12-2011

Automated Pollen Image Classification

Nicholas Quentin Haas
nhaas2@utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Nicholas Quentin Haas entitled "Automated Pollen Image Classification." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

J. Douglas Birdwell, Major Professor

We have read this thesis and recommend its acceptance:

Tse-Wei Wang, Roger Horn

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Automated Pollen Image Classification

A Thesis Presented for

The Master of Science

Degree

The University of Tennessee, Knoxville

Nicholas Quentin Haas

December 2011

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

Acknowledgements

I would like to thank my friends and family whose support has helped me get through the challenges of graduate school, Maria Caffrey for her insight into geography and pollen and my supervising professors Douglas Birdwell, Tse-Wei Wang and Roger Horn, whose care and guidance will never be forgotten.

"Only a small portion of book wisdom ever reaches the real world" - Carl von Clausewitz

Abstract

This Master of Science thesis reviews previous research, proposes a method and demonstrates proof-of-concept software for the automated matching of pollen grain images to satisfy degree requirements at the University of Tennessee. An ideal image segmentation algorithm and shape representation data structure is selected, along with a multi-phase shape matching system. The system is shown to be invariant to synthetic image translation, rotation, and to a lesser extent global contrast and intensity changes. The proof-of-concept software is used to demonstrate how pollen grains can be matched to images of other pollen grains, stored in a database, that share similar features with up to a 75% accuracy rate.

Contents

List of Figures

Chapter 1

Introduction

In the study of computer science, the field of computer vision holds great potential for improving society and making utilization of man-hours far more efficient. Since the dawn of human creativity, painters have recorded the fruits of their imagination and the world around them with images long before ideas were conveyed with writing, such as the cave painting in Figure 1.1. As the skills and tools of artists advanced, some specialized in recording images of events, places or people. While the paintings of artists allow the creative mind, the human element, to have free reign, the craft was not without shortcomings. One such painting by a skilled, specialized artist can be seen in Figure 1.2. Average men could not hope to capture moments in their lives beyond that which they could retain in their own thoughts without the services of a skilled artist and allotting a generous amount of time to properly capture imagery. Events that unfolded quickly had to be recorded from the artist's memory or from the sometimes vague description of others. The greatest strength of an artist is that of being able to improve a scene, to add and remove whatever his creative eye desires. Unfortunately, this also meant that the artist could not capture many precise details objectively, when necessary. A breakthrough came with the invention of photography and subsequent improvements to film and cameras. An early example of photography can be seen in Figure 1.3. Soon, capturing images of events and people

Figure 1.1: An ancient cave painting from South Africa, photographed by Valroe (2008)

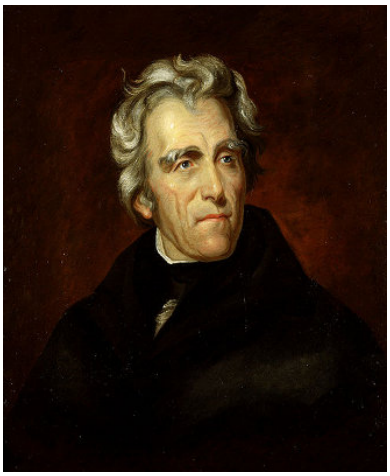


Figure 1.2: A portrait of former US President Andrew Jackson created with modern painting techniques by a professional artist, Thomas Sully (1824)

Figure 1.3: A primitive photograph of former US President Andrew Jackson by Edward Anthony (1845)

became quick, objective and drastically cheaper. Techniques soon developed to edit, enhance and restore images on film and entire new directions for photography opened up. While these techniques offered many ways to alter images to one's liking, the procedures were often slow and expensive to perform with traditional photography. The next revolution in regards to imaging came with the application of computing towards image processing. Traditional photographs could be digitized or captured directly with digital cameras. Digitized images could then be shared or altered using computers and related media.

1.1 Digital Image Processing

According to Gonzalez and Woods (2007), the application of computing towards altering, editing or otherwise enhancing digital images is known as digital image processing. Digitized traditional images that were distorted, noisy, or otherwise lacking in a particular quality can many times be restored or enhanced using digital image processing techniques. In addition, many effects that do not occur naturally

could be added to images and entirely new scenery can be created by blending layers of images together.

An important image enhancement effect is image sharpening. According to Gonzalez and Woods (2007), this procedure highlights areas of significant change, like the edges of objects in an image, and makes such areas more prominent. The related field of edge detection extracts these edges into an edge map, which can be linked together to form the outlines of objects in a rough-sketch of a scene. This process is known generically as image segmentation, as there are many approaches to extracting the significant objects in a scene from a digital image. It is often the first step in an important area of research known as image classification.

1.2 Image Classification

Image classification is a field that attempts to automatically determine the content of a given digital image, according to Gonzalez and Woods (2007), a process that may or may not involve a human observer. This field has numerous applications, even as a filtering step to remove significant numbers of images from consideration prior to a final judgment being given by a human operator. Humans are very good at finding visual patterns and classifying images, but lack the ability to examine hundreds of images quickly. In medical imaging, image classification can be used to detect possible tumors and alert a medical professional that he / she needs to pay close attention to a particular set of images. This reduces the workload on staff and allows for a priority ranking of large sets of images. In fingerprint recognition, images of fingerprints taken from a crime scene can be compared to thousands of fingerprints in law enforcement databases to determine a list of candidates to be examined by a fingerprint expert. There are countless other applications for image classification in which untold batches of images can automatically be analyzed using computer algorithms, monumental tasks for humans given the sheer volume of data in many cases.

1.3 Pollen Classification

Fortunately, many of the image classification applications share common implementations, at least at particular phases. For example, the same image segmentation technique can be used to extract significant objects from photos to be used by a variety of different classification techniques. With the goal in mind of creating a generic image classification system that can be used for a variety of different tasks, one must focus on a particular group of images to evaluate the result of careful selection, implementation and refinement of different components of such an image classification system. In that respect, one turns to pollen grain image classification to find such a challenge. Today, pollen grains from prepared microscope slides are classified and counted by hand in laboratories across the world. If one could develop computer vision software that could aid researchers in this endeavor, resources could be freed and allocated to other tasks.

From a practical standpoint, pollen grains are a good choice for evaluating the effectiveness of a generic image classification system as there is abundant literature available with well prepared and classified pollen grains imaged under a microscope. As described in Section 4.2, these pollen grains fall into distinct morphological groups with identifying features that can be segmented and used to distinguish one from another. In the following chapters, one will describe how one arrived at the selection of the different components to be used by the image classification system, the theory and implementation of the image classification system and the experimental results of its components and as a whole to achieve a 75% accuracy rate for matching a pollen grain with another member of its morphological group and an 85% success rate at ranking a member of the same species in the top 3 results.

Chapter 2

Background

2.1 Chapter Overview

A review of research literature in the fields of image segmentation, representation and classification is presented in this chapter. In each section, the purpose and ideal traits for each component of the classification system are outlined. Section 2.2 outlines various methods to segment an image into component shapes that capture significant features in an image. Section 2.3 shows methods that researchers have employed to represent images / shapes for analysis, classification and storage. The process by which the approaches implemented in the proof-of-concept demonstration software described in Chapter 3 is documented.

2.2 Image Segmentation

2.2.1 Objectives of Segmentation

The first step in the image classification problem decomposes the image into a subset of smaller, isolated subimages of features (objects) present in the original image. Note that due to the conceptually recursive nature of the approach to segmentation, one uses the terms image and subimage interchangeably. The subimages can be further

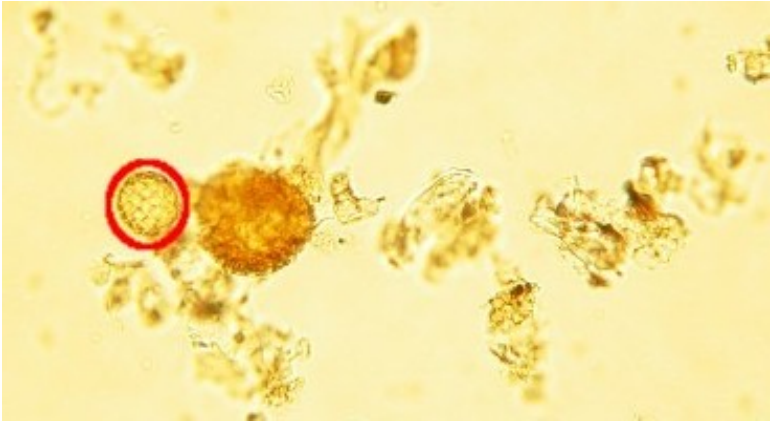
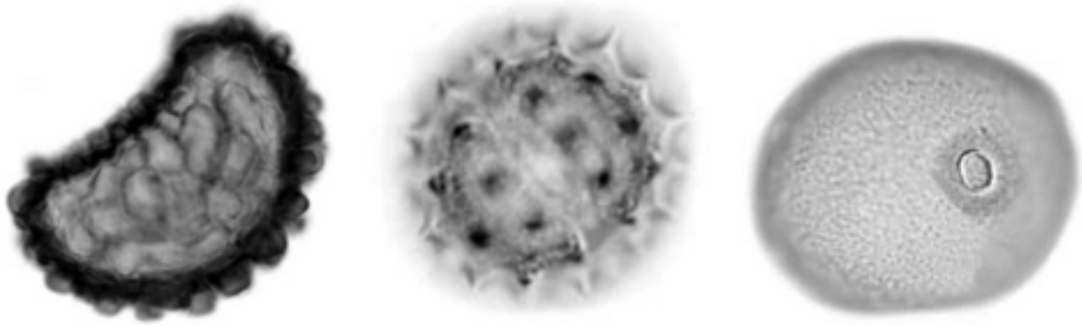


Figure 2.1: Pollen grain cluttered with debris as viewed under a microscope. Prepared and imaged by Caffrey (2010)

decomposed into subimages under certain conditions. There are two main reasons that one would ideally first decompose a larger image into several smaller subimages. First, one wishes to remove the background of an image from consideration. Second, one desires that the internal features of objects present in the target image be isolated into independently stored subimages.

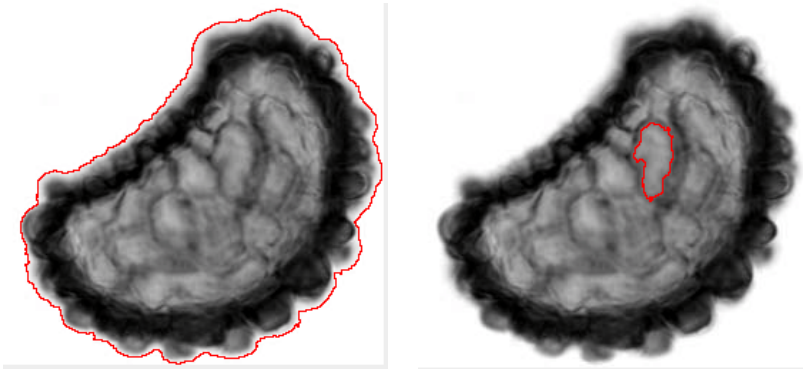
One defines the background of an image as the superfluous imagery that surrounds target objects (or features). In the case of a subimage, the background is its parent image. The background of an image might contain objects that are not part of the target, noise, redundant information (in the case of subimages), white space or other unneeded information. If background imagery is not properly separated from the target's imagery, then it can have a negative impact on the classifier as it will be incorporated as noise in the classification process and distort the potential accuracy of the classifier. For example, one would wish to only consider a pollen grain in a microscope slide (circled in red in Figure 2.1) and discard debris that might surround the pollen grain in a given sample that was not fully cleansed prior to imaging (debris surrounding the red circle in Figure 2.1). Features inside a subimage (segment) can be thought of as distinguishing, observable characteristics present inside an image. These features, for example, could be pores on the surface of a pollen grain that might be a distinguishing feature of that particular species and differentiate it from



(a) Pollen 1 (b) Pollen 2 (c) Pollen 3

Figure 2.2: Various pollen grains collected and imaged by Willard et al. (2004)

non-pollen grains or those of the same genus, such as the rough ridges on the surface of Figure 2.2(a) or the large pore in Figure 2.2(c). With the internal subimages isolated, they can be compared to other images independently. This approach is similar to the classical divide-and-conquer strategy that is applied to many problems in computation in that one decomposes a larger problem into smaller, easier to solve, subproblems. Another inherent advantage of segmenting images into independent subimages is the ability to identify a target amongst several other objects in different contexts or backgrounds. An example of this would be finding a specific species of pollen in a microscope slide that contains multiple pollen species and debris. Successive steps rely upon the successful decomposition of the original image's objects into independent subimages; therefore, the selection of an algorithm to perform this segmentation is critical to the success of the overall classification objectives. Ideally, one wants the segmentation algorithm to produce closed shapes that fully encapsulate the significant features in a given image. One defines a closed shape as the imagery of a significant feature that resides inside a continuous, closed curve (contour) boundary that separates the shape from its background or other shapes that may surround it or encapsulate it. In Figure 2.3 examples are provided of closed shapes of the image Figure 2.2(a), with Figure 2.3(a) being a significant feature's imagery (the complete pollen grain) encapsulated by a red closed contour with a white background. Figure 2.3(b) is a closed shape / significant feature within the shape outlined in



(a) Pollen feature as a closed contour

(b) Pollen body as a closed contour

Figure 2.3: Pollen grain features outlined in red

Figure 2.3(a) (which is also its background) enclosed in a red closed contour. If the segmentation process did not result in closed shapes, there would be ambiguity as to which edges constitute independent features. In turn, these closed shapes contain inside them other closed shapes that recursively capture observable features subject to a stopping criterion typically based on a minimum surface area in pixels for a shape. Segmentation is a subject of ongoing research in the field of computer vision, as it is the first step in a multitude of image processing tasks. As such, there are a wide variety of methods available to implement the segmentation step. In the following paragraphs, previous works are explored in the field of image segmentation, their strengths and weaknesses are discussed, and a method is selected that is judged most applicable to the present application.

2.2.2 Edge Detection

In order to segment an object, one must be able to determine the locations of the edges of an object in an image's border. Ziou and Tabbone (1998) define an edge, in the context of image processing, as points where significant intensity (gray-level) changes occur. Typically, this edge is a change in depth of the surface of an object, an object eclipsing another object, discontinuity, shadow or the outline of an object

against its background. Most edge detectors, algorithms that produce edge maps of an input image, operate on the same general principle. They attempt to exploit changes in intensity across the image to detect the boundary (or edges) of a feature. Ziou and Tabbone (1998) state that two general classes of edge detectors exist, *a priori* and *non a priori*. *Non a priori* edge detectors are generic and are invariant to the type of image presented. *A priori* detectors contain optimizations specific to a particular class of images. Custom tailoring edge detectors to a specific class of images can result in a more effective edge detector, but it is *ad-hoc* and not easily portable to images that are not similar to the ones it was written to process. Since the goal is to create a generic image classification method, only *non a priori* edge detectors are considered, but one acknowledges that it is possible to make a modular system, to accept *a priori* edge detectors at this step, if the system is to be adapted to a more specific group of images in the future.

A closely related image enhancement task is image sharpening, which seeks to emphasize the boundaries of an image. Gonzalez and Woods (2007) liken image sharpening to spatial differentiation, as one is highlighting areas of discontinuities (high degree of intensity change) and deemphasizing areas with a slowly changing intensity level. As such, the gradient (direction of the greatest increase in intensity or first order derivative) and Laplacian (second order derivative) can both be used. Gonzalez and Woods (2007) show that approximations of these first and second order sets of derivatives can be calculated over a digital image by making use of image masks. In Equation 2.1 and Equation 2.2, the approximation formulas for the 2-dimensional discrete Sobel and Laplacian operators are shown, both derived in the literature by Gonzalez and Woods (2007).

$$\Delta f_x = \frac{1}{2} * f(x+1, y) + f(x-1, y) \tag{2.1a}$$

$$\Delta f_y = \frac{1}{2} * f(x, y+1) + f(x, y-1) \tag{2.1b}$$

A_{11}	A_{12}	A_{13}
A_{21}	A_{22}	A_{23}
A_{31}	A_{32}	A_{33}

(a) 8-Pixel Neighborhood

B_{11}	B_{12}	B_{13}
B_{21}	B_{22}	B_{23}
B_{31}	B_{32}	B_{33}

(b) 8-pixel Mask

Figure 2.4: Image mask and pixel neighborhood visualizations, target pixel at the center

$$\Delta^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4 * f(x,y) \tag{2.2}$$

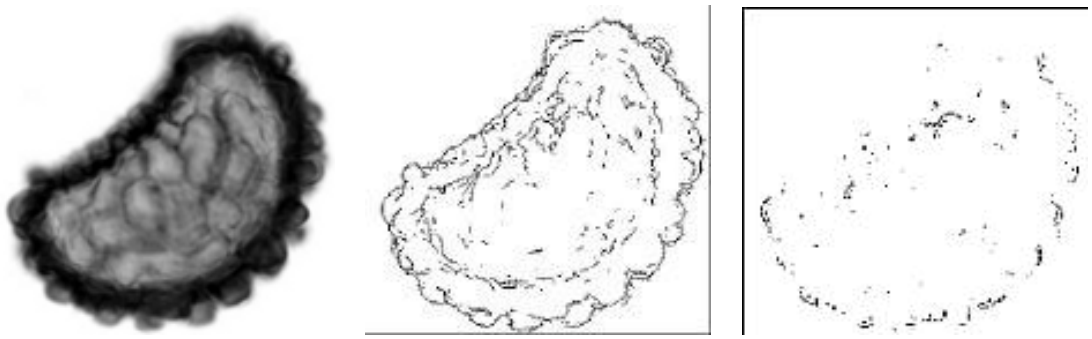
According to Gonzalez and Woods (2007), the direct processing of pixels inside an image is said to occur in the spatial domain. The location of each pixel is referenced by its discrete position in a matrix form of the image, analogous to Cartesian coordinates. This form is quite intuitive as it is close to how humans find locations on maps and spatial representations can quantize components perceived in human vision, such as hue, saturation and intensity. Gonzalez and Woods (2007) demonstrate that masks (Figure 2.4(b)) are small (in relation to the overall size of the given digital image) square matrices that contain weights for a pixel neighborhood (Figure 2.4(a)) in the image around a pixel. Typically, this mask will be centered about a target pixel, and the intensity values of surrounding pixels will be multiplied by the weight of the cell in the mask's matrix that overlays it. The sum of said products then creates a new value for the target pixel (Equation 2.3). This operation is performed for the remaining pixels in the image, with a special case for border regions. Prior to applying the edge detector, an image smoothing operation is typically performed. As indicated by Gonzalez and Woods (2007), smoothing can reduce discontinuities in edges and reduce the impact of noise.

$$C_{22} = \sum_{i=1}^{3} \sum_{j=1}^{3} A_{ij} B_{ij} \tag{2.3}$$

Gonzalez and Woods (2007) state that first order derivative masks typically produce fewer thick edges and second order masks typically are much more sensitive and produce more frequent, thinner edges. Two classic edge detection techniques using masks derived from first and second order derivatives are the Sobel and Laplacian edge detectors. In Figure 2.5 the original pollen grain is shown, along with the pollen grains edge maps generated with the masks for the Sobel and Laplacian filters. In Figure 2.5(b) and Figure 2.5(c), one can see that the first order Sobel filter produces much thicker lines than that of the Laplacian filter. The lines in the Laplacian filter are so fine that there is a significant drop in connectivity between then. This leads to another problem, the fact that the edge detectors are not forming closed shapes. Some detected edges are simply line segments or even dots. Partitioning important features into isolated subimages is critical for the intended approach to classification. These simple, low-level filters / edge detectors are not suitable for fully isolating the closed features one intends to use to classify the images, in part due to their bottom-up nature. A new linking or boundary tracing stage must be added after an edge detector has completed its operations. This additional stage adds extra complexity, degrades performance and according to Kass et al. (1988), may reduce the accuracy of the classifier because mistakes (improper line detection) are difficult to detect and correct at low-levels and tend to have great ramifications as they propagate to new stages later in the classification process.

2.2.3 Explicit Active Contour Models

To solve the problem of segmenting an image into a set of meaningful objects without the addition of a costly new linking stage (when the system has *no a priori* knowledge), higher level active contour models were developed, which are commonly referred to as probabilistic snakes, in their explicit form. According to Kass et al. (1988), active contour models solve the problem of generating closed shapes by reformulating the image segmentation problem as an energy-minimization function



(a) Original pollen grain from Figure 2.2(a)

(b) Edge map of Figure 2.5(a) with Sobel Filter

(c) Edge map of Figure 2.5(a) with Laplacian Filter (darkened for clarity)

Figure 2.5: Segmentation of a pollen grain with two traditional edge detectors

and evolving a curve to fit the boundary of objects in an iterative manner. When visualized, the curve converging to the outline of an object in the image can appear to slither due to iteration towards its local minima. The snake does not attempt to capture the outline of objects on its own. An explicit contour model, which usually takes the form of a spline (piecewise-polynomial function) can be produced by higher-level processes to ensure that the curve evolves towards its local minimum. Higher-level processes can also be used to determine starting points for the snake, after which the snake can fit itself to to the nearest salient curve.

