.

The position of the detected point on the physical pollen tube feature may change slightly

due to that feature going out of focus or due to noise. In a clear, sharply focused pollen tube movie where the point features are very close to the size of a pixel, these inconsistencies are not much of a problem if the feature is actually moving. For features that hover, the changing position of the feature can create motion where none is actually present. When point features are close to the size of a pixel, this variance is small.

The real problems occur with blurry pollen tube images where the point features are multi-pixel blobs and the detected feature position circles around the blob. Averaging the position of the detected feature over several frames can smooth this motion out. Some image processing techniques may be useful in clarifying the blurry features, but in this paper, enough clear pollen tube movies were available that blurry pollen tube movies could be ignored.

Optical flow does not rely on the clarity of any point features. The vector field obtained gives a strong indication of the expected direction that the point features should be moving and their speed. However, the optical flow is averaged over the entire course of the movie. Point features can and do move in strange directions. It is these differences that will help to identify the types of markers being used.

More care will be taken in chapter six to describe the feature vector used to differentiate between makers. A simple feature vector is defined here in order to limit the size of feature space for this example. The comparison between the vector for the tracked features and the vector for the average optical flow is limited to three values: projection of the tracked feature vector onto the average optical flow vector, projection of the average optical flow vector onto the tracked feature vector and the cosine of the angle

between the two.

$$\vec{f} = \begin{bmatrix} \frac{\vec{x}_i \cdot \vec{\nabla}_i}{|\vec{\nabla}_i|} \\ \frac{\vec{x}_i \cdot \vec{\nabla}_i}{|\vec{x}_i|} \\ \frac{\vec{x}_i \cdot \vec{\nabla}_i}{|\vec{x}_i||\vec{\nabla}_i|} \end{bmatrix}$$

Given this representation, a great deal of overlap should be expected between feature vectors produced by different markers. A single feature vector cannot be representative of the entire group. Instead, the great bulk of feature vectors should show enough difference to give a good indication which marker was used to produce a movie.

The data for a feature vector may come from any source. This paper focuses on the relationship between the average optical flow and tracked feature points. Other pieces of information exist such as: the relative intensity of the physical features to the background, feature density, the size and shape of physical features, or the size and shape of vacuoles. The classifier will operate the same as long as the information can be transformed into a number.

Data Normalization

According to Theodoridis and Koutroumbas, in order to eliminate undue influence on the classification process, features that may be in different dynamic ranges should be normalized. Without knowing in advance which dynamic range a feature will fall under, a good method is to first find some statistics on the population of feature vectors. The mean and variance are simple to calculate.

$$\bar{x} = E[x] = \frac{1}{N}\sum_i x_i$$

$$\hat{\sigma}^2 = E[(x-\mu)^2] = \frac{1}{N-1}\sum_i (x-\bar{x})^2$$

Test statistics will be calculated as follows:

$$\tilde{x} = \frac{x-\bar{x}}{\hat{\sigma}/\sqrt{N}}$$

This calculation will be performed independently for each dimension of the feature vector. The mean and variance will be found for the set of all feature-tracking vectors and optical flow field vectors in the same pollen tube movie. This will prevent the features in one movie from unduly affecting any others.

Bayes Rule

Given a feature vector, which marker is more likely to have produced that vector? Statistically, this may be restated as a probability that the feature vector, *x*, falls within a given class of marker, ω. [Theodoridis and Koutroumbas-2006]

$$P(\omega_i|x)$$

According to Bayes Rule,

$$P(\omega_i|x)=\frac{p(x|\omega_i)P(\omega_i)}{p(x)}, \text{ where } p(x)=\sum_i p(x|\omega_i)P(\omega_i)$$

$P(\omega_i)$ and $p(x|\omega_i)$ can be calculated and $p(x)$ may be ignored because it is the same for all classes.

Assuming a normal distribution of feature vectors,

$$p(x|\omega_i)=\frac{e^{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}}{(2\pi)^{l/2}|\Sigma_i|^{1/2}}$$

where l is the dimension of feature space.
$\Sigma_i=E[(x-\mu_i)(x-\mu_i)^T]$ is an $l\times l$ covariance matrix and
$|\Sigma|$ is its determinant.

$P(\omega_i)=\frac{n_i}{N}$, where n_i is the number of feature vectors in the class.

Now a classifier may be defined as the most probable class for a given feature vector.

$$p(\omega_i|x)>p(\omega_j|x)$$

This Bayesian classifier is a simple, linear classifier that assumes all input classes are distributed normally and that separates two or more classes by a quadratic decision surface. This means the boundary between two classes may be an *l*-dimensional ellipse, parabola, hyperbola or line. This quadratic decision boundary comes from the square of the Mahalanobis distance between the test vector x and the means and covariance defined for each class.

$$(x-\mu_i)^T \Sigma_i^{-9}(x-\mu_i)$$

The boundary is simply the limit where a test vector is closer to one class than to another.

Despite all the assumptions needed to get to this point, the quadratic decision surface makes a good base line discriminator. Other classifiers, such as a Neural Network trained on a large number of test cases, may perform better. However, the goal is to determine whether fluorescent markers may be identified by the motion of the physical features that they highlight. The Bayesian classifier can provide information about how the physical features compare in terms of the mean feature vector and covariance matrix used to make the classification. A neural network is a black box classifier. Knowing the weights used in a neural network gives no extra information about the features being classified.

Chapter 4: Results of Straightening the Pollen Tube

Foreground Mask

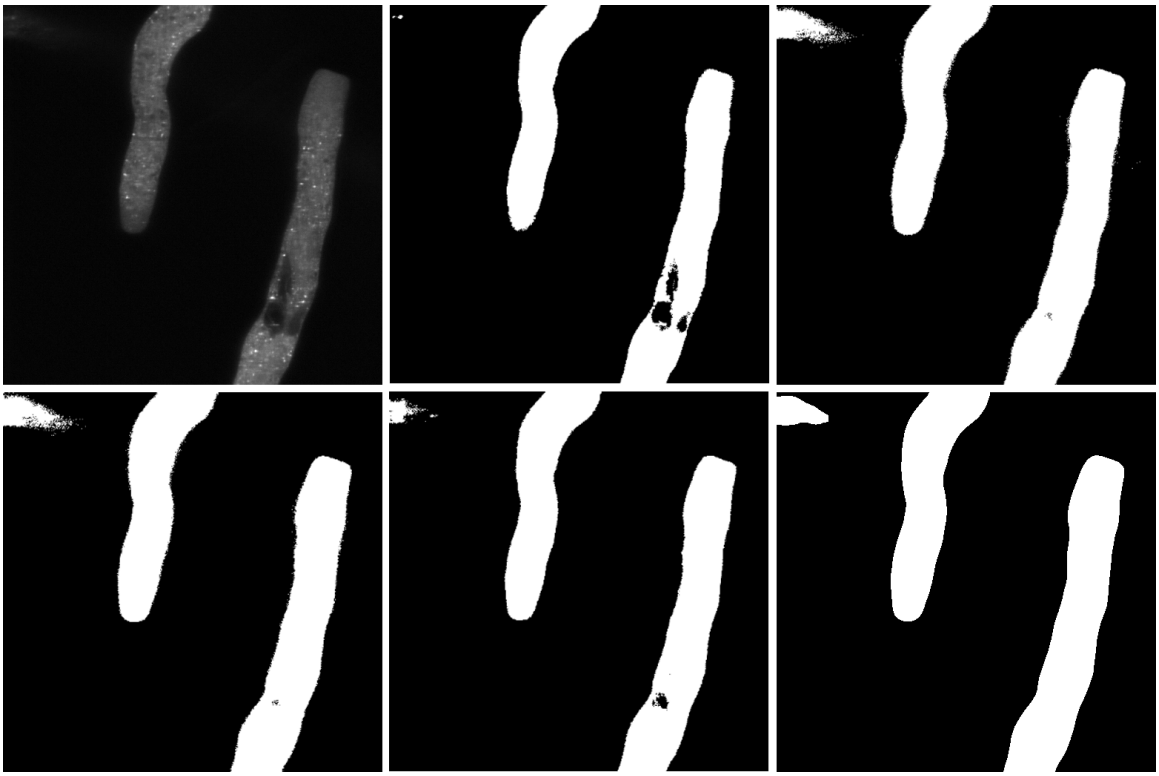

Figure 20: Top row: a) Original Image, b) Otsu Threshold, c) Rosenfeld Threshold. Bottom Row: d) Expected Value Threshold, e) Average of all three thresholds, f) Expected Value Threshold applied to an image with a median Filter with a 5-pixel radius window applied.

Perhaps the most important step in finding a good center line for the pollen tube images is to find a clean foreground mask. The position of the center line is based on the distance between the edges of this mask, so the more smooth the edges, the more smooth the center line and the better the final, straightened pollen tube. The foreground mask is ultimately a threshold. In chapter 1, the expected value was chosen as the threshold value because it works best experimentally. Many other threshold algorithms exist and not all

could be tested. Figure 20 presents a few of the options and their results.

As stated in chapter 1, the value needed to properly threshold the foreground from the background changes as the dyes fade over time. Laser intensity is also an issue and can be manipulated by the microscope operator as the movie is being taken. For these reasons, choosing a perfect value is impossible. A value that works well for one image does not necessarily do well for any other images, even in the same movie. The threshold algorithm must calculate a good value.

With dozens of threshold algorithms available, choosing one to use is an arbitrary selection. A threshold algorithm need only separate the pollen tube from its background without significantly distorting the pollen tube shape. The Otsu algorithm gives good results but vacuoles in the pollen tube show as background. The Rosenfeld algorithm, while less susceptible to vacuoles in the pollen tube, tends to choose a threshold that causes any blurry areas in the image, such as the tip of the pollen tube, to be extra fuzzy. The expected intensity is usually a value somewhere between Otsu and Rosenfeld if the image is clear, but will capture part of the background if the image is blurry. The cleanest looking pollen tube masks can be obtained by averaging the results of these three algorithms together.

Even the best threshold algorithm will not handle noise in the image. The edges of the foreground mask will still be fuzzy. Some method of smoothing the initial image must be implemented before the threshold is applied. The most obvious is to apply a Gaussian filter. However, a Gaussian filter with a large value of σ squashes the foreground, dramatically widening the pollen tube mask. A Gaussian filter with too small

a value of σ will not close the vacuoles and our center line will be broken.

A median filter does a good job of smoothing without widening the mask significantly. The median filter is much slower than the Gaussian filter because each window of pixels must be sorted. In this paper, a median filter with a 5-pixel radius window is first applied to the image. Then a threshold using the expected intensity of the image is applied to obtain the foreground mask. This method smooths the edges of the final mask and closed the vacuole in several cases. The results are good, but the algorithm is slow.

Pollen Tube Tip

Without a tip, a pollen tube has no landmarks that can be used to orient the straightened image. The pollen tube can still be straightened. Chance will determine whether two pollen tubes without tips are straightened in the same direction. Statistically, comparing two different pollen tubes is problematic because they are not in the same population. Whether the position of the tip influences the behavior of the internal structures is debatable. The tip does appear to move independently from the rest of the pollen tube, at least over the course of a two minute movie.

The job of finding the tip, even when one is present in the image, can be difficult. This relates directly to the idea of a well-behaved pollen tube. If the tip touches an edge, the center line will also touch the edge. In some cases, the true end point can be detected.

The center line makes an obvious angle where it jumps toward the edge of the image. In other cases, the center line continues on to the edge as if the pollen tube did not have a tip showing. Instead of trying to detect a sharp change in curvature, both of these movies are simply ignored. In all, one in ten pollen tubes have a visible tip that remains undetectable. The microscope operator could correct the problem more easily than the programmer once informed how the program worked.

The Center Line and Its Effect on Texture Mapping

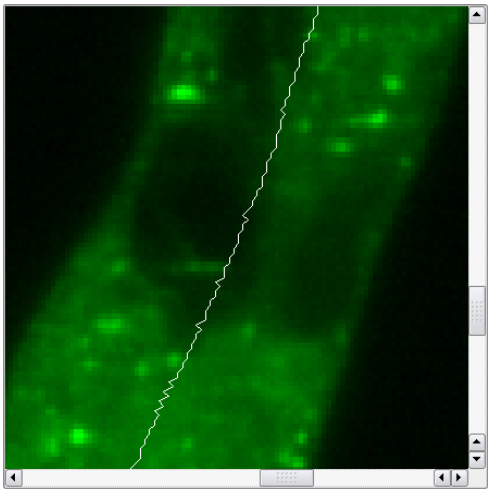

Figure 21: Jagged Center Line

The method used in chapter 1 for finding the center line is at once simple and problematic. A large amount of error is present. Using the magnitude of the gradient of the distance field discretizes the center line to pixel coordinates. This introduces some error in the curve. Thinning the line so it can be walked introduces yet more error. In some cases, the error is as much as two pixels. Another assumption is that the pixels in the thinned center line are evenly spaced and these positions are good enough to define rows in the final image. This error is transferred to the straightened pollen image as errors in the texture mapping process.

The original image is sampled at each calculated position to determine the image intensity at that position. When only one sample is taken per pixel, the straightened pollen tube does not appear smooth. The edges of the pollen tube are jagged and the point features inside the pollen tube appear pixelated. If samples for the same pixel are also being taken from the row above and the row below, the average of these samples smooths out some of the error. The differences between averaging nine samples per pixel and weighting the nine samples with a Gaussian mask are not visibly noticeable between straightened pollen tubes. Weighting with a Gaussian mask requires a division by 16 while averaging requires a division by nine. Weighting with a Gaussian is preferable because the division by 16 can be done with a bit shift, making the algorithm faster.

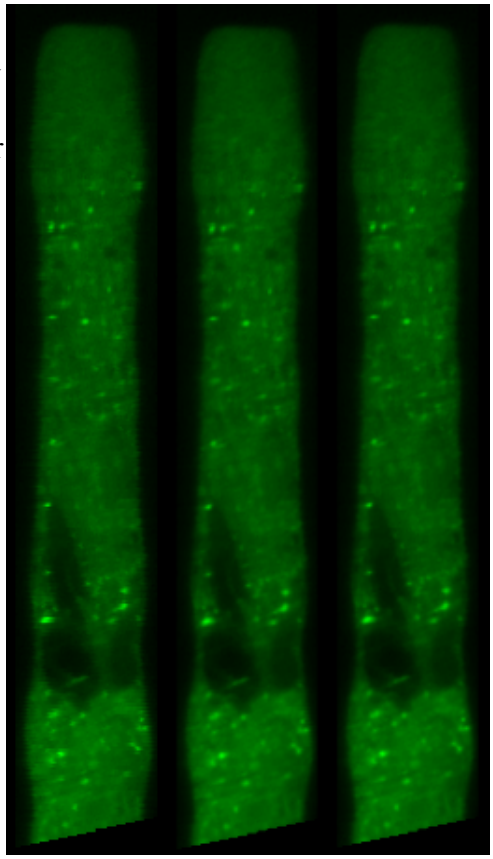

Figure 22: From left to right: 1-sample per pixel, 9-samples per pixel, 9-samples per pixel weighted with a Gaussian mask.

Chapter 5: Results of Feature Detection and Tracking

Feature Tracking

The tracking algorithm is very simple. The only requirement is that the feature in the next frame be the closest to the feature in the current frame, unless there is a better match. For such a simple algorithm with simple rules and expectations, the trade off is memory. The tracking algorithm requires a bipartite graph to compare the features in each frame. The amount of memory required for this is roughly the square of the number of features found in a frame times 8-bytes plus the total number of features found in both frames times 20-bytes, not including the amount of memory required for the frames of animation in the movie. For a movie with approximately 150 features found in each frame, the tracking algorithm takes 181 KB of memory. This poses no problems for a modern computer system.

Tracking fails if many features are detected at the same location. When two or more features overlap, they fight for control of the track and often create parallel tracks. When one of these overlapping features disappears, one of the parallel tracks would be lost. This can create a large number of very short tracks. To correct the problem, features should be selected that are some minimum distance apart. If one version of the feature disappeared then another would remain to take its place, maintaining the track.

Finding the best minimum distance threshold is a circular problem. The more lenient the threshold, the more likely that the track will jump to a different feature. To

prevent this, the minimum distance is shortened, limiting the distance a feature may move. Such a tracking algorithm is only useful if tracking slowly moving features. If the physical features move too quickly, the algorithm loses its track. In order to correctly track these fast features, the tracking algorithm must allow features to cover a distance of many pixels in the space of a single frame, increasing the minimum imposed distance and making the algorithm too lenient. We can either have an algorithm that is good at tracking slow features and not good at tracking fast ones or we can have an inconsistent algorithm that will sometimes track fast features, but tends to allow a track to jump between features.

Despite its shortcomings, this algorithm is capable of following a detected feature through a great percentage of the lifespan of that feature. Point features in a pollen tube can move more than five pixels between frames if the feature is moving quickly. More typically, the physical features do not move much at all and are easily tracked. Any strangeness in the track is a function of the detected features not being exactly centered on the physical feature they are tracking. Most of the feature detection algorithms described here discretize the detected features to the pixel grid when the physical feature may not have moved that far. Consequently, the track may jump around randomly as the detected feature circles the physical feature. If a physical feature is not detectable, a track may be lost. This is problematic in images with low contrast between the physical features and the background.

Feature Detection Parameters

Each algorithm has its own set of parameters that determines how many features will be detected in each frame. Tweaking the feature detection parameters correctly results in the tracking algorithm working well or not working at all. Parameters for different algorithms are tweaked so that roughly the same number of features are found on the first frame of the movie by each algorithm. Starting with the same number of features on the first frame may not be a valid comparison for track lengths because the same features are not being detected with each algorithm. Instead, starting with the same number of features does give us a way to visually compare how the algorithms do their work.

The parameter common to all the feature detection algorithms is a threshold that limits the number of features that get detected. This threshold is often based on the amount of contrast between the center pixel of the feature and the eight pixels surrounding it. This contrast could be based on image intensity or some calculated value such as intensity variance over a small area of pixels around the center. SUSAN, on the other hand, uses this threshold to build a look up table needed to calculate the USAN value.

Tracks may become lost or broken due to issues with the contrast threshold or physical features changing shape and focus. The contrast threshold parameter can be the source of many lost tracks. This parameter is tweaked by the user. The parameter is then used for every image in the movie. As the fluorescence fades and the image contrast lowers, the number of features removed by this threshold increases. As features move in

and out of focus their intensity changes as well, potentially getting removed by this threshold. Even if the feature fails to be detected for one frame, the track is lost.

The next parameter common to all the feature detection algorithms is the minimum Euclidean pixel distance between features, a function that originated with the Harris Corner Detector. Because the Harris detector gives a positive response to the entire area around physical features, some mechanism was needed to reduce the number of features detected. The minimum distance function works by first selecting the best feature in terms of its cornerness value and then each new feature is tested against the selected features to see if it is too close. This method can also be applied to other feature detection algorithms to decrease the amount of overlap in the detected features.

Feature Detection Algorithms

The mean and variance of the track length is used to determine consistency in the feature detection algorithms. This paper had an available pool of 100 movies. Of these, 34 were used for testing because they were all high resolution, showed more physical features, and were well-behaved as described in the previous chapter.