The energy-minimization function presented by Kass et al. (1988) of an explicit active contour model parameterized by $v(s) = (x(s), y(s))$ in Equation 2.4 takes into account the internal energy of the spline, the image forces acting upon the snake and the forces of external constraints. Piecewise smoothness is insured by the internal forces of the spine. The forces behind the image serve to attract the snake towards the salient features inside the image and the constraint forces place the snake near the targeted local minimum. The source of the forces can be input manually or be derived from higher-level processes without user intervention. The fact that a variety of functions can be used to determine these forces lends a high degree of variation amongst different approaches that can be used, a number of which were demonstrated

by Kass et al. (1988).

$$E_{snake} = \int_0^1 (E_{internal}(v(s)) + E_{image}(v(s)) + E_{constraint}(v(s)))\mathrm{d}s \tag{2.4}$$

Snakes provide a mathematically sound and elegant method to both determine and represent the contour of an object in an image. They unify the detection of various features in an object that had in the past relied on separate filters or algorithms to detect and connect into closed contours. Since the problem is reformulated as an energy-minimization problem, previous work in the field of optimization can be applied to the task of image segmentation. Perhaps most importantly, snakes allow higher-level interpretations to influence the capturing of low-level features, something that is absent in classical edge detectors. But, snakes are not the proverbial silver bullet of image segmentation. According to Weeratunga and Kamath (2004), while the parametric form of a snake is highly effective at handling discontinuities and noise in an image, it has severe limitations in regards to its adaptability of changes in topology. This shortcoming is exacerbated under several special cases and can be a severe handicap.

2.2.4 Implicit Active Contour Models

An alternative to active contours is to use implicit level-set approaches rather than an explicit parameterization of the boundary of an object, as with snakes. According to Guichard et al. (2000), level-sets are areas in an image with similar intensity, as defined in Equation 2.5. To illustrate the concept of a level-set and its relationship to the contour (boundary) around a region in an image, view the image as a surface in 3D residing above a select 2D planar region representing the domain of the image, as shown in Figure 2.6. As defined by Weeratunga and Kamath (2004), a level-set, parameterized by an image pixel value $c >= 0$, is the set of points (x, y) in the image domain for which the image's intensity function $f(x, y)$ equals c (Equation 2.5). Various level-lines (the active contours) are shown projected down to the 2D

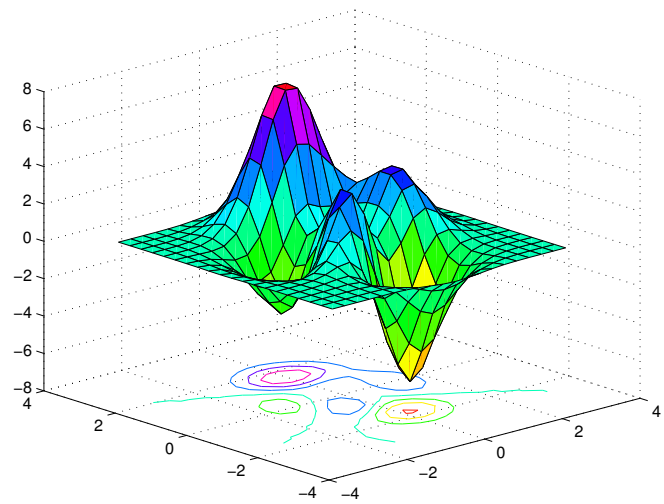


Figure 2.6: Select 2-dimensional Level-lines of the same color as their level-sets in the 3-dimensional terrain shown. Image generated using code based on Matlab documentation by MathWorks (2011)

plane in Figure 2.6. The select intersections of level-set slicing planes of the terrain pictured are colored to match the level-lines plotted on the base plane.

$$\{(x, y) \subseteq f : f(x, y) = c\} \tag{2.5}$$

Guichard et al. (2000) demonstrate via the superposition principle that superposed level-sets are sufficient to reconstruct the original image. An example of select level-sets in a 2D pollen-grain image can be observed in Figure 2.7 in which the same number represents the same level-set. A brute force, yet highly intuitive, approach to capturing level-sets is to use thresholding, as described by Gonzalez and Woods (2007), in which one captures pixels of a certain intensity by applying minimum and maximum bounds to the intensity function and discarding pixels that do not map to values within that range. The result is a binary mask of the same size as the original image with 1 representing pixels in that range and a 0 representing all other values. A visualization of such a mask can be seen in Figure 2.8. This map can be used with a logical AND operation to determine pixels within the level-set. This naive approach is

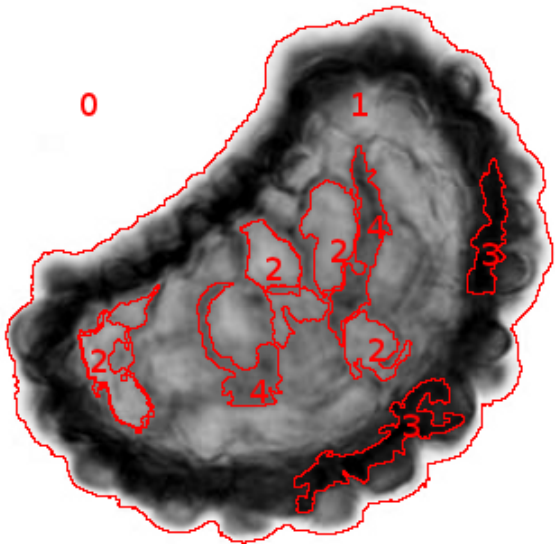


Figure 2.7: Highlighted level-sets of the pollen grain in Figure 2.2(a)

Figure 2.8: Pixel intensity thresholding of the pollen grain in Figure 2.2(a) and the resulting level-set

not only computationally expensive, but also requires the setting of image dependent parameters (such as thresholds as the intensity histogram of two images can be quite different). Due to these drawbacks, this straightforward approach is not satisfactory. A method described by Weeratunga and Kamath (2004) known as implicit active contour models, based on level-set methods originally proposed by Sethian and Osher (1987), offers improved performance over the naive approach. The snakes and implicit active contour models share significant common ground, as both attempt to minimize the energy associated with driving a line to the outline of a given object in an image.

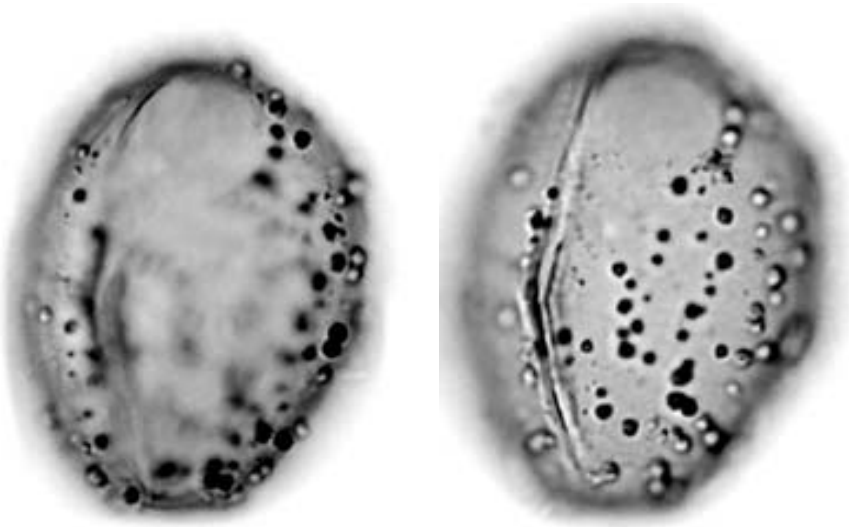


Figure 2.9: A pollen grain with varying fine focus, collected and images by Willard et al. (2004)

The advantage of an implicit level-set approach is that changes in the topography are handled automatically by the algorithm. This avoids pitfalls present with the explicit approach. Any object with peaks and valleys of significant height / depth inherently display sizable changes in topography. Since the objects being photographed under a microscope are not flat, this topological change is noticeable, especially if one was to use the fine-adjustment focus dial on a microscope to view different portions of the pollen grain, as shown in Figure 2.9. According to Weeratunga and Kamath (2004), a drawback to the implicit approach is that it loses some of its robustness in dealing with discontinuities in the outline of an object in an image that might result from noise, angle, vantage point or other conditions. Improvements of the level-set approach over traditional edge detectors were noted in a survey paper by Weeratunga and Kamath (2004) in which they compared the approach with traditional edge detectors on a set of pollen grain images. They found that the implicit active contour approach, while computationally more expensive (despite the traditional approaches requiring an extra smoothing operation prior to segmentation), provided the closed contours

needed to, without ambiguity, discern the proper outlines of objects such as pollen grains.

2.2.5 Fast Level-Line Transform

The performance issues encountered with the implicit active contour approach (Section 2.2.4) were addressed by Monasse and Guichard (2000) with the introduction of a new algorithm known as the Fast Level-Line Transform (FLLT). The primary speed gains arise from exploiting the tree-like structure of the interior of level-contours. Two complementary trees of upper (Equation 2.6a) and lower (Equation 2.6b level-sets are constructed via a region growing algorithm. The two level-sets are specific cases of generic level-sets defined in Equation 2.5. The interiors of shapes in each tree likely contain holes, which are actually shapes in the complementary tree that belong to the other level-set. The trees are merged when shapes corresponding to holes are moved from one tree to the other, resulting in a single tree-of-shapes with a hierarchy based on the geometric inclusion of its component shapes.

Let f be an image, $c_l, c_u \in \mathbb{R}$ be intensity bins where $c_l <= c_u$ and (X, Y) are points in f. Their l_u is an upper level-set, and l_l is a lower level-set when:

$$l_u = f(X, Y) >= c_u \tag{2.6a}$$

$$l_l = f(X, Y) <= c_l \tag{2.6b}$$

2.2.6 Conclusion

It is with implicit active contours determined by the FLLT method that one arrives at a suitable segmentation procedure from which to proceed with the classification problem. To summarize, pollen grains, as well as practical real-world objects, are three-dimensional objects. As such, their images exhibit significant changes due to

topographic variation, focus and perspective. The benefits of active contour models over traditional edge detectors mainly stem from the fact that they allow one to generate unambiguous sets of closed shapes without an additional linking stage. The clear separation of objects by closed contours enables one to consider shapes, during a classification procedure independently of other shapes, producing a divide and conquer approach to classification. This also enables the searching for candidate segments of multiple target segments in parallel.

2.3 Shape Representation and Description

2.3.1 Objectives of Shape Representation

A standardized method of representing, describing and structuring the segmented features of an image is necessary for storing and analyzing shapes. A number of such methods have been proposed to represent and describe shapes, each with their own strengths and weaknesses. The desirable traits of a shape representation, methods used to represent and describe shapes from previous research, and a robust way to structure the shape representations are described in this section.

According to Gonzalez and Woods (2007), shape representation falls into two broad categories. One attempts to represent a shape based on characteristics of the region of pixels inside a shape, another seeks to represent it based on its boundary. Zhang and Lu (2002) note that region based approaches contain more information, as the entire shape's surface is stored and utilized in the comparison. Contour based approaches utilize only information in the boundary of the shape, but benefit from increased simplicity and performance. The representation itself does not necessarily equate to a descriptor, which contains the data employed by a classifier. A descriptor follows a representation. An example by Gonzalez and Woods (2007) states that a shape can be represented by its boundary and described by features of that boundary, such as the distance of that boundary from the shape's centroid at various angles.

For any practical application of an image classification system, storage and performance should be considered highly relevant. The minimization of storage space is highly desirable, as a database of images (and any associated metadata) can grow quite large as one inserts more content, especially if the image is broken down into sets of subimages. If the data structure stores shape representations in such a way that the individual segments can fully reconstruct the image, then there will be no need to store redundant information, such as the original image or data present in other shapes.

The way that shapes are geometrically included within each other in an object can prove useful in the classification process. For example, Pan (2007) showed that metrics based on differences between a shape and a shape immediately inside it (child) can be useful in quickly removing significantly different shapes from consideration during shape comparison. A data structure that captures this hierarchy and allows a quick traversal of a shapes "lineage" has potential speed benefits during the classification phase. An effective data structure to represent shapes is not enough to optimize a collection of shapes. Consideration must be given as to how an individual shape is described such that it is in a form already useful to the classifier.

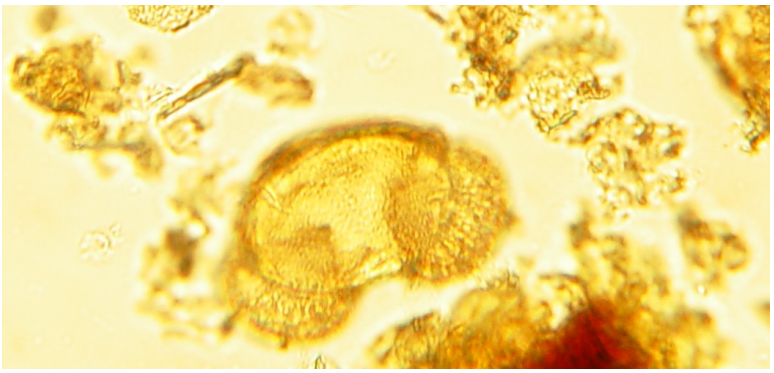
In most cases, two images taken separately of the same object will not be exactly the same. Changes in the object's location (different background), rotation, position (within the image), scale and lighting typically occur between image samplings. For example, given two microscope slides of a core sample containing related pollen grains, the differences in the imagery is quite drastic due to differing origins, separate preparations, different microscope lighting and magnification. Depending on the location from which the pollen was gathered, different debris (plant, animal and inorganic matter) will be present and the preparation process cannot reasonably remove all of such debris prior to imaging. The pollen grain can be located at any position or any rotation on the slide. In Figure 2.10, related pollen grains are shown in two different samples. The grain in Figure 2.10(a) is significantly more magnified than the one in Figure 2.10(b), leading to a change in scale. Their rotations about their

centers and locations on the slide are significantly different, the superfluous debris surrounding them have changed, and the lighting differs. Individual transformations of a pollen grain image are shown in Figure 2.11. For the classification system to match the two pollen grains in Figure 2.10, it requires a descriptor that is invariant to translation, rotation, scale, background and global contrast and intensity changes, as comparing the pixels of their segments directly will not produce meaningful results, in part due to these differences.

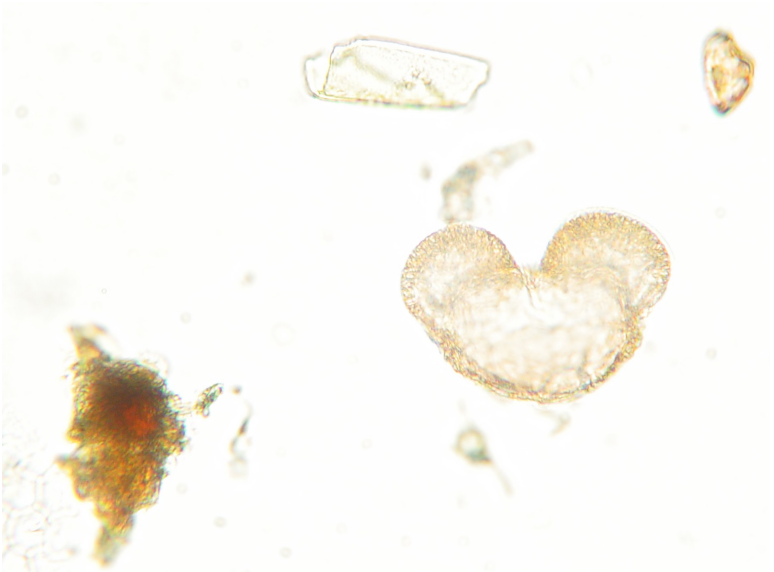
2.3.2 Image Registration

A naive approach to image classification is to directly compare a pair of images using a similarity measure. This is ineffective because the similarity measures are not able to compensate for all of the transformations described in Figure 2.11. These problems are typically mitigated by attempting to place the images into a common coordinate system via an image registration process. Brown (1992) described selection of a classification process as deciding on a feature space, similarity measure, search space, and search strategy . These different components each have different approaches that can be utilized, leading to a large variety of techniques.

Each of these categories is individually examined to better understand how to construct a good classifier from the different components. Brown (1992) defines a vector or object in a feature space as the information extracted from an image that will be used in the comparison. An example is a vector of coordinates representing the border of a subimage in the original image. The search space is defined as the set of transformation methods employed to align the two images. The search strategy is the procedure employed to determine which transformation from the search space should be employed to arrive at the proper alignment, and the similarity measure is the comparison technique used to compare to registered images and generate a value, typically between 0 and 1, that states how similar the algorithm determined the two shapes to be.

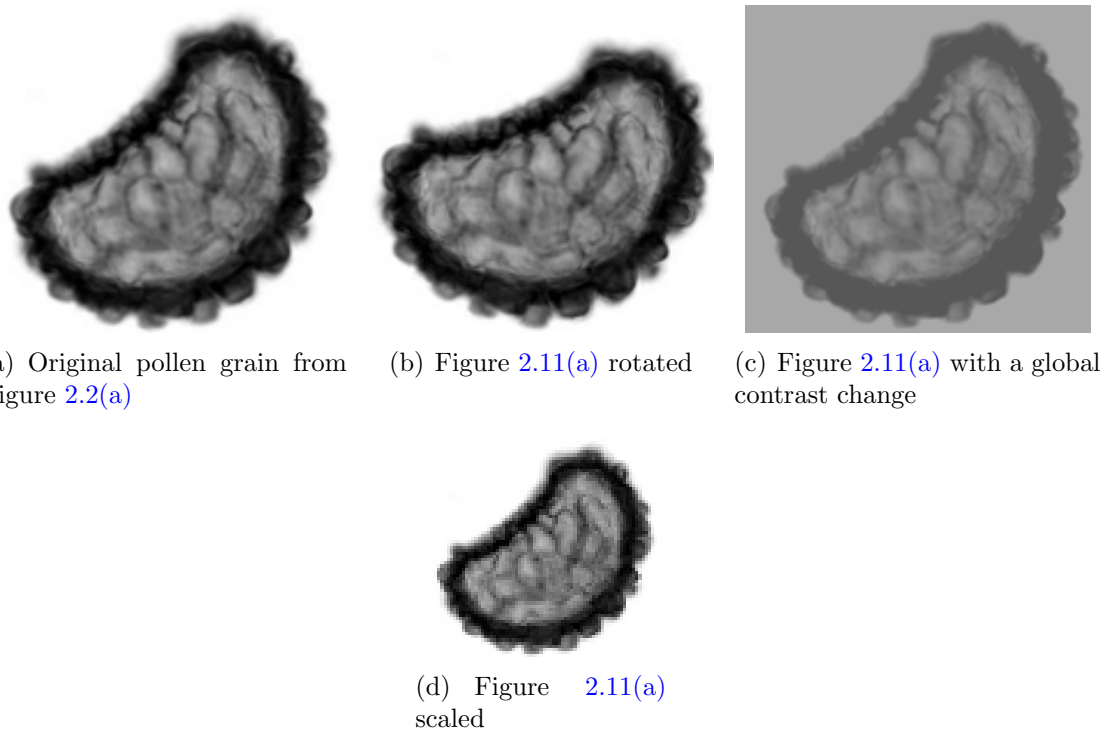


(a) A pollen grain amongst debris in a microscope slide



(b) Similar pollen grain in another sample

Figure 2.10: Related pollen grains in two contexts (differing samples) collected and imaged by Caffrey(2010)



(a) Original pollen grain from Figure 2.2(a)

(b) Figure 2.11(a) rotated

(c) Figure 2.11(a) with a global contrast change

(d) Figure 2.11(a) scaled

Figure 2.11: A pollen grain with various transformations applied

According to Brown (1992) there are typically three types of problems associated with aligning a pair of images taken under different conditions. One is a misalignment, typically caused by a different position from which the images were taken. Typically, this misalignment is solved by a transformation of the image in the spatial domain. Care must be taken to select the proper transformation, as a poorly chosen transformation can further distort the image and lead to information loss, such as the mapping problem of many affine transformations of images described by Gonzalez and Woods (2007). Poorly chosen transformations can also increase the misalignment present, making the problem worse. Next, there is the problem of correcting differences caused by the conditions under which the images were acquired. This might include noise, superfluous imagery or differences in lighting. Finally, the object being observed might have moved between the photographs being taken, or new objects may be present (such as different pollen grains present in a microscope slide). The last two types cause the images to be different. Registration cannot solve

the problem directly, and one must find workarounds as the image can no longer be transformed into something almost identical to the original image.

Use of a similarity measure, such as cosine similarity (Equation 2.7), on a perfectly registered pair of images can be quite effective. However, as Brown (1992) mentioned in her survey paper on image registration methods, many images cannot be globally transformed perfectly due to the removal, addition or changes to features in the image that might have occurred between imaging. For example, consider the two microscope slides consisting of related pollen grains from different samples in Figure 2.10. It is not, in general, possible to find an affine transformation that precisely maps one grain to the other. While image registration techniques have their place, such as landscape identification or pictures of single objects that are taken in a somewhat uniform way, it seems that their added complexity and uncertain reliability preclude this as a satisfactory approach. Let θ be the angle formed between the two attribute vectors $V_{candidate}$ and V_{target}.

$$R = \frac{V_{target} \cdot V_{candidate}}{\|V_{target}\| \|V_{candidate}\|} = \cos\theta \tag{2.7}$$

Once specific objects are segmented from an image, the approach becomes more attractive. One can use certain image registration methods to compare the subobjects individually once the added clutter has been removed. Most image registration approaches would be computationally expensive and thus not advisable for an initial search step. Finally, by limiting our search space to portions of the subimages, such as the boundary, one can further reduce the computational overhead and the space required to store alternative representations of the objects.