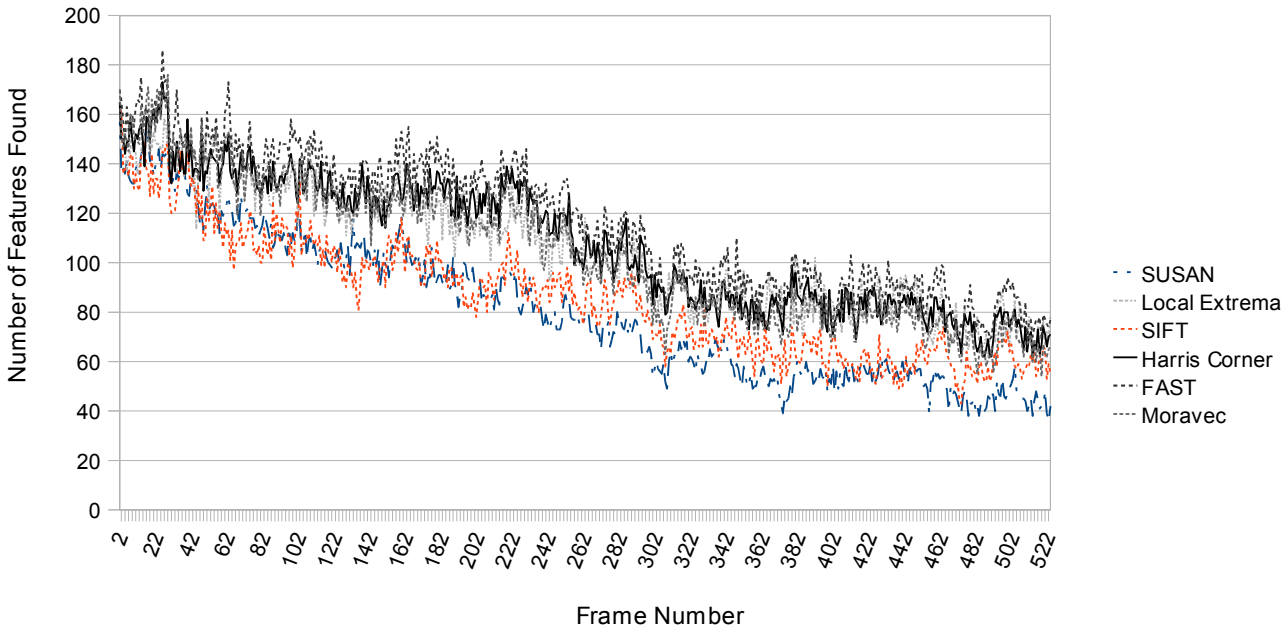

Table 1: The number of features detected over time using seven different feature detection algorithms.

All the feature detection algorithms discussed here are susceptible to intensity changes. The fluorescent proteins fade as they age so the number of features detected trends smaller with each frame. We can easily see this effect if we start with each feature detection algorithm finding roughly the same number of feature in the first frame. As the intensity of the fluorescence fades, the number of features detected goes down. However, the number of features detected does not correspond to the correctness of tracking. While SUSAN tends to detect the fewest features, it is one of the best algorithms for tracking because of its consistency.

Syp21movie.lsm	Extrema	Moravec	Harris	SUSAN	FAST	SIFT
Number of Tracks	6543	5549	4784	3619	4344	5530
Expected Correspondence	7.31	8.51	10.76	11.76	12.15	8.56
Variance of Correspondence	2396.32	1848.82	1383.34	973.23	1169.8	1915.57
Standard Deviation	48.95	43	37.19	31.2	34.2	43.77
Features in First Frame	204	208	213	213	202	225
Time to Find Features	25.16	96.09	45.59	44.15	26.73	422.07

Table 2: Correspondence data for six algorithms tested against Syp21movie.lsm. Their parameters are tweaked to each start with roughly the same numbers of features in the first frame. The time to find features is measured in seconds.

If an algorithm is judged based on how many times it had the longest expected track, longest actual track or smallest standard deviation then the three least interesting algorithms in terms of tracking are SIFT, Moravec's Interest Operator and the Harris Corner Detector. These three algorithms form the middle ground being neither the best nor the worst in any category. The three most interesting algorithms are Local Extrema, SUSAN and FAST. Local Extrema is interesting because it is consistently bad for tracking. SUSAN and FAST are interesting because they are consistently good.

No one algorithm can be described as the best. SUSAN and FAST were the only algorithms that had both the longest actual track and the longest expected track in the same movie. More often, the algorithm with the longest expected track did not have the longest actual track. The algorithm with the longest expected track was more likely to have the smallest variance in track length. The same is true for the algorithms with the shortest track lengths and largest variances.

Occurrences in 34-movies	Extrema	Moravec	Harris	SUSAN	FAST	SIFT
Longest Expected Length	0	0	8	11	15	0
Smallest Standard Deviation	0	0	7	16	8	3
Longest Actual Track Length	0	0	2	12	18	2
Shortest Expected Length	27	1	0	0	0	6
Largest Standard Deviation	31	2	1	0	0	0
Shortest Actual Track Length	21	12	0	0	0	1

Table 3: Number of occurrences in various categories for each of six feature detection algorithms in 34 pollen tube movies.

The three least consistent algorithms for tracking are, in order of worst to best, Local Extrema, Moravec's Interest Operator and SIFT. Local Extrema proved to be the worst algorithm for tracking, as it was expected to be. The Local Extrema algorithm represents the naïve approach to feature detection. Given the same parameters in each run, Local Extrema tended to find the most features and the fewest tracks compared to the other algorithms. Tracks tended to be very short with an expected length of only 1.25% of the total number of frames in a movie. Even so, the longest tracks found with Local Extrema would often be in the range of 60-100 frames or 8-16% of the total frames in the movie.

The SIFT algorithm, more specifically the feature detector portion of the SIFT algorithm, was included as a control. SIFT is not intended to detect point features. Instead, it is considered a blob detector capable of describing whole areas of an image as a single feature. The real power of SIFT is in its feature descriptor which can be used to match features directly, something none of the other feature detectors described here can do. When used as a point feature detector, the centroid of the SIFT feature is used. This may or may not correspond to a physical point feature in the image. SIFT tended to detect the fewest features but tended toward the middle ground in terms of track length.

SIFT is also an order of magnitude slower than the other algorithms because it does so much extra work.

Moravec's Interest Operator tended to detect only the corners of point features, leaving room for a second feature to be detected on the same physical feature, even when a minimum distance between detected features was enforced. The tracking algorithm does not work well if two or more features are detected on the same physical feature. In contrast, the Harris Corner Detector gives a positive response to a small area surrounding the physical features. From this, the local maximum may be selected which more effectively centers the detected feature on the physical feature. While not perfect, this does make the Harris Corner Detector a much better algorithm for tracking than Moravec's Interest Operator.

The three most consistent algorithms for tracking are the Harris Corner Detector, SUSAN and FAST. While the Harris Corner Detector is possibly the most common feature detection algorithm in use, it tended to be slightly slower than the other two in the amount of time needed to detect and track features and less consistent in expected track length and variance of track length. Still, Harris provides a good baseline for both expected length and variance in track length, having the longest expected track length just over half as often as FAST and having the smallest variance just under half as often as SUSAN.

SUSAN proved to have the lowest variance in track length, producing the fewest number of short tracks. SUSAN does tend to detect large numbers of features along the edges of the pollen tube, which can be confusing. It is conceivable that these random

edge features would produce many short, extraneous tracks that have nothing to do with the motion of the physical features in a movie. While SUSAN does occasionally produce tracks near the edge of the pollen tubes, these tracks do appear to be associated with physical features. The edge features do appear to be random and as such do not track.

Initially, FAST rated as one of the worse feature detection algorithms for tracking because it detected many features at the same location. The tracking algorithm would get confused, creating multiple tracks where only one should be. When one of the overlapping features disappeared, one of the tracks would be lost. This created a large number of very short tracks. When a minimum distance function was applied to the detected features, FAST became one of the best tracking algorithms in terms of track length with both high expected and high actual track length over the 34 movies.

Effects of Changing Feature Detection Parameters on Feature Tracking

Changing feature detection parameters has the effect of changing the number of features detected. The smaller the threshold, the more features are detected and the fewer low contrast features get ignored. Therefore,

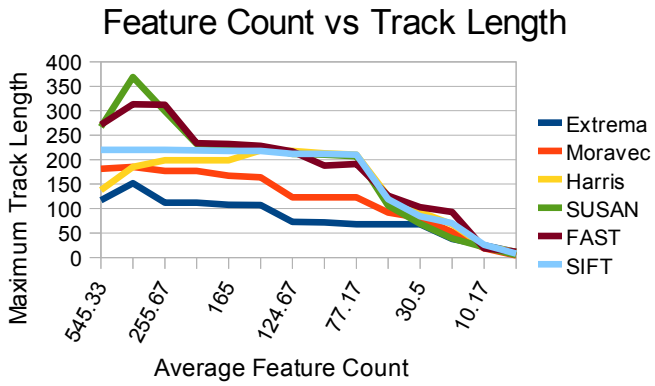

Table 4: Maximum Track Length versus detected features on Syp21movie.lsm.

reason suggests that the more features detected, the longer the tracks and the better the tracking. However, if too many features are present, the tracking algorithm becomes confused and a condition of many small and broken tracks results.

In order to test this, the minimum distance between features in the same image for each algorithm was set to 5, which is the maximum distance allowable between tracked features. The contrast threshold was changed for each tracking algorithm so that a similar number of features were detected in the first frame of the image by each algorithm. The algorithms are then run against the same movie. The idea is place each algorithm on the same footing and to see which algorithm detects the same physical features consistently. Unfortunately, different feature detection algorithms do not detect the same features. Using track lengths does not give a direct comparison. Only a rough idea relationship between algorithms is possible using this method. According to table 5, detecting between 100 and 200 features in the first frame tends to produce a reasonably good expected track length for all feature detectors.

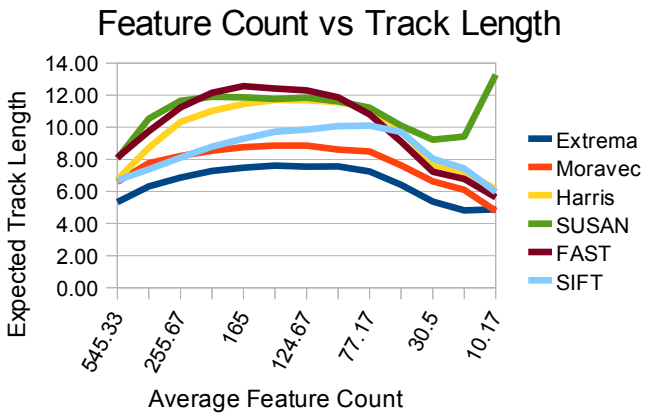

Table 5: Expected Track Length versus detected features on Syp21movie.lsm.

Overall, FAST and SUSAN tend to perform the best in terms of track length, performing better than Harris for large numbers of features. FAST uses difference in

image intensity as its contrast threshold. This is an integer value between 0 and 255. Both SUSAN and Harris use a flowing point value as the contrast threshold. This means that SUSAN and Harris have finer-grained control over the number of features detected.

Effects of Image Processing on the Pollen Tube before Tracking

Image processing is a broad term describing many methods of changing the colors in an image through transforms or filters. The terms "transform" and "filter" are often used interchangeably. There are three types:

1. Linear.
2. Kernel or Window-based.
3. Histogram-based.

Linear filters map one color intensity to another through some mathematical operation. If the range of possible intensities in an image is 0 to 255, inverting the image can be done with the operation $255 - I_{x,y}$, where I is the intensity of the image. The television controls, brightness and contrast, are also linear filters. Brightness is an additive transform – $I + c$, where c is some some constant – shifting the pixel intensity uniformly. Brightness does not change the relative contrast between pixels and, therefore, does not change the number of features detected. Contrast is a multiplicative transform, $(I - 127) * c + 127$, where c is some constant. Gamma correction function is an exponential

transform, I^c, where c is some constant. Both contrast and gamma correction change the contrast between pixels, highlighting or dulling the physical features and changing the number of features detected.

Kernel or Window-based filters calculate a new value for a pixel based on its surroundings. A window is a rectangular group of pixel intensities from the original image. A kernel is usually a matrix where the value at each position in the matrix is multiplied with a pixel intensity at the corresponding position in the window. The resulting values are then added together to find a new pixel intensity. Gaussian smoothing and median filter are examples of a kernel-based transform. By treating an area of the image as a group, noise is reduced. Noise reduction eliminates weak features, reducing the number of features detectable. An unsharp mask sharpens the image, reducing blur while increasing noise.

Histogram-based or statistical transforms take the entire image into account. These transforms change the relative intensities of pixels based on statistical information from the entire image.

Image processing is as much an art form as a technical tool. Each image is different and requires different techniques to make it useful for tracking. More information about these techniques may be found elsewhere. Programs like Adobe Photoshop or GIMP contain dozens of image processing tools.

A straightened pollen tube has a different arrangement of pixel intensities than the original image. This affects how the feature detection algorithms find features. The result is an increase or decrease in the number of features found in the straightened image as compared with the original image.

A straightened pollen tube is a different image and should be treated as such when comparing detected features. However, the same physical features are present in both images. Given the same parameters for the feature detection algorithms, the straightened and original images should detect a very similar set of features if one is transformed slightly. Figure 23 shows the effect that smoothing the original slightly can have on the detected features as compared to the straightened pollen tube. Straightening is not the same as a smoothing filter.

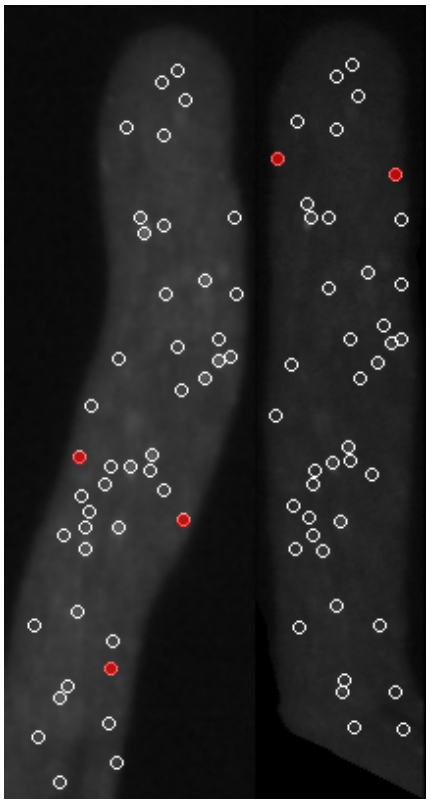

Figure 23: Original pollen tube smoothed with $\sigma = 1.01$, compared with the straightened version. Detected with FAST with a contrast threshold of 7. Mismatched features are highlighted in red.

Chapter 6: General Versus Specific Motion

I would like to note that any statements of biological significance made here have not been vetted by a biologist. They are based on observations made using image processing techniques and computer generated images. They are also made using two dimensional slices of three dimensional objects that may skew the results. Also note that the images shown were made using random colors. While the same color does represent the same type of motion, whether the color is red or blue does not mean anything.

Average Optical Flow Patterns

The average optical flow represents a different type of motion than individual feature tracking. It represents global motion over the lifetime of the movie. Like the lanes of a highway, it shows the direction that the cytoplasm within the pollen tube is flowing. The highlighted organelles do not always move in the same direction as the average flow. Comparing the two types of motion can show that pollen tubes exhibiting a figure eight or spreading motion do, in fact, display organelles

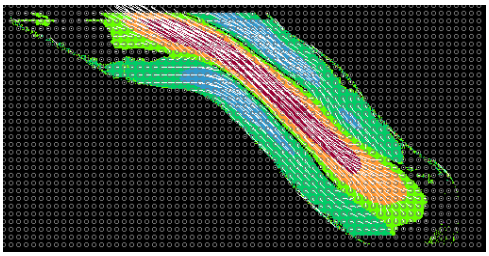

Figure 24: Average optical flow of Ara7-GFP pollen tube capturing reverse fountain flow.

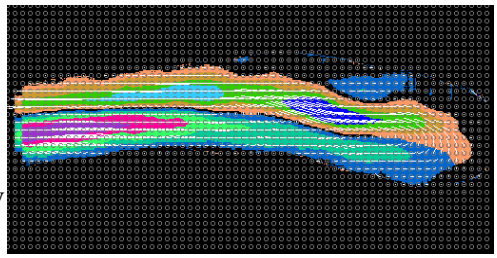

Figure 25: Average optical flow of an Ara7-GFP pollen tube exhibiting loop motion that may be closer to reverse fountain flow if viewed from a different angle.

traveling along overlapping paths.

Inside a pollen tube the organelles move toward the tip along the walls of the pollen tube then flow back down the center of the pollen tube in a reverse fountain effect. Figure 24 shows one case where the average motion in the pollen tube exhibits this reverse fountain flow. Figures 25, 26 and 27 do not.

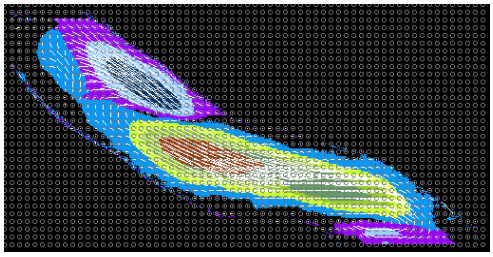

Figure 26: Average optical flow of an Ara7-GFP pollen tube exhibiting figure eight motion. The purple region continues behind the green region.

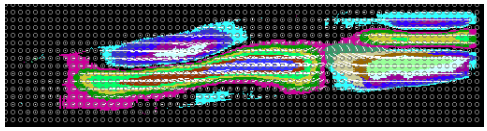

Figure 27: Average optical flow of a Syp21-GFP pollen tube exhibiting spreading motion as well a reverse fountain flow.

In some cases the flow appears to be a loop that travels up one side of the pollen tube and down the other. In other cases, the flow appears to loop behind itself in a figure eight or candy cane pattern.

Figure 27 depicts an interesting phenomenon. The motion in the pollen tube appears to be spreading from a central location. At times these looped structures appear to be viewed from strange angles, so the motion inside the pollen tube appears to be spreading from a central point rather than moving in a loop. This spreading may be a figure eight pattern viewed a little off center.

These patterns describe the general motion found across the 34 movies of pollen tubes. When comparing pollen tubes the flow pattern may hold some further, unknown significance. Therefore, it seems unwise to lump the different motion types together. If the flow forms a loop, comparing it to a pollen tube with a spreading motion may not make sense.

Strange Motion

Strange motion describes the motion of tracked features that do not follow the average optical flow. An average optical flow vector exists at every pixel position in the image. The motion vector of a tracked feature has a starting point and an ending point. For comparison purposes, the average optical flow vector may be associated with either of those points. For simplicity, the two vectors must start from the same position.

The speed and direction of each of these vectors is in units of pixels per frame. The actual speed can be determined by converting the units to meters per second, where the difference between frames of the movie is measured in frames per second and the difference between pixel positions is measured in pixels per meter. The LSM file format contains a voxel size that converts from pixel size to meters and a time interval that converts from frames to seconds.

Strange motion can be visualized by color coding occurrences within the image. The result is a new image where the colored areas show the locations of strange motion in the pollen tube. In areas between clusters of motion, the physical features are more likely to move perpendicular or even backward to the average flow. The tip of the pollen tube is also a region of strange motion. Where the flow of cytoplasm is interrupted by a vacuole, tracked features become stalled and most strangely. Areas where the optical flow is the strongest have the least strange motion.

In order to perform calculations on this strange motion, position and direction must be eliminated from the equation. Both position and direction relate to the

appearance of the motion and not to the effects of the motion. Position can be eliminated by stating that both the optical flow and tracked feature vectors start from the same position. Direction can be eliminated by comparing the angle between the two vectors. The length of each vector is its speed. The dot product or inner product gives the cosine of the angle between the two vectors. Each vector may be projected onto the other. And finally these values may be compared over a period of time by including data from several frames in our feature vector. The result is a 12 to 15 dimensional feature vector composed of repeating the following equation over 4 to 5 frames.

$$\vec{f} = \begin{bmatrix} \dfrac{\vec{x}_i \cdot \vec{\nabla}_i}{|\vec{\nabla}_i|} \\ \dfrac{\vec{x}_i \cdot \vec{\nabla}_i}{|\vec{x}_i|} \\ \dfrac{\vec{x}_i \cdot \vec{\nabla}_i}{|\vec{x}_i||\vec{\nabla}_i|} \end{bmatrix}$$

where $\vec{x}_i$ is the tracked feature in frame i and
$\vec{\nabla}_i$ is the corresponding average optical flow in frame i.

Using Motion Data to Identify Marker Types

The research began using two known and identified markers, Syp21-GFP and Ara7-GFP. Only one type was used in each movie. However, these two classes of marked organelles appear visually alike. The goal of this research is to determine the difference between the organelles highlighted by these markers based entirely on how they move. A Bayesian classifier can be used to produce statistics about the motion of each marker type.