2.3.3 Edge Maps

Another way to represent an image in the spatial domain is by means of its significant edges. Monasse and Guichard (2000) state that typically the image will be smoothed, and then a traditional edge detector will be applied. The resulting edge map is

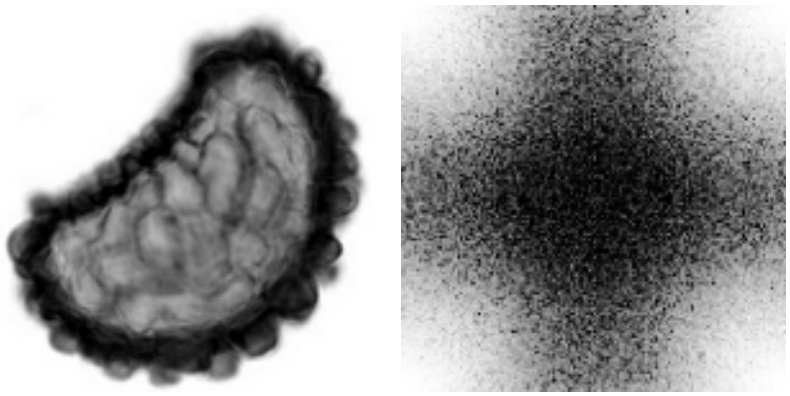
a sketch of the significant detected edges of an image. Methodologies have been developed to extract and represent the segmented shapes. One such approach is the 2.1-d sketch described by Nitzberg and Mumford (1990). In this representation, the linked edges define regions that occupy a layer in an image (usually sorted from the background to the foreground). This form might capture information about which objects are nearest the imaging device; it also creates distinct subimages in the original images by the defined regions in the interiors of boundaries. The advantage of providing the objects in order from background to foreground is not relevant to the objective of this classification system, as a microscope has a very shallow depth of field, and any object significantly nearer or further than the object(s) focused upon will be too out-of-focus to process. The method presented in the original paper also does not seem to consider features within subimages that might be at the same depth as the original object, such as distinctive marks on a surface; therefore, it is unable to capture a shape inclusion hierarchy.

Overall, the use of edge maps generated by traditional edge detectors (Figure 2.5(b) and Figure 2.5(c)) to create representations of an object has several disadvantages. The primary problem is the implementation of a reliable method to link edges into region boundaries. The edges are ambiguous with respect to the geometric shape inclusion hierarchy, as they are not closed. Thus separating interior shapes from enclosing parent shapes becomes difficult and classification metrics based on the relationship between shapes and their immediate enclosing shape become unreliable. Since the edge map does not naturally decompose into clearly defined subobjects (which then become subproblems in the classifier), one also lacks the ability to apply divide-and-conquer techniques to the same degree as other methods. An edge linker can make a pass over an edge map to generate closed contours, but having these two steps operate independently can propagate misdetection or mislinking errors to later stages. One is therefore unable to use higher-level processes (such as the linker) to impact the edge detection process.

2.3.4 Fourier Transformations

According to Gonzalez and Woods (2007), an image (or subimage) can be represented in the frequency domain as opposed to the spatial domain discussed in the section above. In the frequency domain, an image is represented by its Fourier transform and returned to its spatial representation by an inverse Fourier transform. Gonzalez and Woods (2007) state that a Fourier series is a decomposition of a periodic function into a weighted sum of sines and/or cosines of differing frequencies. In the case of digital images (finite and typically, non-periodic), a discretized equivalent formulation of the integral of weighted sines and cosines is used (Equation 2.8). Gonzalez and Woods (2007) then show how the resulting Fourier transform F can be extracted into its magnitude $\mid F \mid$ (Equation 2.9) and its phase ϕ (Equation 2.10) components.

In Figure 2.12(b), a visualization of the magnitude of the Fourier transform of Figure 2.11(d) is provided. Gonzalez and Woods (2007) state that the pixel at the center of Figure 2.12(b) is the average intensity of the pixels in Figure 2.11(d). As one follows the pixels outwards from the center, the lower frequencies represent areas of consistent intensity in Figure 2.12(b), such as featureless areas on the surface or background, and the higher frequencies represent areas of rapid change, like the edges of features in the image or noise. A rotation of the original image will also rotate the Fourier transform, as depicted by Gonzalez and Woods (2007). Once an image is transformed into its equivalent signal space representation, it is operated upon in a global fashion (excluding, for the moment block methods such as JPEG as described in its technical recommendation to the International Telecommunication Union). According to Gonzalez and Woods (2007), several efficient image enhancement techniques (like blurring and sharpening) make use of these properties by manipulating the values over the transform using a filter, then using the Fourier inverse function to restore it to the spatial domain, with results similar to the spatial sharpening filters in Figure 2.2.2.



(a) Scaled (aspect ratio altered) pollen grain in Figure 2.2(a)

(b) Visualization of the magnitude of a Fourier transform of the pollen grain in Figure 2.12(a)

Figure 2.12: Visualization of the magnitude of the Fourier Transform of a pollen grain

Let f be an image of mxn pixels and F be its Fourier transform as defined by its real (R) and imaginary (I) components. F can also be expressed in polar coordinates, defined as follows:

$$F(u,v) = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} f(i,j)e^{-2\pi\sqrt{-1}(ui/m+vj/n)} \tag{2.8}$$

$$F^2 = (R^2 + I^2)^2 \tag{2.9}$$

$$\phi = \arctan\frac{I}{R} \tag{2.10}$$

Using a Fourier transform directly to compare shapes is equivalent to directly comparing images in the spatial domain (without registration) and does not produce a meaningful result in this application due to a lack of similarity measures that can compensate for the affine transformations described in Section 2.3.1. However, approaches using a 1-D Fourier transform (Equation 2.11) of a descriptor of a shape's border signature were shown by Zhang and Lu (2002) to be effective at matching shapes and more robust to noise that might be present in the boundary when compared to descriptors in the spatial domain.

According to Gonzalez and Woods (2007), a border's signature is a one dimensional representation of a shape's boundary. By reducing dimensions, a descriptor becomes less complex, and a similarity measure requires less computation. Conversely, some information is lost, but the task of devising a descriptor that is invariant to the properties outlined in Section 2.3.1 becomes simpler. Many ways exist to derive a Fourier descriptor from the border of a shape. A survey paper by Zhang and Lu (2002) outlines and compares several such methods.

Zhang and Lu (2002) describe four different methods for representing a descriptor based on the Fourier Transform of the border of a shape. All start with a spatial descriptor of the boundary, known as a shape signature, that is transformed into the frequency domain via a 1-D discrete Fourier transform (Equation 2.11). For comparison purposes, the number of points sampled from the border must be consistent across all shape signatures, regardless of their size. A filtering or sampling method is used to reduce the number of samples down to a fixed number. Let f be a vector of values of length n and let F be its Fourier transform

$$F_i = \frac{1}{n}\sum_{j=0}^{n-1} f_j (e^{-2\frac{\sqrt{-1}\pi i}{n}})^j \tag{2.11}$$

The first signature described by Zhang and Lu (2002) is the complex coordinate function, the set of mean-centered complex coordinates of the boundary (Equation 2.13a). The second is the centroid distance function (Equation 2.13d), the normalized distance of points on the boundary from the centroid of the shape (Figure 2.13) by the Pythagorean theorem (Equation 2.14). The centroid has coordinate $(0, 0)$ once the coordinates of the boundary are mean-centered. Mean-centering (Equation 2.12) is necessary to preserve translation invariance. Third is the curvature function, the derivative of the boundary's angles as defined by a sliding window (Equation 2.13b). Finally, there is the normalized cumulative angular function, a measure of the bend of an angle from a fixed reference point on the boundary to the other points along the boundary (Equation 2.13c). In a survey of shape signatures, Zhang and Lu (2002)

demonstrated that centroid distance is the best approach to use for generic shape classification. The following paragraphs demonstrate that a scaled centroid distance description of the boundary preserves significant local and global features of a shape while maintaining the invariances described in Section 2.3.1. Let x_i and y_i be the coordinates of point p_i along the boundary B of Shape S, N be the number of points P in the boundary, V be the shape signature of B, and let k be the length of a sliding window

$$(x_{mc}, y_{mc}) \equiv (x_i - \bar{X}, y_i - \bar{Y}) \tag{2.12}$$

$$CF(p_i) \equiv \arctan(\frac{y_i - y(i-k)}{x_i - x(i-k)})$$

$$AF(p_i) \equiv (CF(p_i) - CF(p_{i-1})) \bmod 2\pi$$

$$v_i \equiv x_{mc} + \sqrt{-1}y_{mc}, \text{ or} \tag{2.13a}$$

$$v_i \equiv CF(p_i) - CF(p_{i-1}), \text{ or} \tag{2.13b}$$

$$v_i \equiv AF(i\frac{N}{2\pi}) - i, \text{ or} \tag{2.13c}$$

$$v_i \equiv \frac{r_i}{\alpha}, \text{ where } \alpha = \frac{1}{n}\sqrt{\sum_{i=1}^{n} r_i^2} \tag{2.13d}$$

$$r_i = \sqrt{x_{mci}^2 + y_{mci}^2} \tag{2.14}$$

In Figure 2.14(a), a shape's boundary is presented with the centroid and two radii labeled. Figure 2.14(b) is the same shape that has been rotated, magnified (scaled) and slightly transposed. With the magnification, the pixel length of the radii are increased consistently (no change to aspect ratio). By normalizing the radii with the norm of the vector of radii (Equation 2.13d) for each shape, the descriptors become invariant to scale. Mean-centering (Equation 2.12) the points

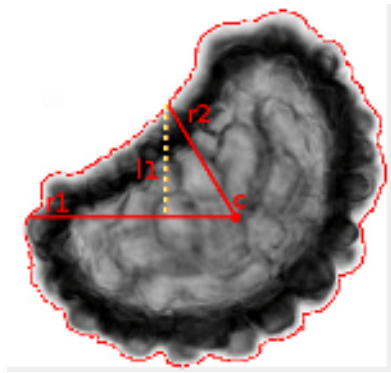
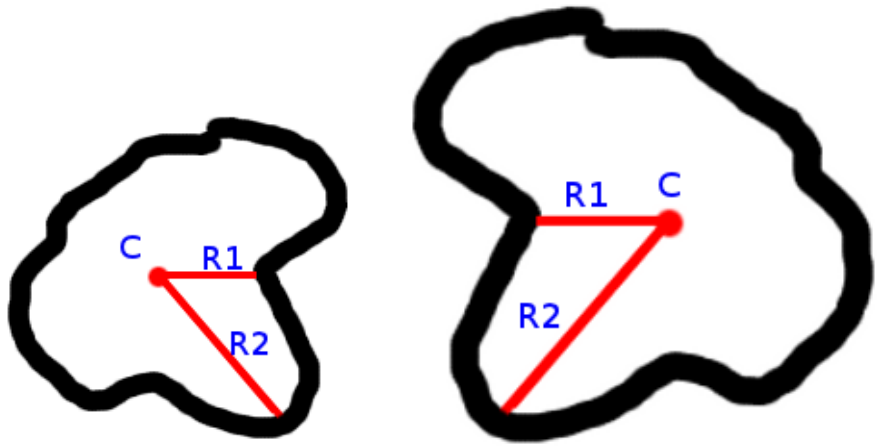


Figure 2.13: Shape from 2.3(a) with centroid and sample radii marked

on the boundary places the center of the shape at coordinate $(0, 0)$ and places the shape in a new basis with the centroid as the origin, making the descriptor invariant to translation. Contrast information is not present as one only uses the distance from the points of the boundary to the center of the shape; therefore, the normalized vector of radii is invariant to contrast changes, as long as the segmentation process that determined the boundary is invariant to contrast. Finally one discards the phase of the Fourier transform by considering only its magnitude, eliminating the presence of angular information and making it invariant to rotation. Once computed, the Fourier descriptor of a shape can be stored in a shape node for quick comparison to other shapes during a search phase using a similarity measure.

2.3.5 Shape Contexts

Belongie et al. (2002) developed a procedure to store shape contexts and use them to compare two shapes. Their approach is to create a circular set of pie shaped bins radiating out from the center of the image. Randomly chosen points along the border are stored in these bins, and the bins are adjusted to be similar to one another (similar to image registration) with a similarity metric applied to the shapes. While their approach showed promise, initially, the literature is vague about how one can account for its stated inherent rotation invariance. However, it can be noted that by



(a) The border of a shape

(b) Figure 2.14(a) with rotation, scale and position changes

Figure 2.14: The boundary of a shape with centroid and two radii labeled

using the silhouette of the shape, one can add information to the system that can be used to better match shapes without adding too much computational complexity.

2.3.6 Global Descriptors

According to Zhang and Lu (2002), global shape descriptors make use of metrics associated with the silhouette of a shape. A metric is typically a real-number that describes a certain aspect of the silhouette, such as area or eccentricity. Metrics of the same silhouette are then placed together in a vector and used as a descriptor. Since the vector is one dimension, it is easy to store and compare against the global descriptors of another shape.

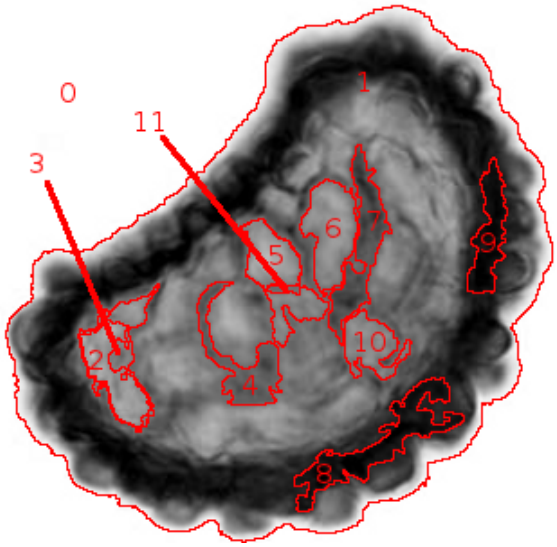
Monasse and Guichard (2000) suggested that relationships between shapes and their interior shapes could be used as metrics for a global descriptor. Pan (2007) successfully used a number of these metrics to return unranked candidate shapes to be considered by another phase of an image classifier. One can devise metrics with invariance to scale, contrast and rotation. For example, the area ratio of a shape

and its interior shape should be relatively similar regardless of scale since this is self-normalizing and no information about rotation or intensity is contained within the metric.

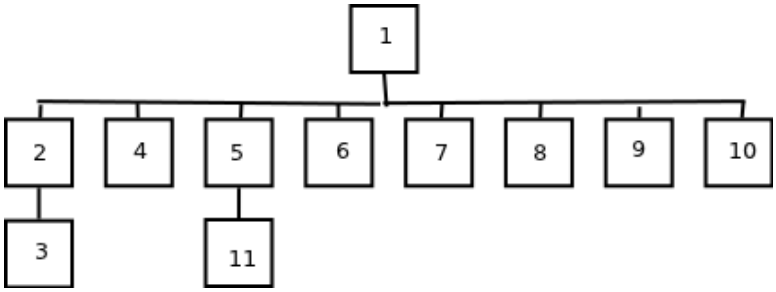
One could generate a ranked list of images using a similarity measure based on the differences between global descriptors of two shapes, but too much information is lost to accurately classify on this alone. Zhang and Lu (2002) showed that global descriptors can only eliminate shapes from consideration that are significantly different from one another, but that they could be used as a quick step to eliminate drastically different shapes from consideration. From an applied standpoint, with the global descriptor referencing its shape, one can execute a single SQL query on a database and return a set of candidate shapes with shapes significantly different from the target shape absent.

2.3.7 Level-Lines and tree-of-shapes

The FLLT Algorithm described in Section 2.2.5 addressed not only the performance issues of implicit active contours, but also created a way of representing the shapes. The shapes are the connected components of level-sets extracted from an image into a tree-like data structure called a tree-of-shapes (TOS) as depicted in Figure 2.15. Shapes in the TOS are organized by their geometric inclusion in an image and are separated from the shapes above and below them in the tree by their level-curve boundaries. Ballester et al. (2003) demonstrate that a tree-of-shapes is sufficient to reconstruct the original image using the shape segments in the tree. The tree-of-shapes also preserves the parent / child relationship between shapes and subshapes, making it trivial to traverse the lineage of a shape to derive metrics based on their relationships, as described in Section 2.3.6. In addition, tree-like data structures are well studied and known in the field of computer science. Storage in many database systems and file systems is implemented using tree-structured data, so the FLLT / TOS algorithm can match the requirements of these systems.



(a) FLLT Segmentation of the pollen grain in Figure 2.2(a) with closed shapes labeled



(b) Tree-of-shapes of the shapes in Figure 2.15(a)

Figure 2.15: Tree-of-shapes decomposition of a pollen grain

2.3.8 Conclusion

One can chose to compare metrics associated with images, attempt a comparison on the subimage as a whole or in part, or a combination thereof. With the observations of Section 2.3 in mind, it becomes clear that a two-stage method is required for computationally efficient image classification. The collection of template images (preclassified reference images) in the database is expected to be quite large, so image registration, or similarly demanding comparison technique, on the entire set of image and subimages becomes impractical. By limiting the feature space to a small set of quantized metrics, the majority of significantly different templates can potentially be excluded from consideration in any given search. Metrics based on global descriptors (Section 2.3.6) and relationships between shapes (parent) and their interior shapes (children) are ideal for this first stage as they contain small, fixed length values that can quickly be indexed and queried by database software. To derive these metrics, especially those relying on descriptors based on the relationship of parent and child shapes, the tree-of-shapes described in Section 2.3.7 is the ideal way to represent the image to allow for quick traversal of shape lineage. With the metrics stored together with the shape, one can execute a single SQL query in a database management system to efficiently implement the first phase.

The 2nd phase, involving a similarity measure, is significantly more computationally demanding and is thus performed on the smaller subset of candidate images retrieved in the 1st phase. Fourier descriptors using centroid distance were determined in Section 2.3.4 to be an ideal way to determine shape similarity as they produced good results in the survey conducted by Zhang and Lu (2002). The Fourier descriptors can be represented as a finite set of real values that can be retrieved from the database and compared relatively quickly using cosine similarity (Equation 2.7). Cosine similarity was chosen due to the successes the measure has garnered in the field of data mining when comparing two vectors (in our case, the two Fourier descriptors), as described by Tan et al. (2005).

Once the cosine similarity is computed, a ranked list of candidate shapes to the target shape can be developed. If the similarity measure is high (approaching one), one can be reasonably assured that a proper match has been found; however, if the highest number is rather low, the shape might be unknown or the image classification system might have failed to find a result due to unforeseen shortcomings of the algorithm or the poor quality of the original images.

Chapter 3

Methodology

3.1 Chapter Overview

This chapter documents the methodology used to implement the image segmentation proof-of-concept software demonstrated in Chapter 4. The methodology was selected after the review of various image segmentation and representation methodologies discussed in Chapter 2. Section 3.2 provides details of the concepts used in the approaches selected from Chapter 2. The basic steps of the Fast Level-Line Transform algorithm as well as the classification system are presented in Section 3.3. Finally, details of the implementation of the proof of concept software, the Shape and Image Database (SID), are given in Section 3.4.

3.1.1 Problem Statement

One seeks to automatically classify an object in a target image by comparing it to objects in a database representing a set of template images and producing a list of candidate images found likely to be most similar to the target image, and ranked by a similarity measure.

3.1.2 Summary of Procedure

Database Template Images

1. Segment the given template image to isolate objects of interest
2. Organize the segmented objects into a tree-of-shapes (TOS) based on their hierarchical inclusion within other objects in the original image
3. Calculate relevant TOS metrics and boundary descriptors for each shape, storing them in their respective node in the TOS
4. Upload the shapes to a database
5. Analyze the TOS metrics for each class of images to determine the normalized range of each metric

Target Image Classification

1. Segment the given target image to isolate objects of interest
2. Organize the segmented objects into a tree-of-shapes (TOS) based on their hierarchical inclusion within other objects in the original image
3. Calculate relevant TOS metrics and boundary descriptors for each shape, storing them in their respective node in the TOS
4. Phase 1 Comparison: Query the database to retrieve candidate shapes that are within the ranges determined above
5. Phase 2 Comparison: Compare each candidate shape to the target shape using a similarity measure on their boundary descriptors
6. Sort the results by the generated similarity value and display the ranked list

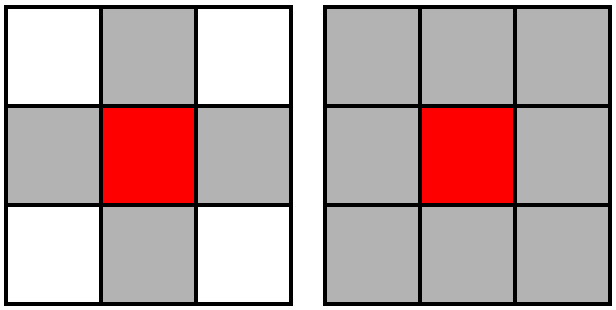
The process begins with the segmentation of a given image into its component objects (shapes) using the FLLT method proposed by Monasse and Guichard (2000). Initially, the FLLT produces two complementary trees-of-shapes, one of upper level-sets and another of lower level-sets based on the pixel intensity of connected components in the image, as described in Section 2.2.5. The FLLT algorithm then merges the two trees into a single tree-of-shapes whose hierarchy reflects the geometric inclusion of the shapes, as shown in Section 2.3.7. Global descriptors (Section 2.3.6) of the shapes and their relationship to their enclosing parent shape are then used to quickly generate a set of matching stored image segments. This step can be performed using a single SQL query on indexed fields to achieve high performance. The Fourier descriptors of the candidate shapes' boundaries (Section 2.3.4) are then compared to those of the target image segment (object) to generate a final, ranked list of images.

3.2 Foundations

3.2.1 Level-Sets

The segmentation step creates subimages (subproblems) from the initial image using the Fast Level-Line Transform (FLLT) algorithm by Monasse and Guichard (2000) first discussed in Section 2.2.5. The algorithm uses a divide-and-conquer approach that uses level-set representations (Section 2.2.4) of regions in the image with the goal of capturing significant features (Section 2.2.1) as shapes separated from enclosed (child) and enclosing (parent) shapes by the level-curves of their boundaries.

As noted in Section 2.2.5, the FLLT algorithm by Monasse and Guichard (2000) iteratively assigns connected pixels to regions in an image that are either part of an upper level-set (Equation 2.6a) or lower-level set (Equation 2.6b) based on their intensity (gray level) during a process called region growing. Essentially, areas of significantly similar intensity are represented by the level-sets of their intensities.