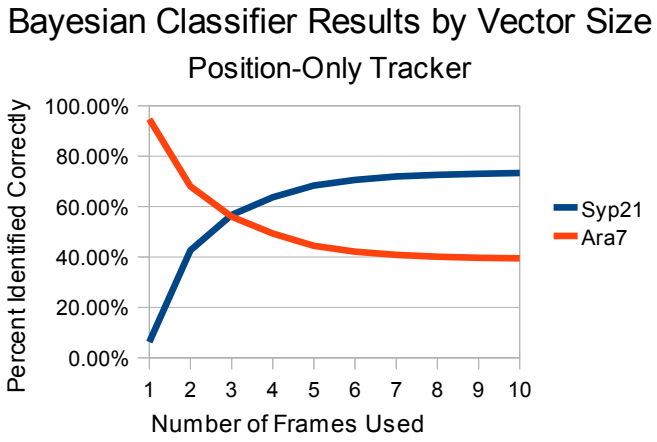

Table 6: The position-only tracker does produce feature vectors that can be classified better than chance if three frames are used. We can do better.

Toward this goal, features were tracked using the Harris Corner Detector with roughly 120 features detected in the first frame of each pollen tube. Both the Euclidean distance between the detected feature points and the Galvin-McCane-Novins equation were used as gain functions. Only tracks longer than 10 frames were used in each case. The feature vectors include two speed values and an

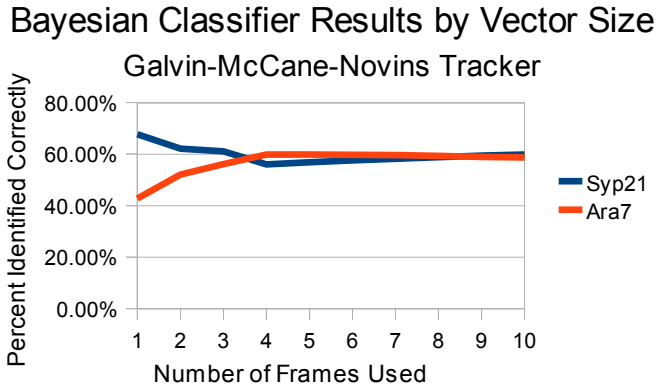

Table 7: The Galvin-McCane-Novins tracker performs more consistently than the position-only tracker.

angle per frame for a total of 15 dimensions for the feature vector if five frames are used.

A Bayesian classifier trained on feature vectors produced in 34 movies identified just under 60% of those feature vectors correctly in a resubstitution experiment. When the position-only gain function was used, the classifier was able to identify both Syp21 and Ara7 markers at a rate better than chance only when data from three consecutive frames was used. In all other cases, the classifier skewed the results toward one marker or the other. When the Galvin-McCane-Novins gain function was used, the classifier returned a better than chance result for both Syp21 and Ara7 markers if two or more frames of animation were used. The classification rate became for both markers the closer to 60% as more frames of animation used. The Galvin-McCane-Novins tracker also produced almost twice as many feature vectors. More tracks longer than ten frames were found and more information was made available to the classifier for training and testing purposes.

The following means were taken from a classifier using a Galvin-McCane-Novins gain function.

$$\text{Syp21-GFP}\ \mu_{Syp21} = \begin{Bmatrix} 57.97 \\ 72.52 \\ 43.18 \end{Bmatrix}, \sigma_{Syp21} = \begin{Bmatrix} 537.2 \\ 479.5 \\ 559.5 \end{Bmatrix}$$

$$\text{Ara7-GFP}\ \mu_{Ara7} = \begin{Bmatrix} -45.44 \\ -56.85 \\ -33.85 \end{Bmatrix}, \sigma_{Ara7} = \begin{Bmatrix} 573.3 \\ 609.9 \\ 558.2 \end{Bmatrix}$$

These represent the first three dimensions of the feature vector. The extra dimensions repeat the first three within about 2% of the value of the first frame. The standard deviation shown is taken from the diagonal of the covariance matrix used by the

classifier. The standard deviation shows significant overlap between the Syp21-GFP and Ara7-GFP data, which accounts for the classifier only correctly identifying 60% of the feature vectors. Dropping some of the data from the training set to be used as a testing set did not significantly change the results.

The first three dimensions of the feature vector are:

1. The projection of the optical flow vector onto the tracked feature vector,
2. The projection of the tracked feature vector onto the optical flow vector,
3. The cosine of the angle between them.

Since the classifier used normalized feature vectors, only relative differences between the two markers can be seen directly. The original values for the mean and standard deviation may be recovered by reversing the normalization.

$$x = \tilde{x}(\hat{\sigma}/\sqrt{N}) + \bar{x}$$

$$\text{Syp21-GFP } \mu_{Syp21} = \begin{Bmatrix} -126.0 \\ -56.04 \\ -0.31 \end{Bmatrix}, \sigma_{Syp21} = \begin{Bmatrix} 278.98 \\ 107.77 \\ 0.6 \end{Bmatrix}$$

$$\text{Ara7-GFP } \mu_{Ara7} = \begin{Bmatrix} -213.38 \\ -108.11 \\ -0.45 \end{Bmatrix}, \sigma_{Ara7} = \begin{Bmatrix} 309.48 \\ 106.25 \\ 0.6 \end{Bmatrix}$$

Based on the recovered means, Ara7-GFP tended to display faster motion closer to the direction of the optical flow. Ara7-GFP marked pollen tubes also exhibited a greater range in the optical flow, meaning the Ara7-GFP pollen tubes were more likely to exhibit both slow and fast motion than the Syp21-GFP pollen tubes. The requirement in the

classifier for multiple frames shows that the relative motion of the tracked features over time matters when comparing Syp21-GFP and Ara7-GFP.

Unfortunately, these statements cannot be applied to all Syp21-GFP and Ara7-GFP pollen tubes. The classifier was based on a population of 34 pollen tube movies. Not all the pollen tube movies contributed significantly to the classifier. Movies with fewer tracks contributed less than movies with more tracks. More Ara7-GFP movies were available than Syp21-GFP movies. Of the 34 total movies, 5 were particularly sparse in terms of number of tracks.

The classifier correctly identified the marker used in 70% of the movies. Of the movies marked by Ara7-GFP, 13 of 18 movies were correctly identified while 10 of 15 movies marked by Syp21-GFP were correctly identified. Of those, all 5 sparse pollen tubes identified as Syp21-GFP. Only 2 of those movies were actually marked with Syp21-GFP. The movies marked with Syp21-GFP that exhibited more fast motion were identified as Ara7-GFP. Therefore a different distribution of movies would likely have produced different results. More work needs to be done in this area to get a better idea of the motion of these two markers.

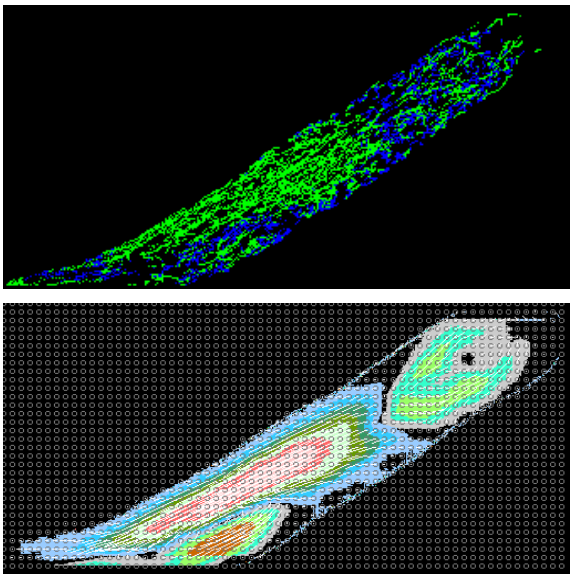

Figure 28: Top) Ara7-GFP classified motion. Green is classified as Ara7-GFP. Bottom) Motion clusters for the same pollen tube.

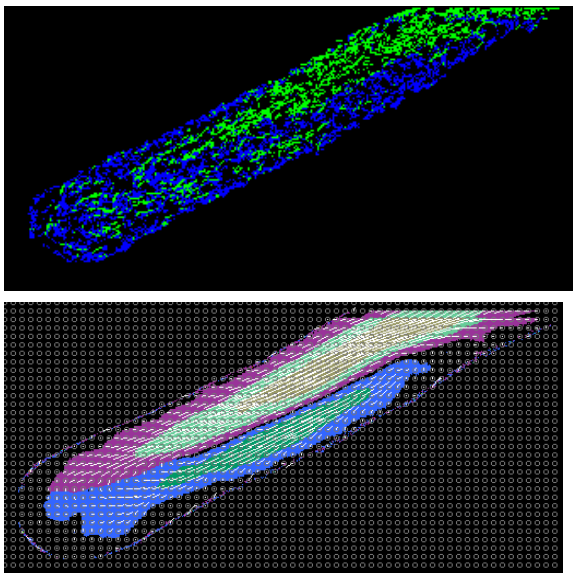

Figure 29: Top) Syp21-GFP classified Motion. Blue is classified as Syp21-GFP. Bottom) Motion clusters for this same pollen tube.

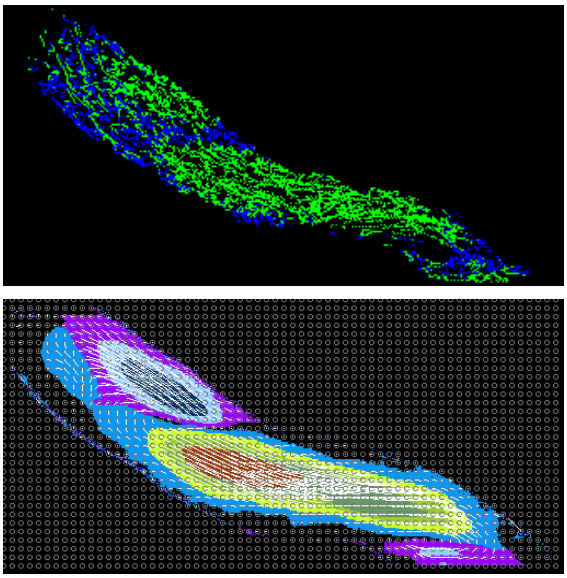

Figure 30: Top) Ara7-GFP classified motion. Bottom) Motion clusters for the same pollen tube.