(a) 4-pixel neighborhood (b) 8-pixel neighborhood

Figure 3.1: Two common definitions of a pixel neighborhood

According to Gonzalez and Woods (2007), connected pixels are sets of pixels in an image such that one can travel from any point to another within the set solely by visiting neighboring pixels that are also in the set. A pixel's neighborhood is typically defined by either 4-pixel or 8-pixel connectivity, as shown in Figure 3.1. The neighborhood definition differs between level-sets to resolve certain issues with the FLLT algorithm, as described in Section 3.3.1. Upper and lower level-sets are specific cases of generic level-sets (Equation 2.5). Three important properties of the level-sets in Equations 2.6 are that they are nested, that the upper-level sets are decreasing, and that the lower-level sets are increasing (ordered by set inclusion). As mentioned in Section 2.2.4, the level-sets in an image can produce the original image via superposition. These two properties form a foundation of the tree-of-shapes representation method described in Section 3.2.2.

While the method in which FLLT determines level-sets is shown by Monasse and Guichard (2000) and Guichard et al. (2000) to be resistant to global contrast changes, localized contrast changes cause significant issues with the determination of actual level-sets. That is, if the contrast change is random across the image, then the problem of segmentation becomes more complex. Non-global contrast change is not considered to be an issue in this classification system as a microscope slide image is assumed to have consistent lighting. In other circumstances, localized contrast changes might

be the result of noise or equipment malfunction. With level-sets in mind, one uses representations of pixel regions formed by them to segment an image.

3.2.2 Tree-of-Shapes

The FLLT, as described in Section 2.2.5, places connected pixels in either upper or lower level-sets. By utilizing the connected and nested / ordered properties of upper and lower level-sets, the FLLT represents them in two complementary tree-like data structures (one for upper, another for lower). By this methodology, the algorithm places the shapes as nodes in the tree-of-shapes based on their intensity, therefore their placement in the tree might not be very intuitive to the casual observer, which tends to lend a geometric interpretation to inclusion. In general, neither tree is sufficient to account for all of the shapes in the original image separately. Shapes present in the original image maybe absent in one tree, and thus becomes evident during TOS construction. To resolve issues of lack of geometric inclusion and separate trees, the upper and lower level-set trees are merged at a later step into a single tree-of-shapes using the level-lines of their boundaries to determine their placement.

Monasse and Guichard (2000) showed that level-lines form the boundaries of these level-sets. They can be defined as a set of Jordan curves, as visualized by the red, closed contours of Figure 2.3(b) and Figure 2.3(a). As described by Hales (2007), the Jordan Curve Theorem has an additional descriptor concept, that of interior or exterior positioning. This descriptor, using the boundary of the connected node / shape, allows one a point-of-reference is used to define a shape.

A shape is a node in the tree-of-shapes, it is a set of connected pixels completely enclosed by a level line in an image. Each shape, with the exception of the root image, has an enclosing level-line / Jordan curve that separates it from its parent (as shown by the red line separating region 1 from region 0 in Figure 2.7). The root node is the original image, whose boundary is that of the image itself (as depicted by 0 in Figure 2.7). Disconnected level-sets can exist, as in a case where a significant number

of pixels that belong to different level-sets separates two regions; this causes them to become different shapes. Disconnected level-sets of similar intensity can be observed in Figure 2.7 where the same label number represents the shapes of similar level-sets that are disconnected. While a shape is enclosed entirely in its Jordan curve, it might contain holes. A hole is a child shape of a shape in the merged tree-of-shapes, and the boundaries of the holes are the Jordan curves of the child shapes. Prior to merging, these holes belong to different level-set trees; thus, they will not become child nodes until they are merged. For example, regions 2, 3 and 4 are holes or children of region 1 in Figure 2.7.

A parent shape in the merged tree is defined by Monasse and Guichard (2000) as the smallest shape that contains a given shape (see the boundary between regions 2 and 3 in Figure 2.15(b)). No child is allowed to have more than one parent, as that would also indicate that two unrelated level-sets are occupying the same space. To better understand the tree structure, see Figure 2.15 for a sample tree and the shapes it represents. All nodes of non-solid shapes will have children, except for the leaf nodes (regions 3, 4, 6 - 11 in Figure 2.15(b)) that form the smallest allowed closed contour, which is defined by a minimum surface area threshold. A child shape, being completely enclosed by its parent, has a smaller area than its parent. Therefore, the surface areas of the shapes are monotone decreasing as one descends the tree-of-shapes. The TOS therefore represents a partial order defined by the area of each shape. With these foundations laid, one describes an algorithm to construct the tree-of-shapes.

3.3 Algorithm

3.3.1 Fast Level-Line Transform (FLLT)

To construct the tree-of-shapes as outlined by Monasse and Guichard (2000), one needs to uncover the regions belonging to the various level-sets in an image. Two

trees are constructed, one of upper level-sets and another of lower-level sets. Once their construction is complete, they are merged by reconciling holes present in one tree with the shapes in the other. The first task is the determination of the level-sets.

Monasse and Guichard (2000) utilize a region-growing approach in the FLLT algorithm to grow the level-sets via their connected components. A labeled example of level-sets determined by this method can be observed in Figure 2.15(a). From an initial, unvisited, local intensity extremum (maximum for upper level-sets, minimum for lower level-sets) as determined by a line-sweep of the given image, one creates a new object. The object contains three sets of pixels: candidate pixels, region pixels and neighborhood pixels. The type of each pixel is determined by the reference intensity (initially, the local extrema). The pixel having the local extremum's value is added to the candidate set. The pixels in the neighborhood around the candidate set are then determined (Step 1).

As mentioned in Section 3.2.1) and described by Gonzalez and Woods (2007), connected pixel neighborhoods are typically defined as either the 4-pixels (3.1(a)) bordering the center square pixel's sides or as 8-pixels (3.1(b)), including four pixels about the center diagonally. The definition of the connected neighborhood of pixels depends on whether an upper or lower level-set tree is being formed. Monasse and Guichard (2000) show, by counter-examples, that the Jordan Theorem does not hold for upper level-sets if one uses an 8-pixel neighborhood definition for both, and that tree construction for interior regions fails when using a 4-pixel neighborhood definition for both. The solution is to use a 4-pixel definition for one tree and a 8-pixel definition for the other.

Neighbors of the candidate pixels are added to the neighbor set. For the lower level-sets (this is similar for upper level-sets, replacing "minimum" with "maximum"), one determines the minimum intensity value in the neighborhood pixel set, adds any candidate pixels to the region set, and flags any pixels in the region pixel set as having been visited (Step 2). One then compares this intensity value to the reference intensity. Three cases exist:

1. If the minimum of the neighbors is greater than the initial local minimum, then this is likely a border, and one should determine if it is an exterior boundary of the region by checking to see if it contains the top, bottom, left or right most pixel of said region. If it does not contain one of those pixels, then it is a hole, and one retains a reference pixel of this hole for later use and adds the neighbors to the candidate set. Finally, one sets the reference intensity (former local minimum) to the minimum of this neighborhood and repeats Step 2.

2. If the reference intensity is equal in value to that of the neighborhood's minimum, add the neighbors to the candidate set and goto Step 2.

3. If the value is less than the reference intensity, set the intensity of all of the pixels in the region to the value of the local minimum and start the pixel scan anew at Step 1.

After a shape is determined, the holes inside its region must be examined, as they too will contain level-sets. Holes are in fact other level-sets that will form the children of the shape that contains them in the finished tree. One starts with the hole reference pixel, mentioned above, and finds its location in the complementary tree. Monasse and Guichard (2000) uses the following situation to illustrate how to discover the equivalent shape. Consider a hole reference pixel h in an upper level-set L_u whose constant range is C_u. Then, by the resulting complementary trees of the FLLT algorithm, this pixel h must belong to a lower level-set Ll where C_l is less than C_u. The shape that resides in said hole is discovered by first finding the smallest shape in the complementary tree that contains pixel h and following its lineage up through the tree (for the shapes containing h) until the intensity of the shape is greater than or equal to C_u. Monasse and Guichard (2000) outline the following procedure for determining the smallest shape containing pixel h (or any given pixel) by examining the possible scenarios for the shape containing it. First, the pixel might not be in a bounded level-set in either tree, and therefore no shape contains it. Second, the pixel could belong to a bounded level-set in one tree, but not the other, so it belongs

to the one that is bounded. Finally, if both are bounded, then it is assigned to the level-set that resides in the interior of its complementary tree. The reciprocal form of this procedure to fill a hole in a lower level-set follows a similar procedure. This matching of holes to shapes in the complementary tree leads to the merging of the two trees into a single tree-of-shapes.

As Monasse and Guichard (2000) describe in the original paper, the trivial case for merging the trees is one in which a tree has no holes; therefore, the hole-less tree is the merged tree-of-shapes, with the addition of the original image as the root node. If holes exist in both trees, then shapes from one tree must be moved to the complementary tree such that the holes in one are filled. The filled tree becomes the merged tree-of-shapes. For a situation where hole H_1 is in shape S_1, one determines if any of the children of S_1 has a hole containing H_1. If so, the hole is accounted for in the original level-set. If not, the shape in the complementary tree corresponds to H_1, and all of its child shapes, are placed in the tree as a child of S_1.

Monasse and Guichard (2000)'s FLLT algorithm's output is a tree-of-shapes with each node representing a level-set's region ordered in such a way that each child is completely contained inside its parent. With this tree, any pixel selected from the image can be mapped to the smallest shape in the tree containing it. The runtime complexity of the algorithm is based on the number of pixels in the original image, N. The sweep of the algorithm visits each non-flagged pixel once, which is $O(N)$ time. The neighborhood that must be analyzed for each pixel is at most 8 pixels in size and in the worst case, $(N - C)$ pixels are neighbors of the level-set of pixel size C, which is of $O(N)$ runtime. To sort the gray-level of neighbors, a sorting algorithm of $O(N * LG(N))$ can be utilized. Determining if a reference pixel to a hole is required can be done by following the border of a line, which can be done in linear time. The algorithm's runtime is therefore $O(N * LG(N))$. This is clearly superior to using the brute force approach to finding level-sets, as if each level-set is a different intensity, which requires $O(N^2)$ time.

3.3.2 Classification Phase 1

Pan (2007) utilizes a two phase shape classification system to arrive at a final, ranked list of candidate images during a classification procedure with a given target image. The benefits of this approach are obvious from a performance and accuracy standpoint. Performance is improved by filtering significantly different shapes in an initial step and preventing the comparison of said shapes in the next phase, as different information about the shape will be utilized in the two phases. Significantly different shapes are discarded from further consideration in the first phase using a set of floating point metrics that can be compared in linear time without the need for a similarity measure. To improve upon previous work by Pan (2007), several metrics that were susceptible to scale were removed, such as the level of a node in the tree of shapes and the total number of descendants of a shape, which are impacted at the bottom of the tree by a minimum surface area threshold that prevents these small shapes from being considered. The system also now discards shapes whose parents and and children are not sufficiently similar to the target shape's relatives. In the next phase, a more performance intensive comparison of the shape's boundary is used along with a similarity measure to generate a ranked list of candidate shapes from the (typically much) smaller batch of candidate shapes generated during Phase 1.

The Phase 1 metrics are determined after the merger of the two trees and are stored in their respective nodes such that they do not have to be recomputed during the classification phase. Each shape has corresponding values for these metrics. Target shape metrics are queried against a relational database table containing preclassified shapes in a single SQL query. If properly indexed and implemented, this single query comparing floating point metric values between a target shape and preclassified shapes in a database can be quite efficient. One metric (Equation 3.4) was used from Pan (2007)'s work; each is invariant to global contrast changes, position, rotation and scale. Additional metrics, that of interior versus exterior intensity ratios (3.1) and

the surface pixel area ratio of parent to child shape, were devised for this endeavor and retain the same invariance to said transformations / conditions.

Equation 3.1 is a metric that measures the change in intensity of a given level-set from its parent. The invariance to global contrast change is achieved by the aggregating factor of the intensity of the level-set in the enclosing level-set. The metric lacks any information regarding object position, orientation, or scale; therefore, it is invariant to rotation, position and scale. This will help filter objects that are not similar in terms of their intensity changes with their parents. An example would be comparing a target pollen grain that contains pores darker than the pollen grain's body versus a pollen grain that contains pores lighter than the pollen grain's body. Those not containing light pores would be excluded. Let S_t be an unknown target shape, S_p be the set of preclassified shapes, S_c be a set of candidate shapes with $S_c \subseteq S_p$, $M_i \in \mathbb{Q}$ be the metric of a given shape S_i and C_j,$C_k \in \mathbb{R}$ with $C_j < C_k$. The intensity metric is defined as follows:

$$M_{intensity}(S_i) = Intensity(S_i)/(Intensity(S_{i-1}) + C) \tag{3.1}$$

Equation 3.2 is a metric that records the cumulative surface area of direct children of a shape, normalized by its overall area. Since our segmentation algorithm is invariant to global contrast changes, such a change should not impact the number of direct children that a shape might contain. The metric stores no information about the position or orientation of the shape or the children, so it is invariant to rotation. Scale becomes an issue with this metric if one forms it on a shape near the base of a tree, as the algorithm prevents children below a certain pixel surface area threshold from being included in the tree. One can use slack to account for slight variations. This will help filter objects that do not contain a certain number of interior features. An example would be comparing a target pollen grain that contains a number of pores (which are segmented as child shapes) versus a pollen grain that contains no pores with a smooth surface. The smooth pollen grains would be excluded. Let S_j

be the jth child shapes of shape S_i. The cumulative child surface area ratio metric is defined as follows:

$$M_{ccsar}(S_i) = \sum_{j=0}^{n} Child.Area(S_j)/Area(S_i) \tag{3.2}$$

Equation 3.3 measures the surface area ratio of a child relative to (divided by) its parent. The division by the parent's area acts as a aggregating factor that reduces the impact of scale upon the image. The metric contains no information regarding position or intensity (other than the fact the child is inside the parent); therefore, it is invariant to rotation and global contrast change. An example would be comparing a target pollen grain with a large pore (child shape) with a pollen grain with a small pore, the pollen grains containing small pores could be excluded as they do not occupy a sizable percentage of the pollen grain's surface area. The relative area metric is defined as follows:

$$M_{area}(S_i) = PixelCount(S_i)/PixelCount(S_{i+1}) \tag{3.3}$$

Equation 3.4 provides a metric of boundary length, this is basically how large the boundary of a shape is, aggregated by the shape's area. Since it contains no intensity or position information, it is invariant to rotation and global contrast changes. The aggregation of the length of the boundary by its surface area reduces the impact of scale changes for any shape that is reasonably large, measured by number of enclosed pixels (which is guaranteed by minimum surface areas). An example of how this metric could be employed would be in elongated shapes when comparing them to round shapes. This is similar to Equation 3.3, but it does not include information about the parent shape. The boundary to area metric is defined as follows:

$$M_{bs}(S_i) = BoundaryLength(S_i)^2/PixelCount(S_i) \tag{3.4}$$

As briefly mentioned above, there will be very slight differences in how the image is segmented when an image of the same object is sampled at a different time under different conditions. This is mostly due to differences in noise between the two images, or global differences in the sampling caused by translation, rotation or scaling. While the segmentation process is invariant to contrast, rotation and (mostly) scale, such alterations can introduce noise or mapping issues that will cause slight variations on the segmentation process. In the case of scale, the leaves of a tree may be eliminated due to minimum surface area thresholds. This only impacts shapes near the bottom of a tree, which can be compensated for by not including leaves or shapes close to them in the search, save for special cases. The metrics therefore require a small amount of slack to be added to their comparisons to account for this slight, unavoidable variation. This slack is determined by the constants in Formula 3.5 - 3.8, with Formula 3.9 defining the final set of candidate shapes from the set of preclassified shapes.

Let $M(S_t, S_p)$ be the metrics resulting from Equations 3.1, 3.2, 3.3 and 3.4 for Shape collection S_i between target shape S_t and preclassified shape S_p. Let C_1 and C_2 be constants where $C_1 < 1 < C_2$. The Phase 1 search selections shapes S_p in collection S_i that satisfy the following constraints:

$$S_{intensity}(S_t, S_i) = S_p \subseteq S_i \mid M(S_t)C_1 <= M(S_p) <= M(S_t)C_2 \tag{3.5}$$

$$S_{ccsar}(S_t, S_i) = S_p \subseteq S_i \mid M(S_t)C_1 <= M(S_p) <= M(S_t)C_2 \tag{3.6}$$

$$S_{area}(S_t, S_i) = S_p \subseteq S_i \mid M(S_t)C_1 <= M(S_p) <= M(S_t)C_2 \tag{3.7}$$

$$S_{bs}(S_t, S_i) = S_p \subseteq S_i \mid M(S_t)C_1 <= M(S_p) <= M(S_t)C_2 \tag{3.8}$$

The set of candidate shapes that satisfy Equations 3.5, 3.6, 3.7 and 3.8 is given by:

$$S_c = S_{intensity} \cap S_{cssar} \cap S_{area} \cap S_{bs} \tag{3.9}$$

By providing slack to the candidate criteria, the set is likely to contain the matching shape as the metrics are (almost) invariant to global contrast and intensity, rotation and scale changes. The problem is that a sizable number of shapes that do not match the target shape are also returned, as a significant amount of information is lost when limiting the comparison to these metrics. The metrics are determined during the construction of the tree-of-shapes and stored in their respective nodes, requiring $O(N)$ operations for N shapes. Searching the database to determine S_c, the set of candidate shapes takes at most $O(N)$ where N is the number of shapes in the preclassified shape database. With a database that indexes the shapes by their metrics, the performance approaches $O(LG(N))$ time by exploiting the sorted tree structure of the database index. At the conclusion of Phase 1, a set of candidate shapes, S_c, has been determined, that, in practice, is significantly smaller than the collection of preclassified shapes.

3.3.3 Classification Phase 2

The final phase of the classification procedure involves an intensive analysis of the boundary of the shape. With the inclusion of Phase 1, this procedure is only performed on a limited number of candidate shapes, with shapes drastically different (measured using the metrics defined above) removed from consideration. A transformed representation of the structure of the border is used to determine the similarity between two boundaries. The boundary is represented in the frequency domain rather than the spatial domain. From this representation, one can remove the phase by considering only the magnitude of the Fourier transform to obtain rotation invariance and use cosine similarity to generate the ranked list.

One first creates a vector of values that represent the details of the border, which is currently stored as a vector of coordinates of the original image, in marching order, inside the node of the tree-of-shapes that represents that shape thanks to the FLLT algorithm. The alternative representation is based on centroid distance

descriptors described by Zhang and Lu (2002). It can be pre-computed when each shape is stored in the database. The border cannot be used directly, as a direct comparison of the borders of the target and candidates would not be invariant to scale or rotation. Invariance to global contrast changes is already accounted for as the border comparison will not utilize any intensity information for its comparison.

As originally shown in Section 2.3.4, to begin the construction of this new representation, one first determines the center of the shape by mean-centering the points of the border (Formula 2.12), after which the coordinate $(0, 0)$ is the center of the shape (see C in Figure 2.13). For each point along the boundary, the radius in pixels (see r1 and r2 in Figure 2.13) is computed and stored in a vector of the same size as the number of points in the boundary array using Equation 2.14. To make the system invariant to scale, the radius values are normalized by their combined magnitude or l_2 norm (Formula 2.13d). A cubic spline filter is used to sample a fixed number of points. As described by Zhang and Lu (2002), the reduction is necessary to utilize a consistent number of sample points for all shapes, regardless of size of the boundary, as a normalization procedure, for noise reduction (smoothing), and to set the number of sample points to a power of two (required by fast Fourier transform, or FFT). The equation for cosine similarity requires the lengths of the two vectors (candidate and target) to be the same.

This vector of normalized radii is transformed into the signal space using the magnitude of the one-dimensional discrete Fourier transformation (DFT) as stated by Gonzalez and Woods (2007) in Formula 2.11. The FFT is used for performance. The phase is not used. This eliminates all dependence on rotation, making the resulting Fourier descriptor (FD) invariant to rotation. After this operation, a fixed length vector containing a descriptor of the boundary that is invariant to global contrast, rotation and scale changes has been obtained and can be stored with its node in the tree-of-shapes for expedited comparison during classification Phase 2.

During classification Phase 2, the Fourier descriptor of the target shape is computed and compared using cosine similarity with the candidate shapes' Fourier

descriptors. The cosine similarity results in a R value (Formula 2.7) for each candidate shape which will then be stored in a vector alongside a reference to the candidate shape. This R value is squared, to gain a certain degree of separation in the values, producing the R^2 value. The list is then sorted, with the highest R^2 value signifying the most similar boundary, and displayed to the user.

3.4 Implementation

3.4.1 Overview

The proof-of-concept for the system described above is called the Shape and Image Database (SID). SID consists of a command-line C++ application used to segment and construct the tree-of-shapes (as well as associated metrics and descriptors), a MySQL database to store and retrieve the segmented objects along with their metrics, and a Java Swing desktop application to provide a graphical user interface (GUI) to explore the segmented images and view the classification results. The development and target platform is the Linux operating system, but the code can easily be recompiled for other operating systems as the libraries and applications are cross-platform. SID can make use of a client-server model: the database and segmentation application can be housed on a central server, and the graphical client, utilized by end-users, can reside on remote machines, as long as the server and the remote machines can communicate via a wired or wireless network, or any other type of communication channel.

3.4.2 Segmentation Application

The C++ segmentation application utilizes the Megawave Library, an open-source image processing library from The Center for Mathematical Studies and their Applications (CMLA). A module for this library, originally created by Monasse and Guichard (2000) and extended by Pan (2007) implements the FLLT algorithm.