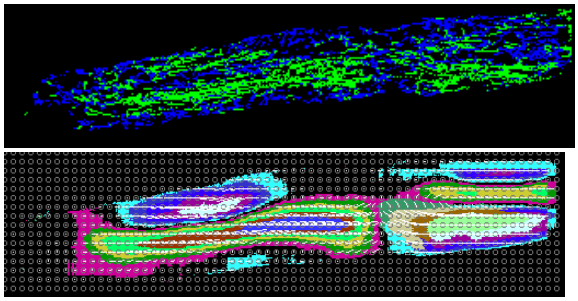

Figure 31: Top) Syp21-GFP classified motion. Bottom) Motion clusters for the same pollen tube.

Chapter 7: Conclusions and Future Work

This project has touched on several research topics in computer programming: image processing, texture mapping, level set, center line extraction, feature detection, feature tracking and Bayesian classification. Various methods of extracting information from confocal microscopy images through automatic processes have been described. Abstraction methods that would allow the pollen tubes to be compared visually have also been provided.

The center line detection can be further improved with better analysis of the shape of the pollen tube. Currently, edges that are cut off by the edge of the image cause incorrect bends in the center line. Also, curve fitting should be used to solve for the curve marked by the list of points already being found. A smooth curve would make texture mapping simpler.

A correspondence tracker was used to track feature points. Many other types of tracking exist that are more robust to problems like occlusion, preventing the track from jumping between physical features and detecting features in near proximity, and also features that pass very close to each other.

The research done for this paper was restricted to two dimensions. All of the algorithms in this paper can be extended to three dimensions. Pollen tubes are three dimensional objects and should be handled as such. Many of the problems currently experienced with features fading out of focus could be solved by tracking the feature through different layers of the pollen tube.

Appendix A – LSM Viewer

One of the goals for this project was to produce a graphic user interface (GUI) that would allow visual comparison of two or more pollen tubes with animation. This GUI was designed to be flexible, giving the user as much control as possible when displaying pollen tube movies. The user interface covers all the topics mentioned in this paper except for classification. The design of the feature vector was considered too complex to be included.

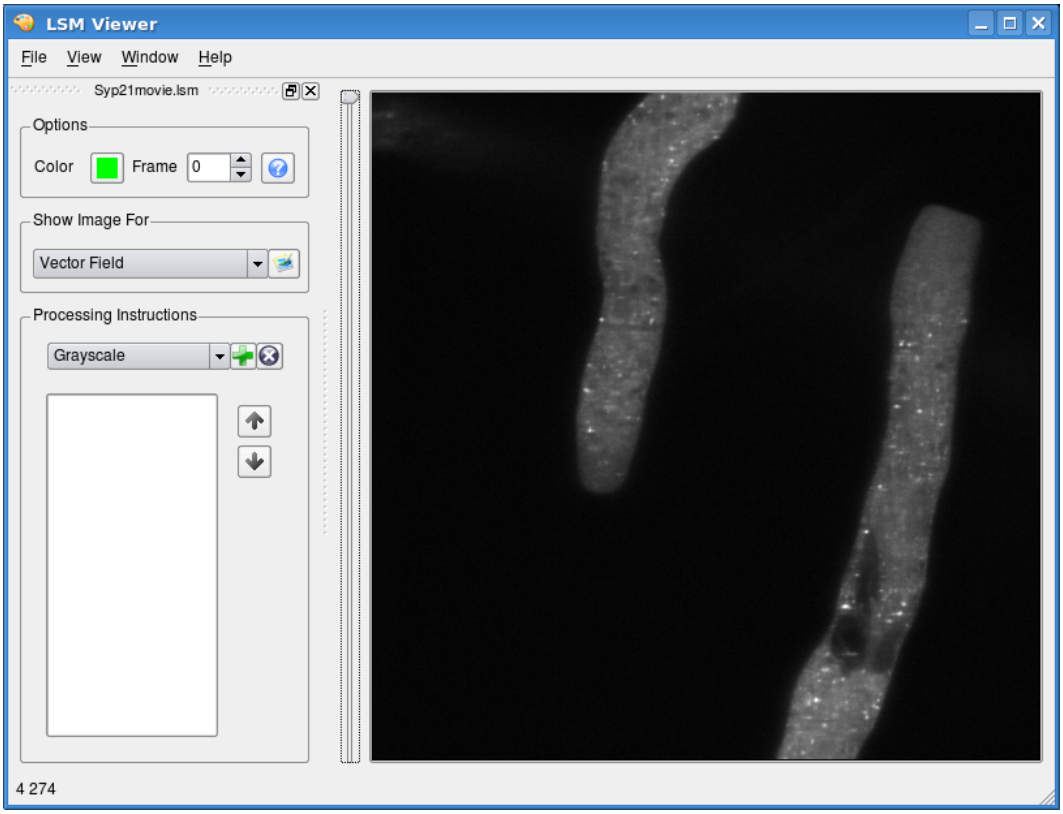

Figure 32: The main window with a gray scale LSM file open and the file's option panel displayed.

A Brief Description of the Code

All the code for this project was written in C++ using gcc as a compiler on a Linux operating system. The user interface was designed in Qt 4.4 and wraps the software that performs all the image processing, feature detection and tracking. Several external libraries were used:

- FAST (http://svr-www.eng.cam.ac.uk/~er258/work/fast.html)
  - The FAST9_decode function was copied from the original source.
  - This code was modified to handle floating point gray scale images.
- FreeImage (http://freeimage.sourceforge.net/)
  - Reads and writes many image formats. Used in some cases to save.
- Gnu Regex (http://www.delorie.com/gnu/docs/regex/regex_toc.html)
  - Used to parse URI strings.
- libtiff (http://remotesensing.org/libtiff/)
  - Used to read real TIFF files. Libtiff does not read LSM files correctly.
- Sqlite3 (http://www.sqlite.org/)
  - Used by the GUI to store settings such as the last opened directory.
- SUSAN (http://users.fmrib.ox.ac.uk/~steve/susan/susan/susan.html)
  - The original source was copied except for the main program.

The above websites were valid as of February 2010.

The user interface is, unfortunately, single threaded. This means that once an operation begins it must be completed before the program can continue. This leads to a very poor user experience. Creating a multi-threaded feature detector and tracker would improve the response time dramatically, but it would also require modifying large sections of code that were originally meant to run in batch mode. The original research for this project did not include an end user and gigabytes of pollen tube data would be processed over the course of many hours. While the user interface is capable of doing some operations in real time, feature detection, feature tracking and finding the average optical flow take time.

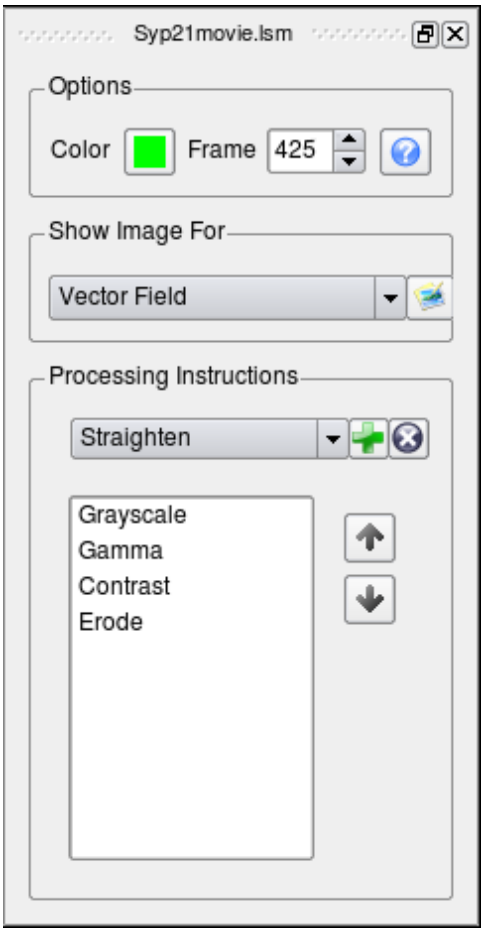

Figure 33: Options dialog.

Opening an LSM file

The GUI only allows images with the extension ".LSM" to be opened. The path to the last file opened will be saved so the file open dialog will start in that directory the next time a new file is requested. Only uncompressed LSM files are readable

because libtiff could not be used due to the strange numbers of color channels in the LSM format. Technically, libtiff can be used to read a gray scale LSM file, but much of the data provided in this project used two color channels. Libtiff breaks when trying to open a two color channel image. Other Zeiss microscopes may have more than three lasers. Opening LSM files produced with these microscopes is not possible.

Once an LSM file has been opened, the user will be presented with an options dialog docked to the left side of the main window. This options dialog gives the user control over the opened pollen tube movie. If more than one LSM file is opened, more than one options dialog will be present on the screen. These dialogs may be stacked, docked to opposite ends of the main window, allowed to float or simply closed. A closed options dialog may be reopened by selecting the name of the pollen tube from the Window menu.

There are three sections to the options dialog: Options, Show Image For and Processing instructions. Options allows the user to select a color for the pollen tube. This color will be used when two or more pollen tubes are viewed together and Alpha Blending is selected from the View menu. The color will also be used to display features and tracks if Show Image is unselected from the View menu. Frame shows the current frame of animation being displayed for this pollen tube movie. The pollen tube may be animated by changing the frame number using either the mouse scroll wheel or clicking the up and down arrows.

The up and down arrows on the keyboard also change the frame number when the control is selected. The last button the options group is the LSM Information button which will be described below.

The Show Image For group is a list of special processing instructions that produce single images from the entire pollen tube movie. These are:

- Vector Field
- Motion Clusters
- Strange Motion Map
- Track Density Map
- Hough Circle Transform
- Distance Field
- Distance Field Gradient
- Straighten Effect

The button in this group activates the selected operation, opening an options dialog if necessary so the user can make changes to the settings. Once the user presses the OK button, the operation starts, halting the execution of the program until it completes.

Vector Field

Figure 34: Vector Field

The Average Optical Flow is a two dimensional array of vectors. The vector at any position describes the average speed and direction of motion for that position over the entire movie. The image produced describes each vector with a circle and a line. The circle represents the origin of the vector. The line represents the direction and magnitude of the vector.

Motion Clusters

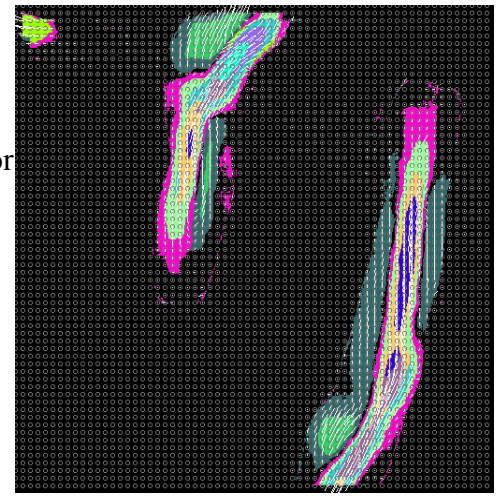

Figure 35: Motion Clusters.

Motion clusters are very similar to the vector field. The vectors in the field are clustered using K-means clustering. The clusters are drawn by choosing a random color for a cluster then assigning that color to each pixel belonging to the cluster. The original vector field is drawn over top the motion clusters.

Strange Motion Map

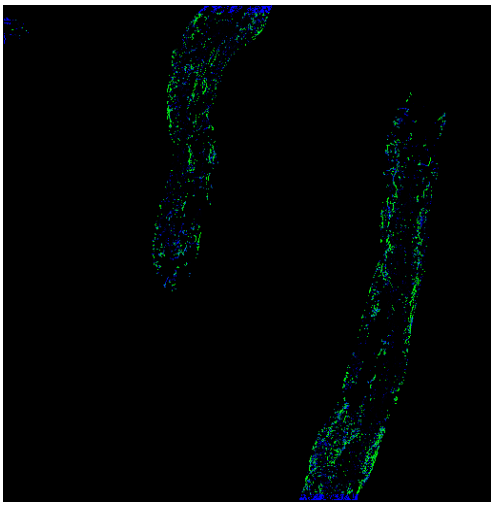

Figure 36: Strange Motion Map.

Strange motion is the difference in motion between the average optical flow and individually tracked features. The strange motion map draws a green dot at every position where the motion is close to opposite the average flow and a blue dot at every position where the motion is closer to 45 degrees off from the average flow.

Track Density Map

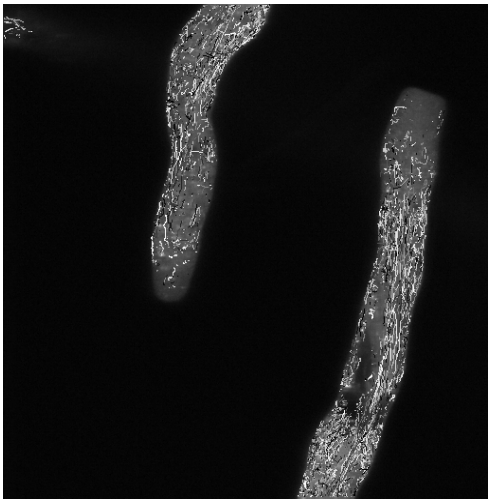

Figure 37: Track Density Map.

The track density map is a quick method of viewing all the tracks found in a given movie at once. This map shows high traffic and lower traffic areas. It can also be used to decide whether the selected feature detection parameters are appropriate to the current pollen tube.

Hough Circle Transform

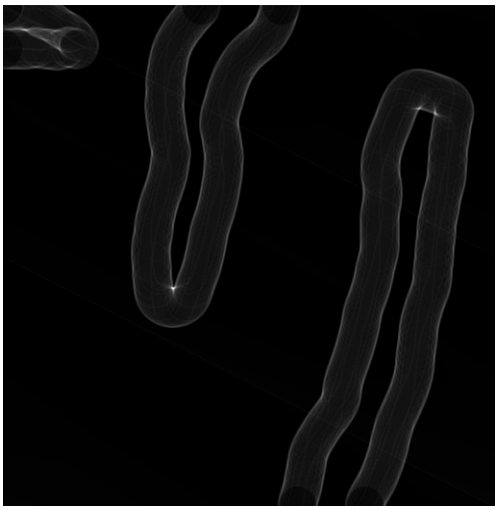

Figure 38: Hough Circle Transform performed on the pollen tube descending from the top of the image.

The Hough Circle Transform is used to find the tip of the pollen tube. It is usually performed on the foreground mask so it requires some pre-processing to work properly on just any pollen tube. The usual method is to apply the following processing instructions in order: Grayscale, Median Filter and Threshold.

Distance Field

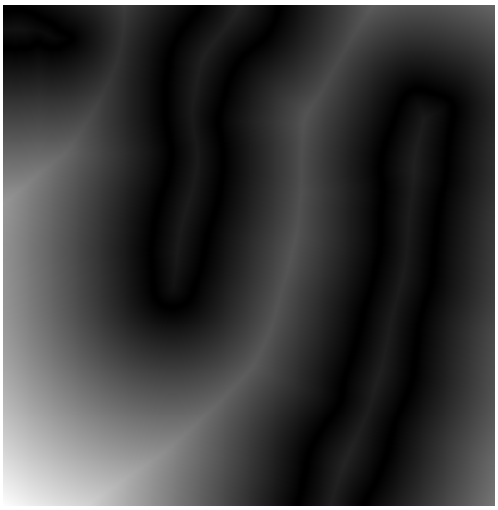

Figure 39: Distance Field.

The distance field is simply the distance at any point to the nearest edge. This can be represented graphically by taking the maximum distance and scaling that to 255 so it equates to a normal, gray scale value. The distance field itself is just a ramp. For a human, more information can be obtained from its gradient.

Distance Field Gradient

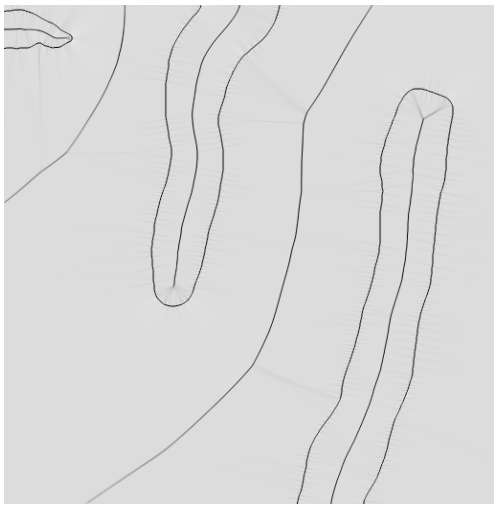

Figure 40: Gradient of the Distance Field.

The gradient of the distance field is the change in distance in the X and Y directions at any point on the distance field. For most of the distance field, the change is constant because the distance increases at a constant rate. Only the ridges of the distance field and the edges of the original image have no real change.

Straighten Effect

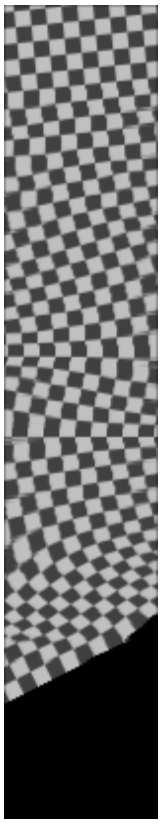

Figure 41: Straightening Effect.

When the pollen tubes were first being straightened, some question arose as to how distorted the original image became due to the texture mapping process. A simple method of testing this was proposed. Instead of just straightening the original image, another image could have the straightening algorithm applied to it in a way that would make the distortion visible. An 8x8 checkerboard image was with the same dimensions as the original is used in this case.

Processing Instructions

The last group on the options window is the processing instructions block. The combo box contains a list of image processing instructions. The plus sign next to the combo box adds the instruction to the list box below. The circled X button removes a selected instruction from the list box. The up and down arrows are intended to change the order of the processing instructions. However, these don't work very well. It works better to simply delete the out of order instructions and add them again. The available instructions are as follows:

- Gray scale: convert a multi-color image to grayscale.
- Straighten: straighten one of the pollen tubes.
- Scale: Change the scale of the pollen tube.
- Invert: Invert the image intensities.
- Threshold: Apply one of seven threshold types.
- Brightness: Apply an additive linear filter.
- Contrast: Apply a multiplicative linear filter.
- Log Transform: Apply a log function to the image intensity.
- Power Law: Combines both gamma correction and brightness.
- Gamma: Apply an exponential linear filter.
- Levels: Combines gamma correction and manual histogram stretching.
- Stretch To Max: Set the brightest intensity in the image to white.

- Histogram Equalization: Spreads out the image intensities evenly.
- Smooth: Apply a Gaussian blur filter.
- Unsharp Mask: Subtract Gaussian blur from the image, sharpening it.
- Median Filter: Apply a median filter to reduce noise.
- Dilate: Choose the maximum intensity from nearby pixels.
- Erode: Choose the minimum intensity from nearby pixels.
- Open: Apply erode and dilate in that order.
- Close: Apply dilate and erode in that order.
- Frame Difference: Subtract the previous frame from the current.

One of the important first steps when applying processing instructions is to first create a copy of the image. This may be done by selecting grayscale, straighten or scale. Otherwise, the linear filters and kernel filters will be applied to the original image each time it is refreshed. Any positive brightness function applied to any image at most 255 times will create a completely white image.

Any number of processing instructions may be applied in any order. However, each instruction is applied to an image in turn each time that image is displayed. The more processing instructions, the slower the program.

LSM information

Six pieces of information about the state of the microscope are available to the user.

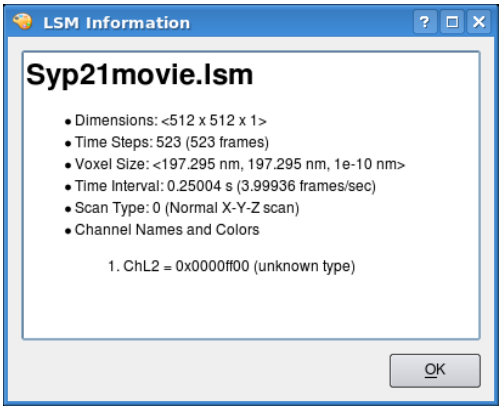

Figure 42: Microscope state information and movie information.

- Image dimensions.
- Number of frames.
- Size in nanometers for a voxel or three-dimensional pixel.
- Time in seconds between frames.
- Scan Type
- Color channels

This information is accessible from the question mark button on the options dialog. The Voxel Size is used by the program to scale two pollen tubes to the same relative size. The Time Interval is used by the program to animate the pollen tube at the same rate relative to real time. The channel names and colors determine which lasers were active and their settings.