Significant revisions were made to the library and module to improve stability, robustness, introduce features and facilitate integration into the SID system. Cimg, an open-source image processing library hosted on SourceForge.net, is used to handle basic reading, writing and other operations upon images of various file formats (typically, PNG). The open-source Alglib library by S. Bochkanov is used for the cubic spline interpolation of the boundary of shapes. The open-source FFTW library by M. Frigo is used to compute the Fourier transform of vectors of data. C++ was the language of choice due to the availability of these libraries, speed, popularity, and the author's familiarity with the language. It should be noted that all descriptors and metrics are computed and stored in the database during the construction of the tree-of-shapes for each image as described in Chapter 3.

3.4.3 Database

MySQL serves as the information storage and retrieval system for SID. A relational database management system (RDMS) such as MySQL has an appealing feature set including quick and efficient storage and retrieval of information and APIs for Java and C++ to access the data programmatically. By utilizing a database rather than operating directly on files, the C++ and Java applications remain reasonably platform independent, the amount of new code is reduced (code complexity is transferred to the database vendor), and the design allows for rapid indexing of stored information for quick retrieval using SQL commands. MySQL was chosen since it is comparable to commercial database software and is free. To port SID to another database vendor, the C++ and Java database interaction sections would require reimplementation using the new database vendor's API.

3.4.4 Graphical Shape Explorer

Java is used to implement the graphical front-end for SID (Figure 3.2) to display the results of segmentation and to view the results of comparisons between shapes.

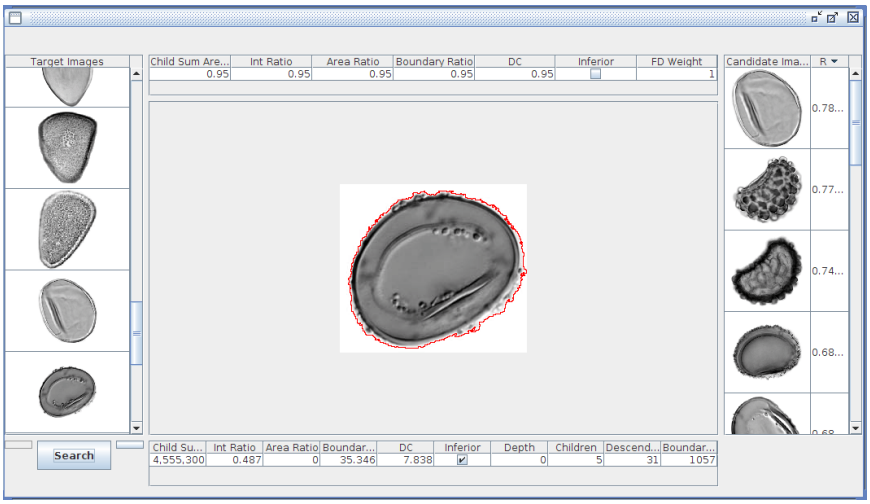


Figure 3.2: Screenshot of SID's shape explorer GUI

Java was chosen due to the author's familiarity with developing GUI applications in the language and its cross-platform capabilities. LITUTK, a general purpose Java Swing library by The Laboratory for Information Technology at The University of Tennessee, was used to provide basic functionality for the GUI (such as a desktop, wrappers for the downloading of files, and resizing of images). The viewer is client-side; therefore, the intense computational demands required by the segmentation and comparison application are removed. Therefore, there is no need to have the application execute natively on the platform and the platform invariant Java VM can be utilized to provide a consistent GUI across any platform. More importantly, Java Swing is often used in Rapid Application Development (RAD) methodologies for GUI related tasks to reduce the amount of time required to develop the application. Cosine similarity is performed on the descriptors by this application after retrieving the candidate images.

Chapter 4

Experiment and Results

4.1 Chapter Overview

The goal of this experiment is to explore design options and provide recommendations for an object recognition system. Objects are defined as regions in images identified by their boundaries and interiors, including objects contained within these interiors (Section 2.2). They can be represented as nodes in a tree arranged by their geometric inclusion (Section 2.3.7). The regions and the trees are determined using the FLLT algorithm on the raw images (Section 3.3).

Object identification is attempted by comparing the features (metrics) of a target object to those of previously identified objects stored in a database. A two-phase process is developed to exploit the indexing capabilities of modern databases. The first phase is performed as an SQL query on the database. The results are references to candidate objects that have indexed features that are real-valued that are between a set of specified upper and lower bounds (Section 3.3.2). The features are indexed by the database to provide a logarithmic bound on complexity, or retrieval time, for each search. This first phase is intended to exclude the majority of significantly dissimilar objects in the database from further consideration.

The second phase is more computationally demanding (Section 3.3.3). It compares each candidate object returned during the first phase to the target object (object to be identified). A comparison of one or more features can be used, and each comparison yields a non-negative real number. A larger returned value implies the target and candidate objects are more similar than the target and candidate objects that produce smaller values, with respect to that specific feature. Computed similarity values using different features of a candidate object can be combined using, for example, a weighted linear combination. The resulting combined similarity measure can then be used as sorting criteria for candidate objects by quantitatively determined similarity to the target object, producing a ranked list of similar candidates. The second phase requires a linear search, or evaluation of candidates, unless a similarity measure can be precomputed, indexed, or bounded in a manner that excludes some candidates from further consideration.

A design is explained and evaluated in this report that utilizes the features in the list below for phase one. These attributes are selected in a manner that allows them to be reasonably invariant to image scaling, translation, rotation and global contrast and intensity changes (invariance is approximate because of discretization effects). Surface area change ratio was omitted since the evaluation considers only the smallest shape that completely captures the pollen-grain, therefore the surface area ratio is undefined.

- Cumulative child surface area ratio
- Exterior / interior intensity change ratio
- Boundary / surface area ratio
- Average scaled radii length

The features selected for use by the second phase are the shape of the object's boundary and the texture of the interior of the object. The boundary feature descriptor approach is fully implemented and evaluated in software. The texture

region feature approach is provided as a point of reference for future improvements. Both boundary and texture features should be implemented in a manner that causes the computed similarity measures to be invariant to scaling, translation, rotation, global contrast and intensity changes, as in phase one; otherwise, the flexibility added by incorporating this requirement into previous stages is lost. There are many possible design choices for the algorithms and data structures to be used for boundary and texture similarity. Sections 4.5 and 4.6 provide results and discussion of experiments that were performed in order to make informed choices for parameters to the image classification system.

As a preliminary test, this software is used to process a database consisting of images of pollen grains from different species that correspond to the pollen morphological groups in the list below. Each image is transformed to produce five different images, corresponding to a random scaling, translation, rotation, contrast or intensity change. This results in a set of 120 images from the original 24 (not including the original images). If the features and boundary similarity measures are invariant to these transformations, then a search for matches to each image in the database should yield four matching candidates (not including a self-match), and the similarity measures for each pair of transformed images arising from the common ancestor should all be equal (or nearly so, because of pixelation and non-one-to-one mapping as described by Gonzalez and Woods (2007)) and larger relative to that of any other candidate for that target object. The results of this test, where the selected objects are the entire pollen grains in the image, are presented and discussed in Section 4.7.

- inaperturate
- monoletes
- periporates
- triletes

Section 4.8 documents the experimental design and results for a large evaluation of the implemented design. Pollen images representative of the pollen morphological groups of the list above are selected to form a database of 24 images. Each group is represented by 3 pollen species, and there are two different images of pollen grains for each species. The objective of the design is to develop software capable of classifying pollen grain images into the correct pollen morphological group. A classification to the species level is highly desirable, but is considered a separate problem as morphology does not necessarily follow modern taxonomic hierarchy. Since SID is designed to be used by a knowledgeable specialist, identification may be accomplished by at least ranking a specimen of the correct group within the top ranked K (displayed) matching candidates to a target object.

Section 3.4 describes the Shape and Image Database (SID) software and ancillary software required to evaluate the two phase process, where only a boundary similarity measure is used in the 2nd phase is given. The implemented design does not limit the users to selection of a top-level object in the tree of shapes description of an image. Any object may be utilized, within the limitations imposed upon the smallest objects (in number of enclosed pixels) by image resolution. Thus, a user may select image features (objects) that are likely to uniquely identify a pollen grain and search for matching objects from the reference collection. This process is illustrated using examples in Section 4.4.

It is recognized that use of only the boundary of objects in the second phase is an extreme limitation. It can be expected that information derived from the interior of a target object will provide significant improvements. Section 5.1.4 develops a method for extraction of a feature vector that is invariant to the required transformations and represent information about the texture of an object's interior. It is believed that this can provide a basis for future work and continued improvement of the Shape and Image Database.

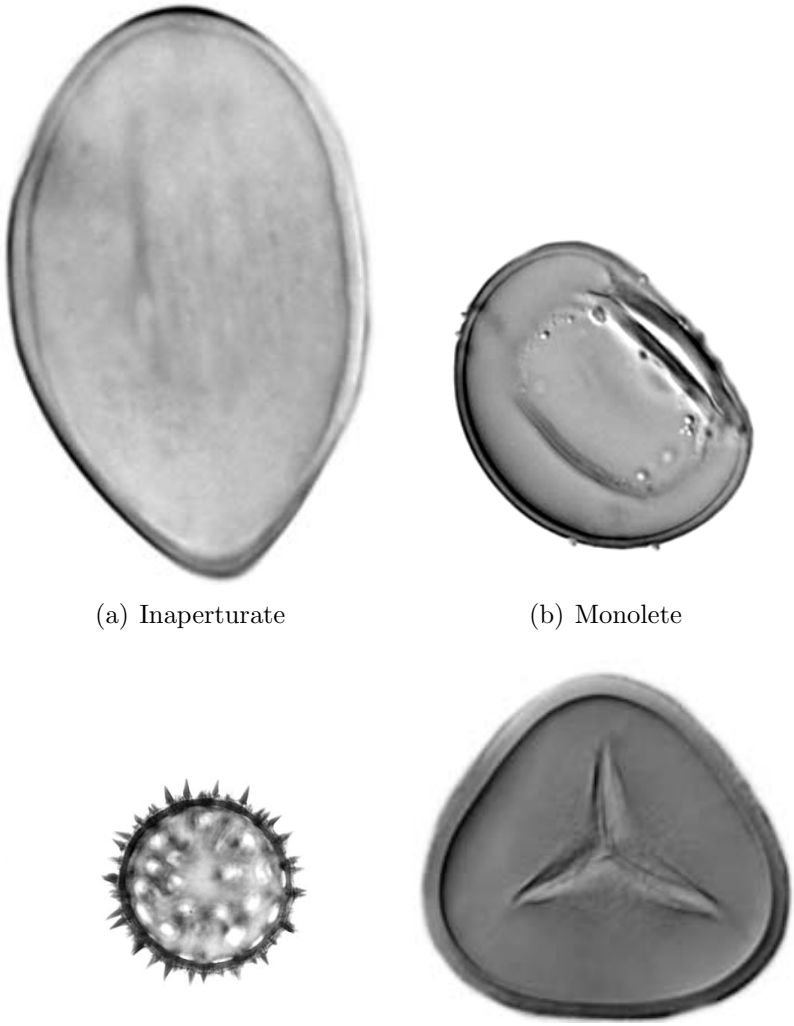
4.2 Test Data Set

4.2.1 Morphological Groups

Pollen grain images, collected and imaged by Willard et al. (2004) for an atlas of pollen species in Florida, were used in the experiment to determine the accuracy and performance of the design options for the proof-of-concept software. The collection was selected due to their well-prepared and well-photographed imagery of individual pollen grains. The paper also classified the pollen into groups with distinct physiological features. Six pollen grains from four morphological groups were selected for a total of 24 images. The morphological groups include inaperturate, monoletes, periporates and triletes. Kapp et al. (2000) describe inaperturates (Figure 4.1(a)) as a pollen grain lacking apertures, monoletes (Figure 4.1(b)) as having a long single "scar" on their surface, periporates (Figure 4.1(c)) as having many evenly distributed pores and triletes (Figure 4.1(d)) as having a tri-radiate "scar" on the surface.

4.2.2 Affine Transformations

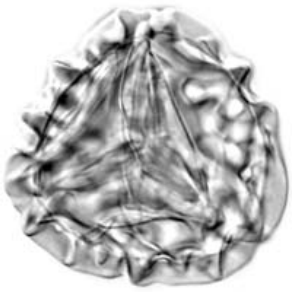
An objective of the this image classification system is invariance to global changes in intensity, contrast, scale, rotation and translation, as described in Section 2.3.1. The different pollen grains in Figure 4.1 obviously experience such variations between images, but since they are different pollen grains, the problem becomes more complex. To test that the system's invariances hold under the standard affine transformations, each pollen grain from each of the groups in Figure 4.1 has random degrees of each of the affine transformations applied artificially using the image processing capabilities of PerlMagick (Figure 4.2).



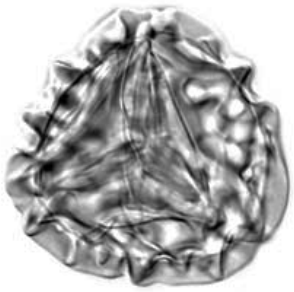
(a) Inaperturate (b) Monolete

(c) Periporate (d) Trilete

Figure 4.1: Sample pollen grains from each morpholical group used in the experiment. Pollen grains collected and imaged by Willard et al. (2004)

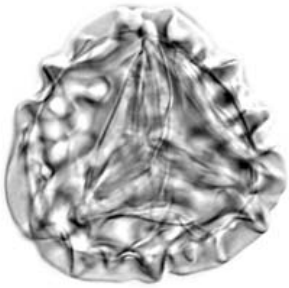


(a) Original Pteris vittata

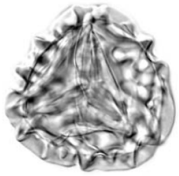


(b) Pteris vittata with a random global contrast change

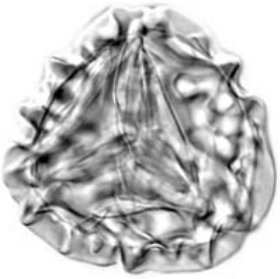
(c) Pteris vittata with a random global intensity change



(d) Pteris vittata with a random rotational change



(e) Pteris vittata with a random scale change



(f) Pteris vittata with a random translation change

Figure 4.2: Pteris vittata with random affine transformations applied using ImageMagick Studio's PerlMagick. Pollen grains collected and imaged by Willard et al. (2004)

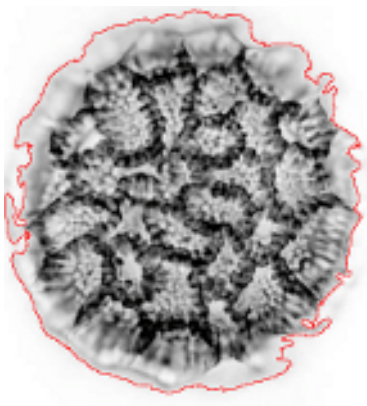
4.3 Segmentation

4.3.1 Methodology

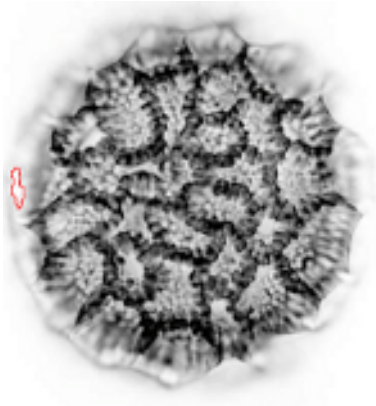
As described in Section 2.2.1, image segmentation is the first step in this image classification procedure. A successful decomposition of an image into objects of significance is vital for the proceeding steps of the classifier. The procedure encompasses both the discovery of level-sets that form the objects and the construction of the tree-of-shapes, as conducted by the FLLT Algorithm (Section 3.3.1). A visual inspection and discussion of the results of the segmentation process are given below.

4.3.2 Results and Discussion

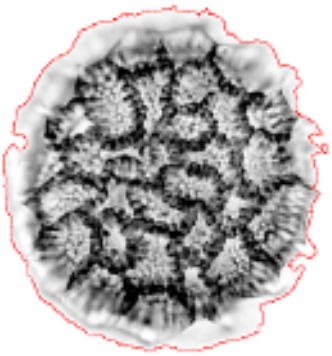
The segmentation procedure correctly captured the outer boundary of all pollen-grains in the images, save for two. In the two failed cases, the pollen grain was situated too close to the image frame and thus did not form a tree structure with the pollen grain's outline in the image as the parent shape, as shown in Figure 4.3. This effect was successfully corrected by placing the pollen grain image in a larger frame such that the border of the pollen grain does not approach the boundary of the image. In several morphological groups, the FLLT segmentation procedure successfully captured significant features of the pollen grains (Figure 4.4). These significant features include identifying features for particular morphological groups. A trilete's scar was captured as an object in the tree of shapes in Figure 4.4(d). Individual barbs of a periporate were captured as independent objects on the surface of the pollen grain in Figure 4.4(c). The smooth surface texture of an inaperature was captured in Figure 4.4(b). Finally, the distinguishing scar of a monolete was isolated as shown in Figure 4.4(a). The individual barbs and surface features of the periporates and scars of monoletes were typically captured. In certain cases, the edges of the boundaries of significant features were distorted due to the depth of field blur incurred by the microscope, thus possibly causing classification problems later on due



(a) The majority of the pollen grain's boundary captured by the FLLT

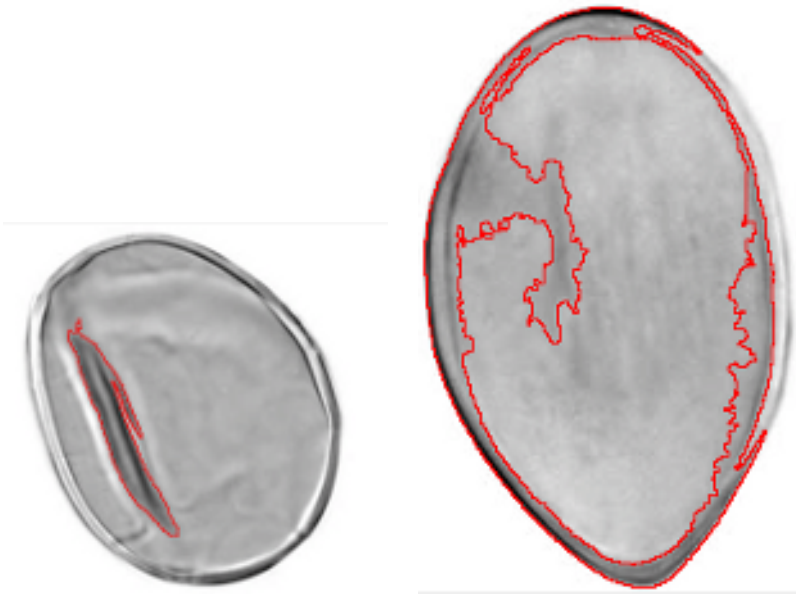


(b) A portion of the pollen grain in Figure 4.3(a) that failed to be included



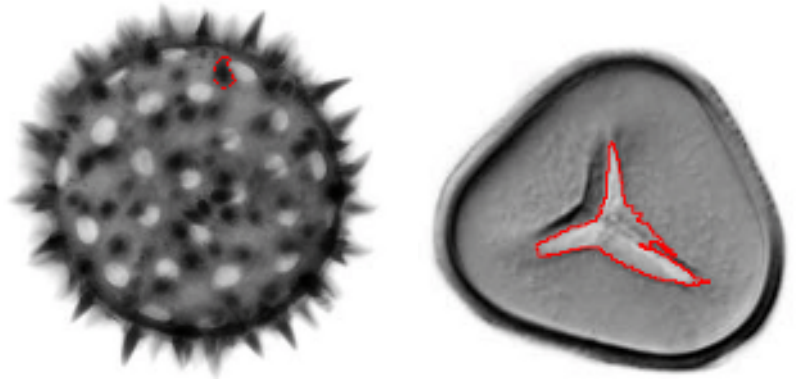
(c) Complete pollen grain captured after reframing

Figure 4.3: A "shattering" effect of a pollen grain's FLLT segmentation when situated near the image's boundary. Pollen grains collected and imaged by Willard et al. (2004)



(a) The distinguishing scar of a monolete captured by the FLLT

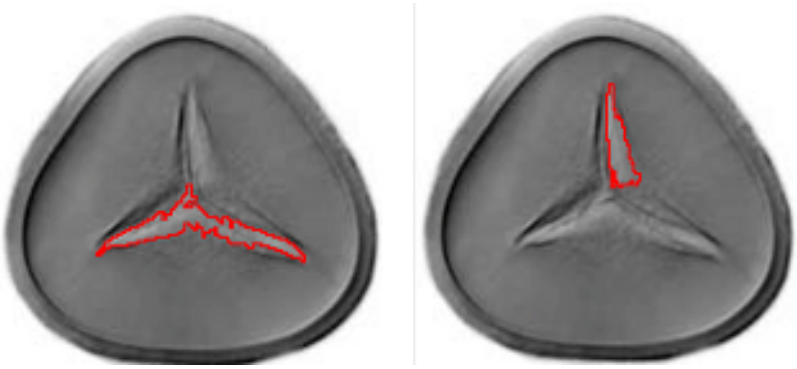
(b) The smooth surface of a inaperature captured by the FLLT



(c) A spike on the surface of a periporate captured by the FLLT

(d) The distinguishing tri-radiate scar of a trilete captured by the FLLT

Figure 4.4: Capturing of significant features as independent objects in pollen grains using the FLLT. Pollen grains collected and imaged by Willard et al. (2004)



(a) Partial capturing of a distinguishing tri-radiate scar of a trilete by the FLLT

(b) Partial capturing of a distinguishing tri-radiate scar of a trilete by the FLLT

Figure 4.5: Failure of the FLLT to capture a significant pollen grain feature as a single object. Pollen grains collected and imaged by Willard et al. (2004)

to the distinguishing features being in independent objects, causing distortions of descriptors and metrics. Blurred boundaries between the surface of the pollen grain and a significant feature on that surface can impact the isolation of said significant features into the same region in the tree-of-shapes, as shown in Figure 4.5. The same distortions of insignificant areas being in focus and significant features being out-of-focus also caused the FLLT to capture insignificant crescent shaped regions (Figure 4.6) or superfluous areas between features of interest (Figure 4.7) at the expense of fully capturing distinguishing features such as the triangular slit adjacent to the red contour in Figure 4.7. The boundary of the pollen grain itself can contain significant information. A "spiky" boundary might indicate a periporate or a smooth boundary might indicate an inaperature. If the microscope image has this outer boundary out of focus, a spiky boundary can appear smooth and vice-versa (if certain surface features brought into focus are rough), as shown in Figure 4.8. This observation can be quantized by using the DC component of the Fourier transform of the boundary's centroid distance as described in Section 2.3.4. The DC component is the first element of the Fourier transform and in the context of a centroid distance descriptor, it is the average distance of the normalized (scale invariant) radii from the center of a shape to

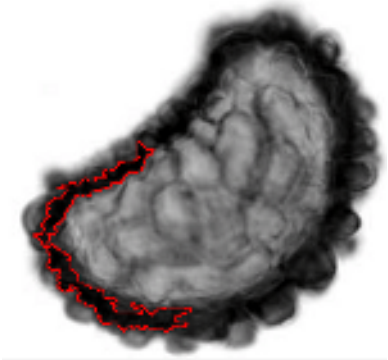


Figure 4.6: Superfluous crescent segmentation of a pollen grain region by the FLLT. Pollen grain collected and imaged by Willard et al. (2004)

Figure 4.7: Superfluous interior segmentation of a pollen grain by the FLLT. Pollen grains collected and imaged by Willard et al. (2004)

points along its border. Elementary geometry states that the average radii of a circle is longer than that of other shapes of the same boundary length due to a circle having a larger surface area than shapes of equivalent boundary size. It stands to reason that smoother, circular shapes will have larger DC components than other, spiky shapes. The experiment reflects this intuition as the DC components of the triletes segmented with spiky boundaries (Figure 4.8(c) and Figure 4.8(d)) are roughly 30% lower than the smooth trilete boundaries (Figure 4.8(a) and Figure 4.8(d)). The human observer, without the aid of a red outline of the contour, would likely consider the shapes of Figure 4.8 as having the same boundary as the human eye and mind can better interpret the outline of a shape that might be somewhat out of focus.