The microscope used to create the pollen tube movies used in this project had two lasers. The green channel was used for the laser highlighting the green fluorescent protein and the red channel was used alternately for a red fluorescent protein or differential interference contrast (DIC). Note that the color of the channel is also given. Each color channel represents the gray scale results of one

of the lasers and does not actually represent a color. This is the reason libtiff cannot read an LSM file correctly.

Saving the Current View

The work area of the LSM Viewer is the black window that contains the pollen tube images. The contents of this window may be saved to one of four file formats: PNG, JPG, XPM or PPM. XPM is a simple, text-only format used by the X window system in Unix and included directly into C code. PPM is the portable pixmap format originally created by Jef Poskanzer as a way to transmit images over email. Both XPM and PPM are uncompressed file formats.

The entire black area of the window is saved as it appears on the screen. Any space around the current pollen tube is also saved. This allows multiple pollen tubes to appear on the screen at once and both their images be saved.

Animating the movie

Multiple methods exist for animating pollen tubes. The first has already been suggested. This involves changing the frame number in the options dialog. The

mouse scroll wheel may also be used over the work area itself, animating all the pollen tubes at once. A slider bar also exists to animate all the pollen tubes at once. The slider bar chooses the frame to display based on the number of frames in the movie, regardless of the time interval between frames. The View menu also has a Start Animation option that will start a timer to increment the frames at a given rate. This rate may be changed by the Set Animation Speed option in the View menu. The minimum rate is one frame per second.

When comparing multiple pollen tubes, the relative speeds of the movies may be different. This can be corrected somewhat by choose Use Base Time Step. Faster pollen tube will appear to move at normal speed. Slower pollen tubes will appear to pause between frames, letting the faster pollen tube catch up.

Automatic Feature Detection and Tracking

Starting the automatic feature detection algorithm is as simple as selecting Show Features from the View menu. This will pop up a Feature Detection options dialog. The Feature type determines which controls are enabled. Changing the contrast and edge thresholds will change the number of features found in the first frame. If too more than a few hundred are detected, the program may pause while doing the detection. The edge threshold is meant to determine whether a feature

has too little cornerness in SIFT. In all the other algorithms, the edge threshold is the minimum distance between detected features in pixels. A value of 5 in the edge threshold is usually good. Each algorithm has a different contrast threshold. Once OK is clicked, the program will pause as features are detected. This may take up to ten minutes for SIFT. Other algorithms may take up to one minute. Features are displayed as circles.

Tracking the detected features works exactly the same way as detecting the features. The feature detection options dialog pops up so features may be selected. The tracking algorithm is automatic. Lines are drawn for any detected track up to the current feature. If Keep Old Tracks is selected in the View menu, all the old tracks will remain on the screen if they are at least 10 frames long. Older frames will be drawn in a darker color, gradually fading to black as the track reaches the age of the entire pollen tube movie.

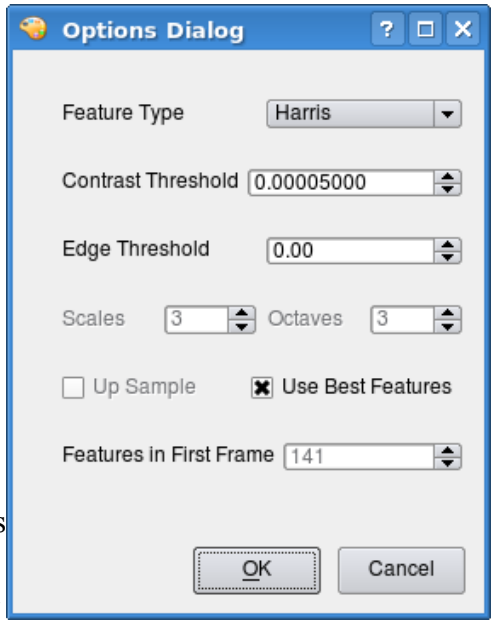

Figure 43: Feature Detection Options Dialog

Unchecking Smooth Tracks in the View menu will remove the five frame smoothing function that averages the position of the detected features. For slow moving features, smoothing reduces the amount of zig-zag motion present.

Unchecking this menu item will show the track in the form that it was tracked.

Manual Feature Selection and Tracking

Manual feature detection may be enabled by selecting the option from the View menu. The user may then add features by clicking on the pollen tube image with the mouse. Features may be removed by clicking on the existing feature circle. Every position selected is automatically saved to the database so it may be called up again the next time this same movie is opened. Manual feature selection is a tedious process. Features may not be moved. They may only be deleted and added.

Manual feature tracking may be enabled by selecting the option from the View menu. This will disable manual feature selection. Tracking is done by choose a feature in one frame, advancing to the next frame and choosing a corresponding feature. This pair of points is assigned as a match. Manually detected features and tracks may be exported to a file and later imported using options on the File menu.

The Settings Database

The settings database is automatically maintained by the program as a file in the same directory as the program called settings.db. This file can be opened using the sqlite3 command line utility. It contains information about the last saved file, the last opened file, the optical flow file directory and others. It contains all the manually selected features and tracks. It also contains the settings used to create the optical flow files that have been cached to the hard drive.

If an optical flow file exists, any function that needs the average optical flow will open that file first rather than performing the time consuming task of recreating it. If the settings change or the settings file is deleted, the optical flow file is recreated whether the file exists or not.

BIBLIOGRAPHY

1. Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes." Copyright 1980, the Computer Science Department at University of Rochester, NY.

2. Carl Zeiss International, "File Format Description – LSM 5.xx, Release 2.0." Courtesy of Karsten Rodenacker, http://ibb.helmholtz-muenchen.de/homepage/karsten.rodenacker/IDL/FileIO.html.

3. Galvin, B., McCane, B. and Novins K. "Robust Feature Tracking." Copyright 1999, Computer Science Department, University of Otago, Dunedin, New Zealand.

4. Harris, C. and Stephens, M., "A Combined Corner and Edge Detector", Copyright 1988 by The Plessey Company plc.

5. Heckbert, P. S., "Fundamentals of Texture Mapping and Image Warping." Copyright 1989, Paul Heckbert. Masters Thesis for Department of Electrical Engineering and Computer Science, University of California, Berkeley.

6. Kuhn, H. W. "The Hungarian Method for the Assignment Problem." Copyright 1955, Naval Research Logistics, Volume 52, issue 1, pp 7 – 21.

7. Lindberg, T.. "Feature Detection with Automatic Scale Selection." International Journal of Computer Vision, volume 30, number 2, 1998.

8. Lord, E. M. and Russel, S. D., "The Mechanisms of Pollination and Fertilization in Plants." Copyright 2002, Annual Review of Cell Division Biology, 18, pp 81 – 105.

9. Lowe, D., "Object Recognition from Local Scale-Invariant Features." Proceedings of the International Conference on Computer Vision, Sept 1999, pp. 1 – 8.

10. Lowe, D., "Distinctive Image Features from Scale-Invariant Keypoints." Copyright 2004, International Journal of Computer Vision.

11. Lucas, B. and Kanade, T. "An Iterative Image Registration Technique with an Application to Stereo Vision." Copyright 1981, Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 674 – 679.

12. Mikolajczyk, K., and Schmid, C., “Indexing based on scale invariant interest points,” Copyright 2001, International Conference on Computer Vision, Vancouver, Canada, pp. 525–531.

13. Moravec, H. P., “Visual Mapping by a Robot Rover”, Copyright 1979, Stanford Artificial Intelligence Laboratory.

14. National Institute of Health, “ImageJ: Image Processing and Analysis in Java.” http://rsb.info.nih.gov/ij/

15. National Institute of Health, “LSM Reader.” http://rsb.info.nih.gov/ij/plugins/lsm-reader.html

16. Otsu, N., ‘‘A threshold selection method from gray level histograms,’’ Copyright 1979, IEEE Trans. Syst. Man Cybern. SMC-9, pp 62–66.

17. Rosenfeld, A. and De la Torre, P., ‘‘Histogram concavity analysis as an aid in threshold selection,’’ Copyright 1983, IEEE Trans. Syst. Man Cybern. SMC-13, pp 231–235.

18. Rosten, E. and Drummond, T., “Fusing Points and Lines for High Performance Tracking,” Copyright 2005, IEEE International Conference on Computer Vision, Volume 2, October 2005, pp. 1508 – 1511.

19. Rosten, E. and Drummond, T., “Machine Learning for High-Speed Corner Detection,” Copyright 2006, European Conference on Computer Vision, Poster Presentation, May 2006, pp. 430 – 443.

20. Sage, D., Neumann, F. R., Hediger, F., Gasser, S. M., and Unser, M., “Automatic Tracking of Individual Fluorescence Particles: Application to the Study of Chromosome Dynamics,” Copyright 2005, IEEE Transactions on Image Processing, vol. 14, no. 9, pp. 1372 – 1383.

21. Sethian, J.A., “Level Set Methods and Fast Marching Methods,” Copyright 1999, Cambridge University Press, pp. 3 – 16, 44 – 48.

22. Shafique, Khurram and Shah, Mubarak, “A Non-Iterative Greedy Algorithm for Multi-frame Point Correspondence.” Copyright 2003, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 51 – 65.

23. Silicon Graphics Inc., “LibTIFF – TIFF Library and Utilities.” http://www.libtiff.org/

24. Smith, S.M. and Brady, J.M. “SUSAN - a new approach to low level image processing.” Copyright 1997, *Int. Journal of Computer Vision*, 23(1):45--78, May 1997.

25. Theodoridis, Sergios and Koutroumbas, Konstantinos. “Pattern Recognition: Third Edition.” Copyright 2006, Academic Press, pp. 631 – 634.

26. Tsien, Roger Y., “The Green Fluorescent Protein,” Copyright 1998, Annual Review of Biochemistry, 67 pp 509-544.

27. Veenman, C. J., Reinders, M. J. T., and Backer, E., “Resolving Motion Correspondence for Densely Moving Points,” Copyright 2001, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 1, pp. 54 – 72.

28. Zhao, Hongkai, “A Fast Sweeping Method for Eikonal Equations.” Copyright 2004, Department of Mathematics, University of California, Irvine.