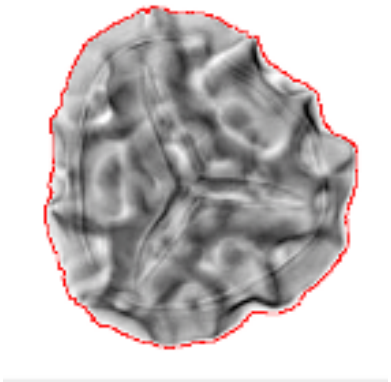
4.4 Internal Object Exploration

4.4.1 Overview

An important benefit of tree-of-shape segmentation is the isolation of distinguishing features into subobjects that can be compared individually to each other. The subobjects can include the "spiky" edges of a periporate, the tri-scar of the triletes or the single scar of the monoletes. These comparisons can be used by themselves to discover objects with similar internal features or they can be used in conjunction with other metrics to measure similarity of the overall shape. In this section, one shows these features captured and compared by the proof-of-concept software.

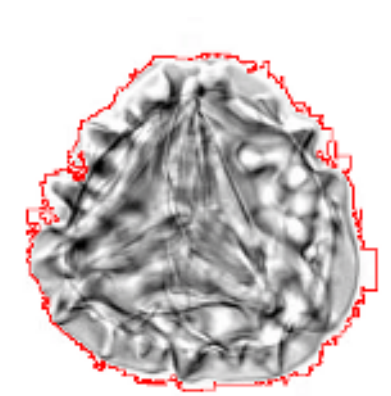
4.4.2 Results and Discussion

As demonstrated in Section 4.3, a number of distinguishing features were captured. For periporates, the rough surface features that distinguish them are of the most interest. The spines captured on the surface of a periporate are shown in Figure 4.9, with a red outline of that feature against the original image. The three spines were among the top matches for one another with an R^2 value of around .7. These

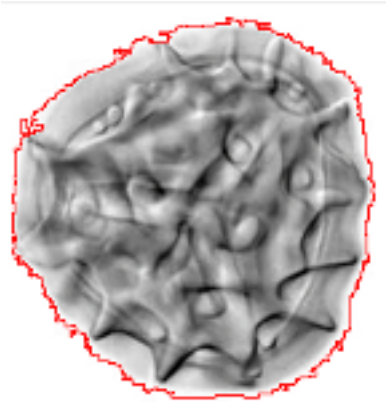


(a) Boundary of a trilete captured as smooth (DC 9.061)

(b) Boundary of a trilete captured as smooth (DC 10.092)



(c) Boundary of a trilete captured as spiky (DC 6.726)



(d) Boundary of a trilete captured as spiky (DC 7.777)

Figure 4.8: Boundaries of pollen grains changing with differing focus depth when segmented using the FLLT. Pollen grains collected and imaged by Willard et al. (2004)

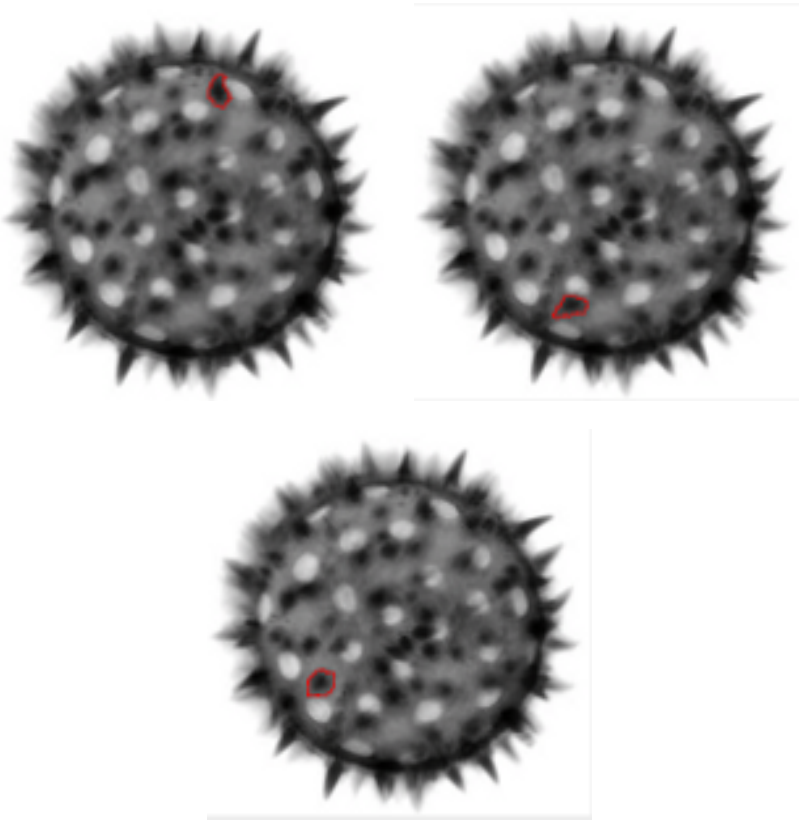


Figure 4.9: Capturing of a spine on the surface of a periporate by the FLLT. Pollen grains collected and imaged by Willard et al. (2004)

surface features were identified in both members of the same species of periporate, with equivalent features, among the top matches, shown in Figure 4.10. Monoletes have a distinguishing, single, long scar that is typically visible on their surfaces, as noted in Section 4.2. This feature was captured in some images, as shown in Section 4.3. In Figure 4.11, the scar is matched to its equivalent scar in another monolete with an R^2 value over .8, among the top matches. Another top match arose as a trilete, which has a scar similar to that of a monolete to the human observer due to the position in which it was photographed. Triletes have a distinguishing, tri-radiate scar that is typically visible on their surfaces, as noted in Section 4.2. Unfortunately, for the majority of the cases, the scar failed to be captured in its entirety due to other surface features incorporating the tri-radiate scar, as shown in Section 4.3. However,

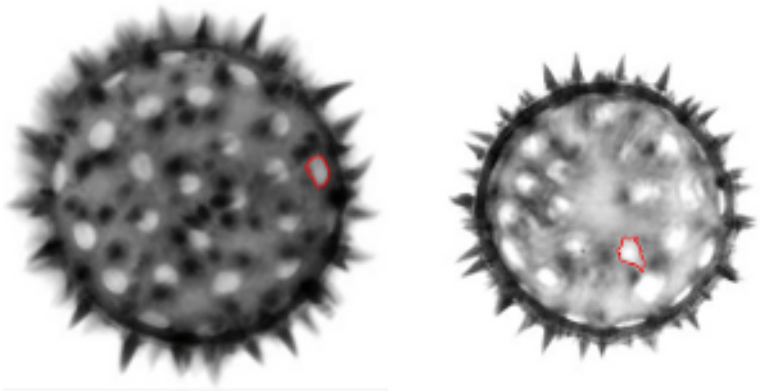


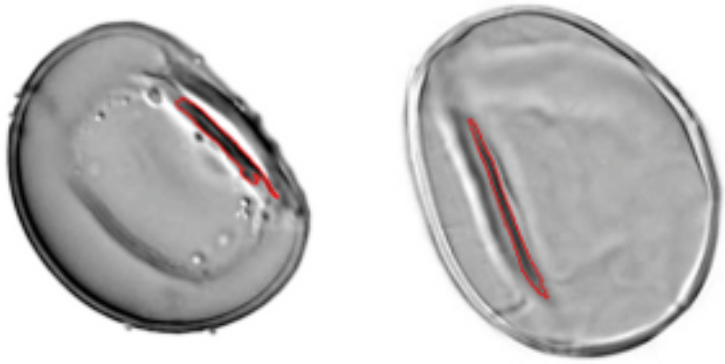
Figure 4.10: Successful matching of periporate surface features using the image classification system. Pollen grains collected and imaged by Willard et al. (2004)

the triangular outer shape of several pollen grains in this group was useful in aiding in the classification of triletes, as demonstrated in Section 4.8. Periporates, having a smooth surface, lack distinguishing features that can be captured as subojects. However, the texture of the region captured can be used to provide descriptors for a region-based classifier, as described in Section 5.1.

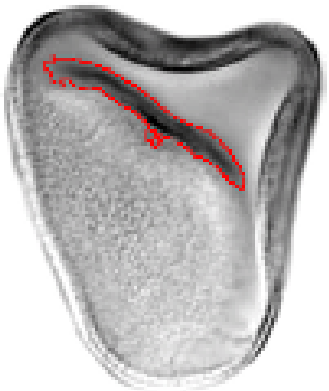
4.5 Phase 1 Descriptors

4.5.1 Methodology

There are a number of different metrics available that can be quickly used to omit significantly different shapes from consideration, as described in Section 3.3.2. In this portion of the experiment, possible constraint values to use in Phase 1 are evaluated. By limiting the values to a certain +/- percentage difference from the target value being classified, one can eliminate significantly different shapes from consideration. If the constraints are too loose, then a large number of non-matching shapes will result and be considered in Phase 2, thus decreasing performance. If the constraints are too tight, then potentially matching shapes will be excluded from consideration in Phase 2.



(a) A scar captured in a monolete (b) A scar matched in a monolete



(c) Incorrect match with a monolete-like scar in a Trilete

Figure 4.11: Successful matching of monolete surface features using the image classification system. Pollen grains collected and imaged by Willard et al. (2004)

To determine ranges for each constraint, one first normalizes and mean-centers the values of the raw TOS metrics. Since the TOS metric values will be determined by a +/- percentage of the target's metrics, the data can be best visualized after normalization and as a distance from the mean (which is zero after mean-centering). While the raw values for each morphological group can be orders of magnitude different from each other, one still needs common constraints for Phase 1. The common ranges are the +/- percentages of a target's metrics discussed above, where the raw number is multiplied by fixed percentages to yield an acceptable range of values of the candidate shapes to be returned. The experiment in this chapter makes use of the top-most shape of the pollen grain in the tree-of-shapes (the immediate child of the original image), therefore the area ratio metric is excluded from consideration and the evaluation of its worthiness is left to future work.

4.5.2 Results and Discussion

Boundary Size Ratio

As shown in Figure 4.12, the boundary size ratio values for each group fluctuate between -0.2 and 0.65. All of the morphological groups are below 0.4, save for periporates. A few of which have pronounced spiky edges significantly different from the other members of the same morphological group, which cause smaller relative surface areas for the size of their boundary. This is due to it being further from a circle, which calculus states has the most surface area for the smallest boundary size. The sizable difference between the maximum ratio and the minimum ratio in most groups is accounted for by the focus of the boundary segmenting the edge as spiky in some cases rather than smooth, as demonstrated in Section 4.3. An additional 20% of slack is added to account for any additional noise or variation in segmentation, therefore one considers a range of -0.25 to 0.8 about the average value or from .25 to 1.3 if shifted into the positive range. To reach the maximum value from the minimum value and vice versa, the constraint percentage will be +/- 5.2x the original value.

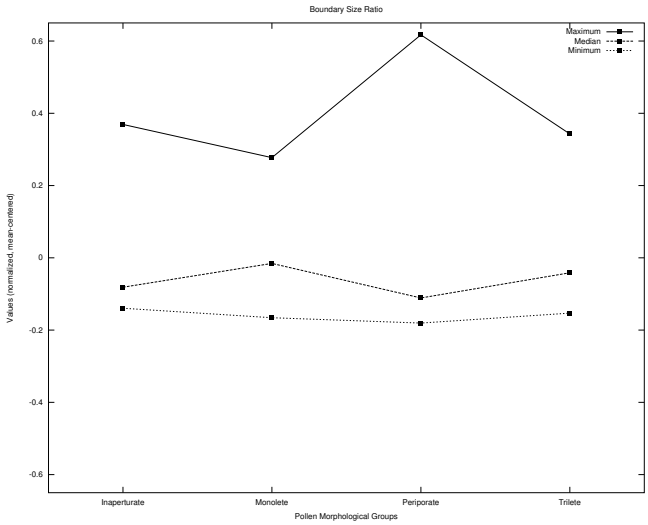


Figure 4.12: Value range for each morphological group's boundary size ratio Phase 1 metric (normalized, mean-centered)

Cumulative Child Area Ratio

As shown in Figure 4.13, the cumulative child area ratio value fluctuates between 0.2 and -0.2. An additional 20% of slack is added to account for any additional noise or variation in segmentation, therefore one considers a range of -0.25 to 0.25 about the average value, 0.25 to 0.75 in the positive range. The resulting constraint will be +/- 3x the original value.

Average Boundary Radii (DC)

As shown in Figure 4.14, the cumulative child area ratio value fluctuates between 0.1 and -0.1. There is a sizable difference between the minimum and maximum values in the triletes that is not as pronounced in the other morphological groups. This is due to the focus level on the boundary of the shape, producing a spiky edge on an otherwise smooth edge during segmentation, as demonstrated in Section 4.3. An additional 20% of slack is added to account for any additional noise or variation in segmentation, therefore one considers a range of -0.12 to 0.12 about the average

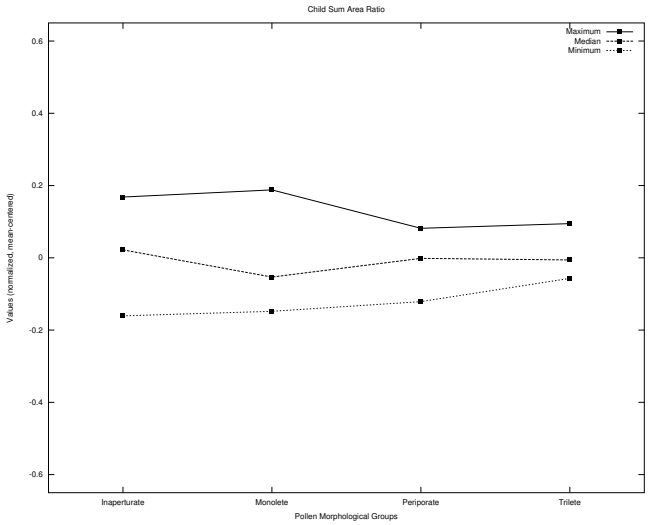


Figure 4.13: Value range for each morphological group's cumulative child area ratio Phase 1 metric (normalized, mean-centered)

value, 0.12 to 0.36 in the positive range. The resulting constraint will be +/- 3x the original value.

Intensity Ratio

As shown in Figure 4.15, the cumulative child area ratio value fluctuates between -0.05 and 0.1. The minimum values and maximum values are closely aligned with one another, as is expected since all images are against a white background with only the highest level shape considered. In tests with different backgrounds, this metric will likely not be useful for the top-most shape of a pollen-grain as it only provides useful information in a consistent manner for shapes in the interior of pollen grains (since it relies on intensity differences between the interior and exterior portions of a shape's boundary). Further research is necessary to determine the impact of different backgrounds and the use of this metric on top-level shapes as well as the effectiveness of it on interior shapes. An additional 20% of slack is added to account for any additional noise or variation in segmentation, therefore one considers a range of -0.06

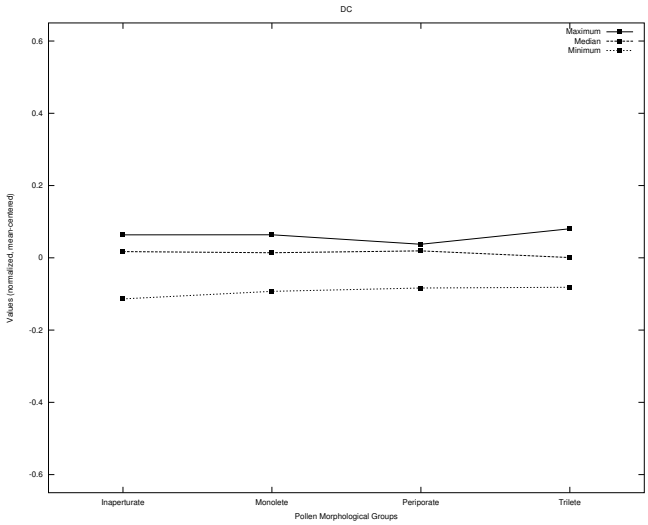


Figure 4.14: Value range for each morphological group's average boundary radii (DC) Phase 1 metric (normalized, mean-centered)

to 0.12 about the average value, 0.06 to 0.24 in the positive range. The resulting constraint will be +/- 4x the original value.

4.6 Phase 2 Descriptors

4.6.1 Methodology

The Phase 2 descriptors are derived using the Fourier transform of the centroid distance boundary signature, as described in Section 2.3.3. The interpretation of the values of this vector can be based on observations of the Fourier transforms of other signals, as described by Gonzalez and Woods (2007). The first element is the DC component, which for a centroid distance signature of the boundary represents the normalized average length of the radii. The DC component, being a single real-valued general descriptor of the boundary of the shape, is considered as a metric in Phase 1. As noted in sections above, calculus teaches us that for shapes with the same boundary length, this value is maximized as the shape approaches a perfect

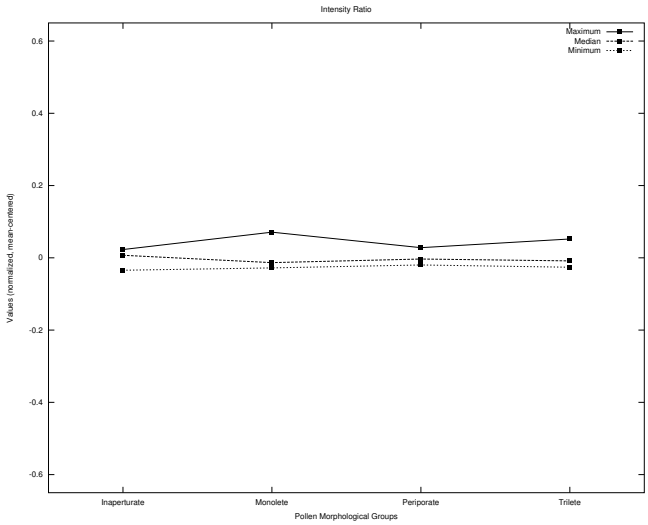


Figure 4.15: Value range for each morphological group's intensity ratio Phase 1 metric (normalized, mean-centered)

circle. The second-half of the values of the fast Fourier transform are a mirror image of the first-half; thus, they can be excluded.

The lower elements (frequencies) represent areas of no change or slow change in the radii, while the higher elements represent areas of rapid change in the radii. There is the possibility of localized improper segmentation, producing a boundary that can contain noise. As visualized in Figure 4.8, this noise can take the form of a highly irregular boundary. By dropping some of these higher dimensions from the Fourier descriptor, performance can be improved (less data to process by the classifier) and noise can be reduced. This is, in effect, a low-pass filter that will result in the smoothing of the boundary, as described by Gonzalez and Woods (2007).

There also remains the question of the minimum number of sample points along the border that are required to gain an accurate picture of the shape of the boundary. The sampling serves to place the descriptor into a fixed number of points, since the boundaries vary in length, for future comparison by the classifier, which requires vectors of the same length. In addition, the overall shape of the boundary needs to be preserved, while reducing dimensions, thus decreasing computational complexity.

This portion of the experiment attempts to determine an optimal number of sample points of the boundary's radii and the optimal number of dimensions to keep from the FFT of the sampled radii vector. To explore this, many combinations of sample size and Fourier descriptor dimension lengths were evaluated.

4.6.2 Results and Discussion

Sample Points

To evaluate the number of sample points that should be used to generate the Fourier descriptor, the 24 test images were evaluated using powers of 2 from 8 to 512 for the number of sample points. A number larger than 512 was not considered given the size (in pixels) of the images in the test set. The first 1/4th of the vector was retained, which is a common low-pass filtering technique in the frequency domain to smooth a signal. The number retained is few enough to prevent noise from having a significant impact and enough to ensure that an accurate description of the boundary is preserved. The results in Figure 4.16 show that 256 sample points appear to be ideal, as the accuracy decreases after that number as redundant noise is compounded. Further examination of images of different sizes is required to determine if this number remains optimal for significantly larger images than those in the test set, as the sparsity of the sampling will increase as an image grows in size. If one visualizes the results of the sampling upon the boundary, one sees that the edges of shapes become significantly different from the original shape as one samples fewer points using cubic spline interpolation. One must balance computational performance (fewer dimensions that require an FFT) with retaining as much useful boundary information as possible to create an accurate representation of the boundary in a lower dimensional space. As depicted in Figure 4.18, one can see that the features of the original boundary remain preserved at around the same number of 256 sample points shown to be effective at providing an accurate description of the boundary.

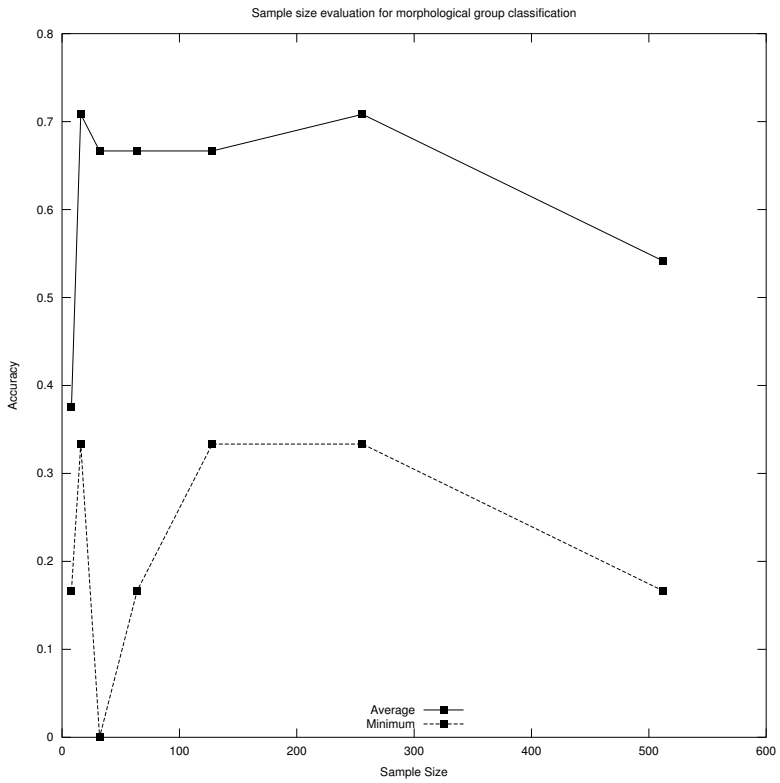


Figure 4.16: Accuracy evaluation results of varying the number of boundary sample points (1/4 of dimensions kept)

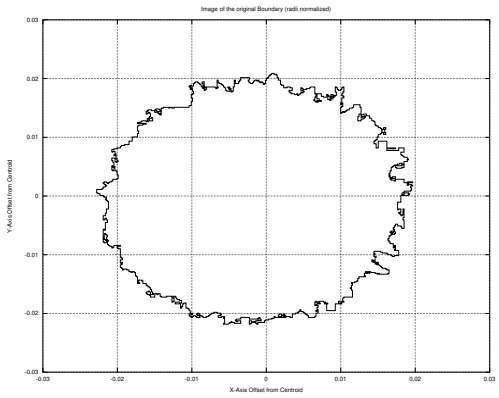
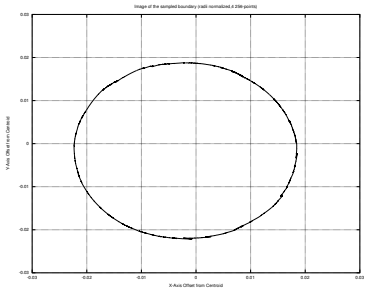
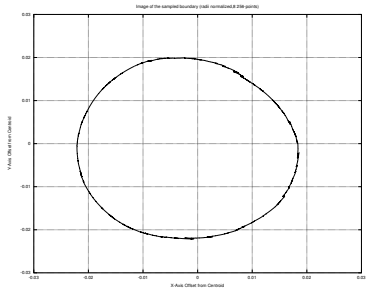


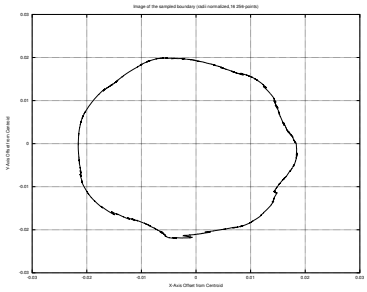
Figure 4.17: The boundary of a shape after normalizing its radii and mean-centering (centroid as the origin)



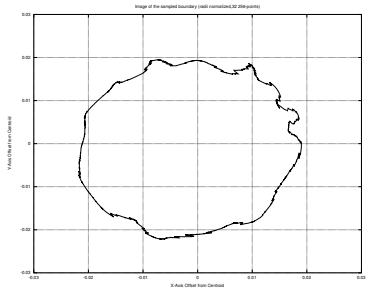
(a) 4 sample points



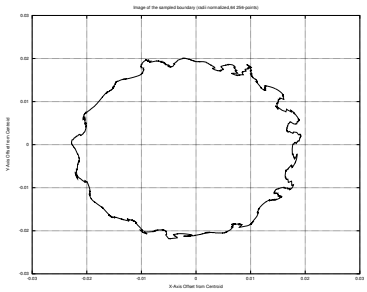
(b) 8 sample points



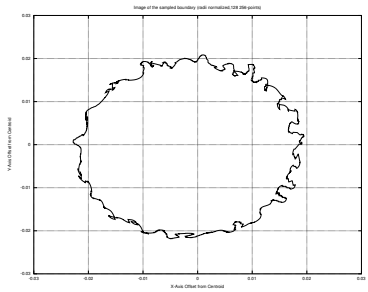
(c) 16 sample points



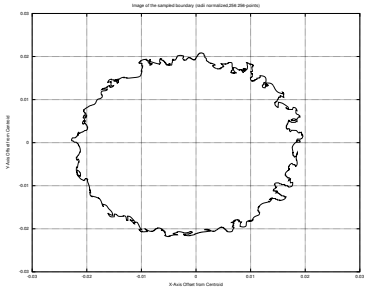
(d) 32 sample points



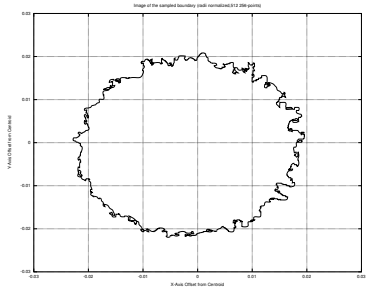
(e) 64 sample points



(f) 128 sample points



(g) 256 sample points



(h) 512 sample points

Figure 4.18: Results of a cubic spline sampling of the boundary in Figure 4.17 with varying sample sizes

Dimensionality Reduction

Having determined a satisfactory sample size, one begins to examine how many dimensions one should keep from the Fourier descriptor. To do this, the same set of test images was used with a 256 point sample size, and all possible reduced dimensions for multiples of 2 from 2 to 128 dimensions were kept. As stated above, the higher frequencies likely represent noise from improper segmentation due to blurring or other distortions, with lower frequencies capturing the areas of slow change and points along the spikes of a pollen grain (if present). As shown in Figure 4.19, after around the 32nd dimension, the benefits of adding higher frequencies vanishes and somewhat further beyond, the average accuracy begins to decline due to the noise incurred by inclusion of higher frequencies. By lowering the dimensions kept to 32, one can not only reduce the impact of this noise, but also improves the performance as the vectors being compared are significantly smaller. The effectiveness is expected to remain if the number of sample points is increased, although, further experimentation would be necessary verify this. To visualize the effects of Fourier descriptor dimensionality reduction, a boundary was plotted in Figure 4.20 with various numbers of dimensions retained. As one can see, the low-pass filter smooths the edges of the sampled boundary as one decreases the number of higher dimensions retained. With 128-dimensions retained, one can see that the boundary resembles the original (including much of the noise) image in Figure 4.17. As one decreases the dimensions retained, one can see that the shape is smoothed to the point that it becomes a circle when only the DC component is used. The dimensionality retention determined above, 32, visually appears to capture the overall shape without significant amounts of noise.

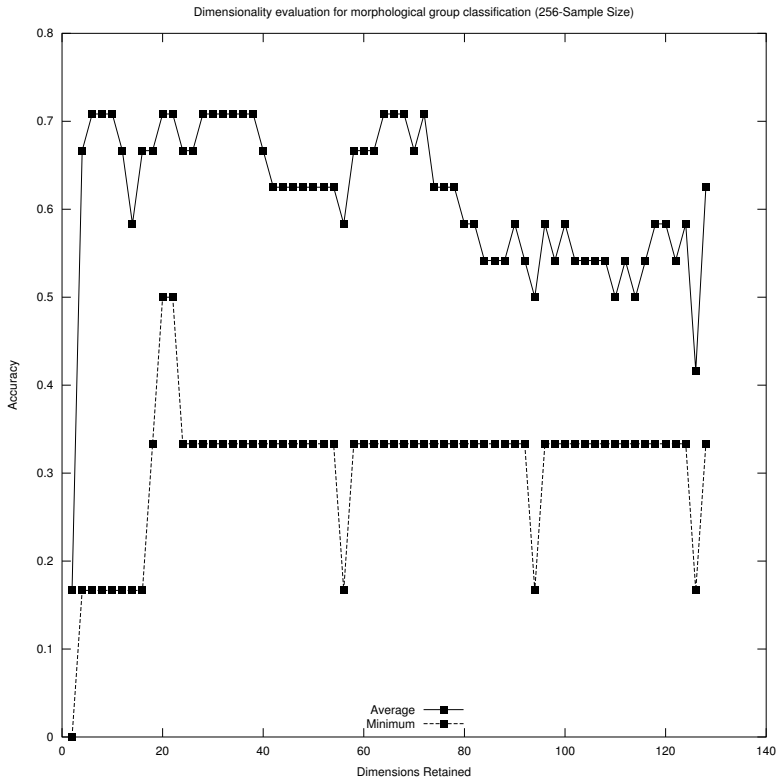
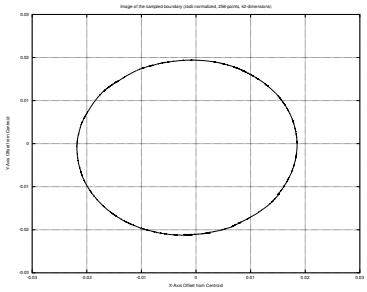
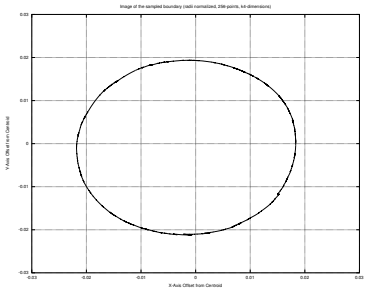


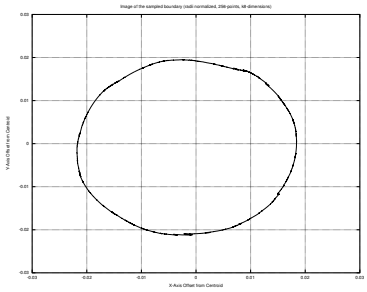
Figure 4.19: Accuracy evaluation results of varying the number of retained dimensions from a boundary's Fourier descriptor (256 sample points)



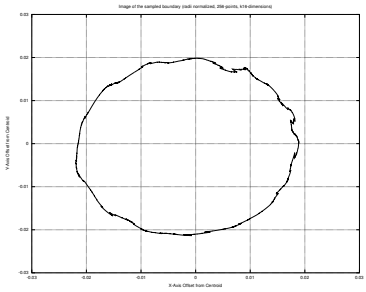
(a) 2 dimensions



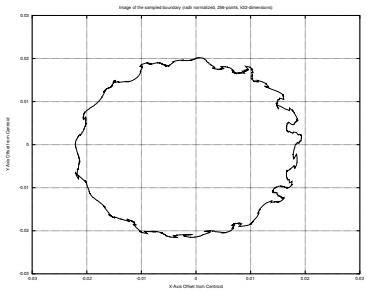
(b) 4 dimensions



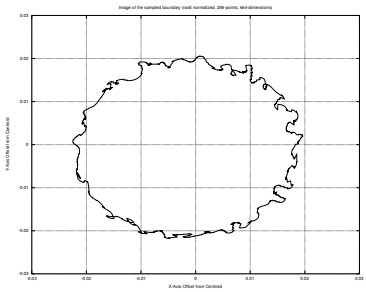
(c) 8 dimensions



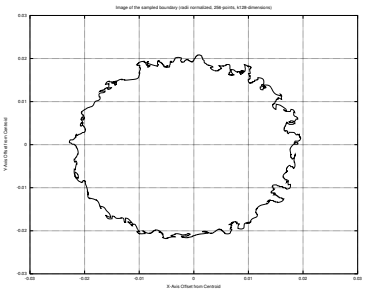
(d) 16 dimensions



(e) 32 dimensions



(f) 64 dimensions



(g) 128 dimensions

Figure 4.20: Results of a 256-point cubic spline sampling of the boundary in Figure 4.17 with various Fourier descriptor dimensions retained

4.7 Affine Transformation Invariance

4.7.1 Methodology

The first test of the performance of the image classification proof-of-concept software will be to determine if the centroid distance Fourier descriptors of Phase 2 are invariant to the affine transformations described in Section 2.3.1. Each pollen grain in the set is transformed using each type (scale, rotation, translation, global contrast and intensity change) by a Perl script with PerlMagick's image processing library. This results in 144 pictures (counting the original) from the original 24. If the system is invariant to these changes, then each of the related images will have a R^2 similarity value greater than .95 (approaching 1) with the original image, and other images will rank significantly less. Since the transformations are artificial, significant noise is introduced if the data is interpolated / extrapolated for non-one-to-one mappings, which is most prominent with scaling. Experimental verification of Phase 1 descriptors are not considered and will be evaluated in detail in future work, as the constraints determined in Section 4.5 are considered sufficiently generous.

4.7.2 Results and Discussion

Summary

A summary of the accuracy for the centroid distance Fourier descriptors amongst the different categories of transformations is shown in Figure 4.21, with each bar representing a transformation and its height showing how many have R^2 similarity values approaching 1 with their original image. In general, image transformations that resulted in a one-to-one mapping of the original image to the transformation produced results consistent with the system being invariant to the transformations mentioned above. Once interpolation / extrapolation of non-one-to-one mappings took effect in select transformations, the performance suffers, due to the noise incurred by the interpolated / extrapolated data causing the FLLT to segment the (sometimes

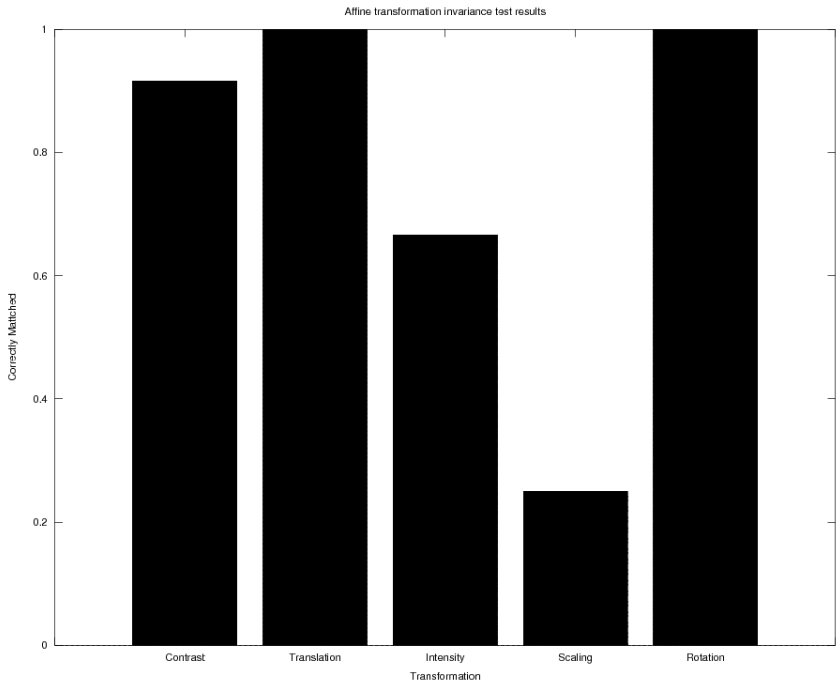
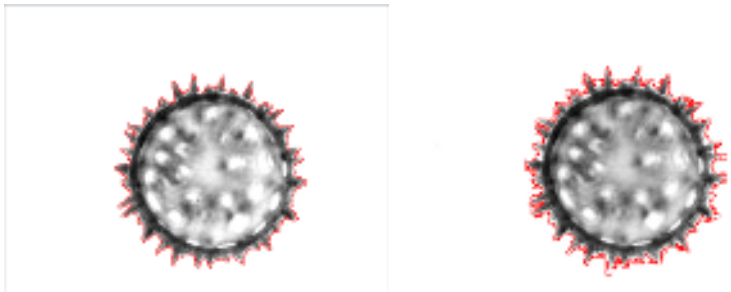


Figure 4.21: Percentage of transformed images that (near) perfectly match (R^2 greater than .95) their original images using centroid distance Fourier descriptors

blurred) boundaries differently. Separation between transformations of the original pollen grain and different pollen grains was significant, with no image derived from a different pollen grain achieving an R^2 value of greater than .95. Pollen grain image transforms of different images are nearly always lower in R^2 value than those of the same pollen grain image (with the possible exception of rotation transforms), typically in the lower .8 range or lower. Details of the results from the experiment are in Appendix B.

Rotation

The centroid distance boundary Fourier descriptors of the shape proved to be completely invariant to rotation with all rotated images approaching an R^2 value of near 1. This transformation had perfect one-to-one mapping and no interpolation or extrapolation when only random rotations of multiples of 90 °are considered. Gonzalez



(a) Boundary of an periporate (DC 4.283)

(b) Boundary of the same periporate with global intensity change (DC 5.78)

Figure 4.22: Intensity changes causing FLLT segmentation differences. Pollen grains collected and imaged by Willard et al. (2004)

and Woods (2007) demonstrate that other artificial rotations do not result in one-to-one mappings, and the frame of the image itself must be altered to accommodate the image's new dimensions.

Global Intensity

Images with global intensity changes achieved an R^2 value approaching 1 about 68% of the time. This drop in accuracy is caused by the blurred regions close to the original boundary becoming significantly different from the border and/or boundary, thus being included in a differing region during the segmentation process. In the case of increasing intensity, the blurred areas near the boundaries become more intense (thus more like the background) more quickly than the object (being dark and thus lower in intensity value) after increasing overall intensity by a scalar percentage, as shown in Figure 4.22.

Global Contrast

Images with global contrast changes achieved an R^2 similarity value close to 1 with their original image about 92% of the time. Significant changes in contrast coupled with boundaries not in perfect focus caused the FLLT algorithm to determine different

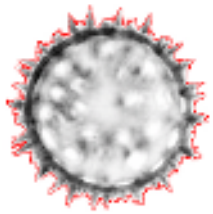


Figure 4.23: Pollen grain in Figure 4.22(a) with a global contrast change (DC 4.493) causing differences in FLLT segmentation

borders by inclusion / exclusion of pixels near the transitional border region of some, as shown in Figure 4.23.

Translation

The centroid distance boundary Fourier descriptors of the shape proved to be invariant to rotation with all translated images approaching an R^2 value of near 1. The translation technique used resulted in a one-to-one mapping and did not involve any interpolation. In translation techniques involving non-one-to-one mappings (and thus extrapolation or interpolation), the accuracy decreased drastically to between 40% to 60%. Images taken in the system should be expected to behave more like the former rather than the artificially processed images with the use of optics or physical positioning over digital transformations that tend to incur interpolation.

Scaling

Scaling affine transformations achieved a 25% accuracy rate for R^2 matches near unity. The interpolation / extrapolation effects of non-one-to-one mappings impact scaling greatly, causing the boundaries to become deformed when compared to their original images (Figure 4.24). Typically, the R^2 values are higher than those of unrelated images, but fall short of approaching an R^2 value of 1. For example, the scaled pollen grain in Figure 4.24 has an R^2 value of 0.9, and thus was excluded from consideration.

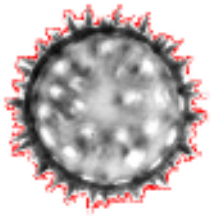


Figure 4.24: Scaling of the pollen grain in Figure 4.22(a) (DC 4.438) causing differences in FLLT segmentation

4.7.3 Conclusion

From the results, it becomes evident that spatial interpolation has a significant adverse impact on the FLLT segmentation of an image and, by extension, the classifiers that make use of descriptors derived from the boundaries of the segmented objects. Image transformations with one-to-one mappings and no interpolation did not segment differently, achieving R^2 values approaching 1 with the original image. Image transformations with limited non-one-to-one mappings, such as sinusoidal contrast changes and scalar intensity changes (more so, due to higher intensity values being multiplied by the scalar), experienced limited differences in segmentation. Transformations with significant spatial interpolation, such as scaling, suffered the most and were segmented significantly differently in many cases. Further experimentation is needed to verify that the issues of differing segmentation arise when one uses a more advanced (fair) intensity altering transform rather than scalar multiplication.

4.8 Morphological group classification

4.8.1 Methodology

The group classification test will determine if the image classification system is capable of classifying a pollen grain into its proper morphological group using both Phase 1

and Phase 2 classification. The constraints determined in Section 4.5 will be used for the TOS metrics in Phase 1 and a sample size of 256 with 32 dimensions retained will be used in Phase 2, per the experimental results in Section 4.6. Only the cosine similarity measure of the FD descriptors is considered with Phase 1 being used entirely to determine candidate shapes, future work will determine an optimal weighing of a similarity measure generated by comparing TOS metrics and the similarity measure of the Fourier descriptor. Each pollen grain image will be used as a target image to compute the R^2 values resulting from the classification procedure for all images and all shapes. Only the smallest shape that captures the overall pollen grain is used for each target, but it is compared to all of the shapes / child shapes. A pair of different pollen grains from the same species is present within each morphological group, as described in Section 4.2.

4.8.2 Results and Discussion

Summary

A member of the same morphological group was the top matched shape in 18/24 instances by R^2 value. A trained human observer is easily able to determine the correct morphological group match based on the top 3 results as a member of the proper group is present in the top 3 results of each comparison. In 19/24 cases, the majority of the top 3 matches were of the same morphological group. In 20/24 cases, the pair image of a pollen grain of the same species as the target is in the top 3, with 8/24 instances of the same species being the top match.

Monoletes

Pairs of the same species are the top match of each other in all but one case. The monoletes correctly matched to other members of its own morphological group in 5/6 of the test cases. In the one case where the top match was not of the same morphological group, inaperturate grains occupied the majority of the top 3 positions

in the ranking. In that case, the edges of the target monolete pollen grain were quite smoothed compared to the other pollen grain images due to the level of focus. This likely impacted the ability of the Fourier descriptor to properly identify the shape's outer morphological structure as the smooth exterior was quite similar to its top matches from the inaperturate group. The blurring effect on segmentation is examined in Section 4.3.

Trilete

In each trilete image, the matching species pair is in the top 3 results by R^2 value. In 4/6 of the cases, the top match is a member of the same morphological group. In 5/6 cases, the majority of the top 3 matches are in the same morphological group. The species pair that failed to have a member of the same morphological group as its top result was drastically different from other species pairs of the same morphological group to the casual human observer and they could be mistaken for Monoletes and Inapertures, as the classifier matched them. The differences between Acrostichum danaeifolium and the other triletes can be observed in Appendix A.

Inaperturate

A member of the same species is in the top 3 results of 5/6 of the inaperturate test cases. A member of the same morphological group is the top match in 3/6 of the cases with the majority of the top 3 matches being from the proper morphological group in 3/6 cases. In each of the failing cases, the inaperturates were mismatched to monoporates, which are visually similar to inaperturates, as seen in Appendix A. The single failing case of the monoletes mismatched to the inaperturate group, further indicating similarity between the two groups.

4.8.3 Periporate

Periporates did the best of all morphological groups, with members of the same morphological group occupying the top R^2 value in each test case. In addition, the top 3 matches in all cases were of the proper morphological group. The pairing image of the same species appeared in the top 3 results of the periporate test cases 5/6 of the time. The pollen grains in this group contain many spikes and barbs along their surfaces, as such their outer borders share a similar pattern with one another.

4.9 Conclusion

In this chapter, the strengths and weaknesses of the FLLT segmentation of pollen grains were demonstrated and analyzed. Tree-of-shape Phase 1 metrics were evaluated to determine optimal constraints with an emphasis on preventing members of the proper morphological groups from being excluded from consideration. Future work will focus on striking a balance between this rather conservative approach and more aggressive settings to eliminate more significantly different candidate shapes from consideration. Sample size and dimensionality retention parameters for centroid distance Fourier descriptors were evaluated and determined to be mostly invariant to affine transformations that did not involve significant spatial interpolation. Pollen morphological group matching using Phase 1 and Phase 2 classification was evaluated and found to be considerably accurate when one considers the information available to the classifier. In Chapter 5, methods are proposed to further refine and improve the implemented image classification system.

Chapter 5

Conclusion

5.1 Future Work

5.1.1 Distinguishing Feature Capture

The first challenge is improving upon the capture of distinguishing features as shapes. If a technique could be developed to merge geometrically adjacent shapes that form a region of significance as one shape, then that shape could then become a valuable descriptor of a given object in an image. For example, the triletes described by Kapp et al. (2000) have a distinguishing three-pointed scar on their surface. The image segmentation program currently captures such features as multiple shapes, as can be observed in Figure 5.1. However, a step to perform "shape linking" resembles an approach that implicit active contour models had hoped to avoid, edge linking (Section 2.3.3). A better approach would be to make the algorithm less sensitive to subtle changes in intensity or to implement a heuristic to predict adjacent shapes that should be merged during tree-of-shapes construction.

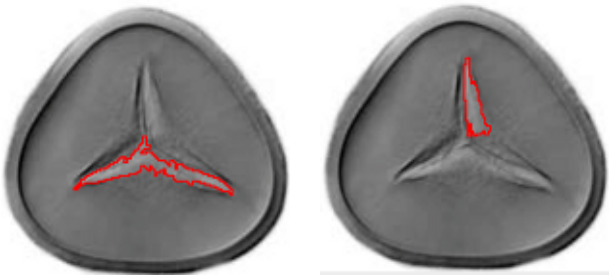


Figure 5.1: A trilete's distinguishing feature fragmented in capturing. Pollen grain imaged by Willard et al. (2004)

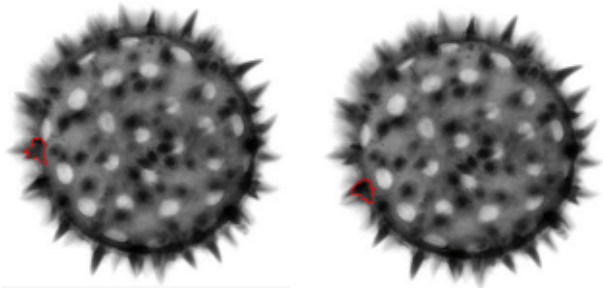


Figure 5.2: Periporate's repetitive distinguishing features. Pollen grain imaged by Willard et al. (2004)

5.1.2 Repetitive Internal Feature Descriptor

Another method may prove useful in identifying shapes with repetitive features, such as the spikes in Figure 5.2. A system to discover repetitive features that are the descendants of a larger shape could be developed to store, as metrics, their numbers and boundary descriptors. Such descriptors could be stored in the parent node to be quickly analyzed during an image search.

5.1.3 Subtree Comparison

A drawback of the image classifier created by Pan (2007) is that the user is required to select a shape in a given target image to compare to the candidate shape collection. The Shape Explorer tool allows one to use SID to select individual component shapes for comparison and view the results of the classification procedure on those shapes.

SID was not deeply impacted by this same limitation as the images examined had only one object present in the image against a white background, so only the child of the parent image was automatically selected for comparison. As one attempts to expand this system to include multiple independent objects in the same image, this limitation will become evident in SID as well. This shortcoming can be addressed by utilizing a multilevel analysis of the tree-of-shapes. Shapes from the root down of the target image can be compared to "generations" of shapes in the image database in a recursive comparison of subtrees. For example, given a global candidate match in Phase 1, the children of both the target and the candidate will be recursively traversed and compared to one another with the similarity of the subtrees used in the final R^2 similarity measure. For objects that contain subobjects, this procedure becomes a region-based method, but for relatively smooth objects without subobjects, region descriptors that make use of the texture within the object are required.

5.1.4 Region Descriptors

As stated in Section 2.3, region descriptors hold the potential to describe the internal features of shapes, even those that do not contain subobjects. The texture of the shape's interior can be analyzed to generate a region descriptor that can be utilized by a classifier to generate a similarity measure. A method has been devised that would allow the sampling of the intensity of the internal structures of the shape in a manner than is invariant to scale, rotation, translation, global contrast and intensity changes. The shape is mean-centered to position the centroid of the shape at coordinate point (0,0), which eliminates variance to translation. The intensity values of the shape are normalized to reduce global intensity change variance. From the centroid, one then takes a fixed number of equally spaced pixel neighborhood intensity samples, as shown in Figure 5.3. By equally spacing the neighborhoods of sample points based on the radii length, one eliminates scale variance. One then filters the sampled pixel neighborhood with, for example, a median filter, to reduce the number of data

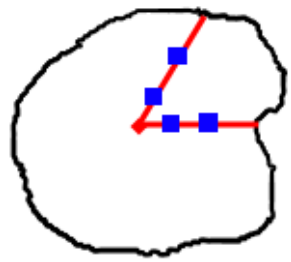


Figure 5.3: Shape outline with radii in red and sample neighborhoods in blue

points and to reduce noise (a smoothing of the interior of the shape). The value(s) resulting from this filter are then placed in a vector. This process would be repeated for a fixed number of radii of equal angle from one another across the surface of the shape. The attribute vector is then transformed into the frequency domain by a FFT. Dimensions are reduced (the number of which to reduce requires experimentation to determine) to reduce the impact of noise and to improve computational performance, as with the Fourier boundary descriptors in Section 2.3.4. Only the magnitude of the resulting frequency vector is kept to remove rotation variance. The resulting vector is then compared using cosine similarity to the region descriptor of other shapes generated using the same method, in a manner similar to Phase 2 of SID (Section 3.3.3). This additional phase should greatly aid with the classification of shapes that have generally smooth surfaces, like inaperturates, as well as providing additional information to the classifiers of other shapes.

5.1.5 Phase 1 Descriptors

The Phase 1 classification remains somewhat unexplored and much room for improvement exists. A detailed exploration of the metrics by a Principle Component Analysis (PCA) could offer insight into which metrics are useful and suggest which ones can be discarded to remove noise and improve performance. A deeper analysis could produce a weighting scheme to be used in conjunction with a similarity measure in Phase 2 to allow metrics that more distinctively capture shape information to be

weighted more heavily than those with lesser capabilities. Different metrics can also be introduced that help to describe the shape and not add too much computational complexity to the phase one comparison.

5.2 Closing Remarks

Pollen images were matched to members of their morphological group 75% of the time, with a member of the proper morphological group present in the top 3 results by R^2 value in every experiment. Two or more out of the top three results of each test image belonged to the proper morphological group 80% of the experiments conducted with the pairing image of the same species appearing in the top three results 85% of the time. The goal of achieving image invariance to rotation, translation, global contrast and intensity changes was shown by experiment to be at least partially successful. Further evaluation of scale changes with images captured using optical magnification rather than digital image pixel interpolation is required to fully verify scale invariance, but current experimental evidence shows promise, and the techniques used are theoretically invariant to scale. Various parameters for the classifier were evaluated with the results of their impact upon the accuracy of classification demonstrated. With the ideas outlined in Section 5.1, it is possible for this classification system to significantly improve.

Bibliography

Bibliography

Ballester, C., Caselles, V., and Monasse, P. (2003). The tree of shapes of an image. *ESAIM: Control, Optimization and Calculus of Variations*, 9:1–18. 32

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 24(24). 30

Brown, L. G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, 24(4). 21, 23, 24

Gonzalez, R. and Woods, R. (2007). *Digital image processing*. Prentice Hall. 3, 4, 10, 11, 12, 15, 19, 23, 26, 28, 39, 42, 50, 56, 74, 75, 83

Guichard, F., Morel, J.-M., and Ryan, R. (2000). Contrast invariant image analysis and pdes. 14, 15, 39

Hales, T. (2007). The jordan curve theorem, formally and informally. *The American Mathematical Monthly*, 114(10). 40

Kapp, R., Davis, O., and King, J. (2000). *Pollen and Spores*. The American Association of Stratigraphic Palynologists, 2 edition. 58, 90

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331. 12, 13, 14

Monasse, P. and Guichard, F. (2000). Fast computation of a contrast-invariant image representation. *IEEE Transactions on Image Processing*, 9. 18, 24, 31, 38, 39, 40, 41, 42, 43, 44, 51, 117

Nitzberg, M. and Mumford, D. B. (1990). The 2.1-d sketch. In *Proceedings of the Third International Conference on Computer Vision*. 25

Pan, Y. (2007). *Preferential Image Segmentation*. PhD thesis, The University of Tennessee. 20, 31, 45, 51, 91

Sethian, J. and Osher, S. (1987). Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49. 16

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison Wesley, 1 edition. 34

Weeratunga, S. and Kamath, C. (2004). Investigation of implicit active contours for scientific image segmentation. In *Visual Communications and Image Processing 2004*, volume 5308, pages 210–221. 14, 16, 17

Willard, D., Bernhardt, C., Weimer, L., Cooper, S., Gamez, D., and Jensen, J. (2004). Atlas of pollen and spores of the florida everglades. *Palynology*, 28:175–227. xi, xii, xiii, xiv, 8, 17, 58, 59, 60, 62, 63, 64, 65, 67, 68, 69, 70, 84, 91, 99

Zhang, D. and Lu, G. (2002). A comparative study on shape retrieval using fourier descriptors with different shape signatures. In *Proceedings of 5th Asian Conference on Computer Vision*. 19, 27, 28, 31, 32, 34, 50

Ziou, D. and Tabbone, S. (1998). Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559. 9, 10

Appendices

Appendix A

Pollen Image Test Set

A.1 Overview

The following images were used in the classification experiment discussed in Chapter 4. All pollen grains were collected and imaged by Willard et al. (2004). They are divided into their morphological groups. The images with artificial affine transformations applied are images from the same data set that were transformed using ImageMagick Studio's PerlMagick to add translation, rotation, scale and global contrast and intensity changes; they can be found in Section 4.1.

(a) pistia stratiotes

(b) pistia stratiotes

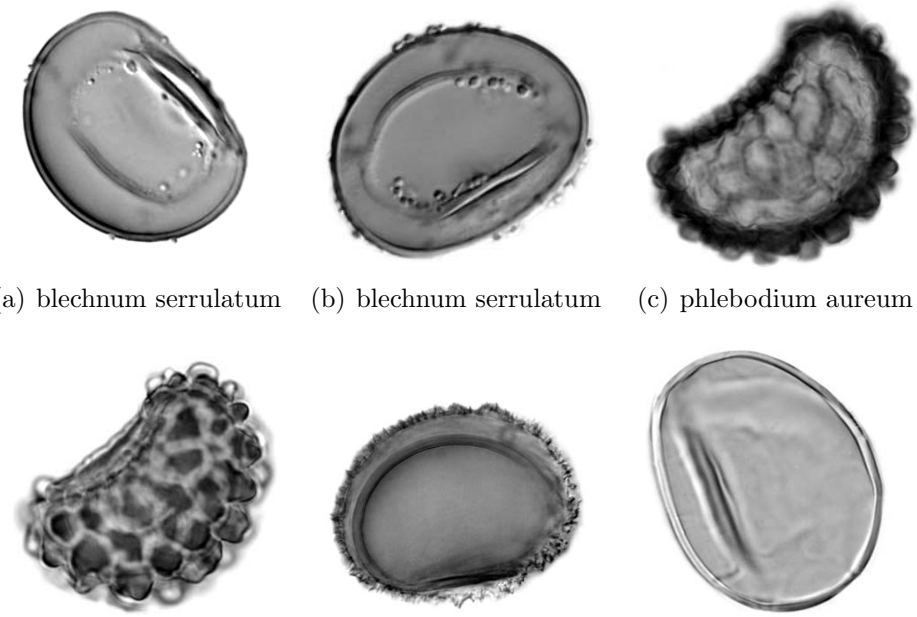
(c) rhynchospora colorata

(d) rhynchospora colorata

(e) schoenoplectus taberaemontani

(f) schoenoplectus taberaemontani

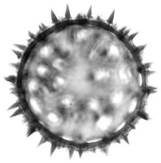
Figure A.1: Inaperturate pollen grains used in this experiment



(a) blechnum serrulatum (b) blechnum serrulatum (c) phlebodium aureum

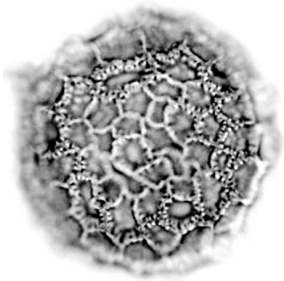
(d) phlebodium aureum (e) thelypteris kunthii (f) thelypteris kunthii

Figure A.2: Monolete pollen grains used in this experiment



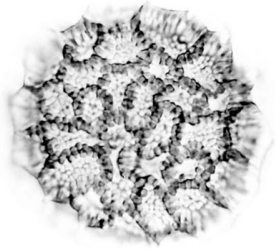
(a) ipomoea pes-caprae

(b) ipomoea pes-caprae



(c) polygonum densiflorum

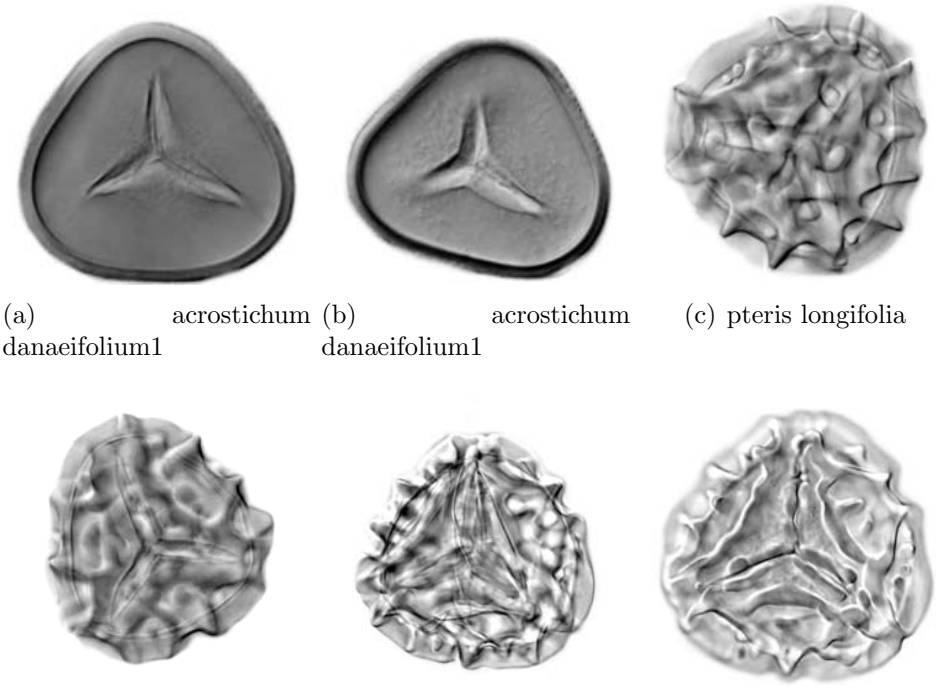
(d) polygonum densiflorum



(e) polygonum hydropiperoides

(f) polygonum hydropiperoides

Figure A.3: Periporate pollen grains used in this experiment



(a) acrostichum danaeifolium1 (b) acrostichum danaeifolium1 (c) pteris longifolia

(d) pteris longifolia (e) pteris vittata (f) pteris vittata

Figure A.4: Trilete pollen grains used in this experiment

Appendix B

Affine Transform Results

B.1 Overview

This appendix contains the top 10 matches in a comparison of each original image to the other images in the affine transform test set using their centroid distance Fourier descriptors. Translation, rotation and, to a lesser extent, global contrast and intensity transformed images typically score close to 1 (exact match) with their original image. Images with changes in scale tend to score significantly less. Transforms of different images typically score significantly less than either. A full analysis of the results listed in this appendix is provided in Section 4.7.

B.2 Results

B.2.1 Acrostichum danaeifolium1

Transform Type	Species Image	R^2 Value
contrast	Acrostichum danaeifolium1	1.0000
translation	Acrostichum danaeifolium1	1.0000
rotation	Acrostichum danaeifolium1	0.9999
scale	Acrostichum danaeifolium1	0.8237
scale	Acrostichum danaeifolium2	0.7366
intensity	Acrostichum danaeifolium2	0.7277
translation	Acrostichum danaeifolium2	0.7277
rotation	Acrostichum danaeifolium2	0.7253
scale	Pteris vittata1	0.7017
scale	Pteris longifolia1	0.6641

B.2.2 Acrostichum danaeifolium2

Transform Type	Species Image	R^2 Value
translation	Acrostichum danaeifolium2	1.0000
intensity	Acrostichum danaeifolium2	1.0000
rotation	Acrostichum danaeifolium2	0.9960
scale	Acrostichum danaeifolium2	0.8454
rotation	Pteris vittata2	0.7843
contrast	Pteris vittata2	0.7759
translation	Pteris vittata2	0.7759
rotation	Acrostichum danaeifolium1	0.7283
contrast	Acrostichum danaeifolium1	0.7277
translation	Acrostichum danaeifolium1	0.7277

B.2.3 Blechnum serrulatum1

Transform Type	Species Image	R^2 Value
contrast	Blechnum serrulatum1	1.0000
intensity	Blechnum serrulatum1	1.0000
translation	Blechnum serrulatum1	1.0000
scale	Blechnum serrulatum1	0.9870
rotation	Blechnum serrulatum1	0.9811
rotation	Pistia stratiotes1	0.8572
contrast	Pistia stratiotes1	0.8569
intensity	Pistia stratiotes1	0.8569
translation	Pistia stratiotes1	0.8569
scale	Pistia stratiotes1	0.8167

B.2.4 Blechnum serrulatum2

Transform Type	Species Image	R^2 Value
translation	Blechnum serrulatum2	1.0000
contrast	Blechnum serrulatum2	1.0000
intensity	Blechnum serrulatum2	1.0000
rotation	Blechnum serrulatum2	0.9843
contrast	Phlebodium aureum2	0.8102
intensity	Phlebodium aureum2	0.8102
translation	Phlebodium aureum2	0.8102
rotation	Phlebodium aureum2	0.8101
rotation	Thelypteris kunthii2	0.7481
contrast	Thelypteris kunthii2	0.7474

B.2.5 Ipomoea pes-caprae1

Transform Type	Species Image	R^2 Value
translation	Ipomoea pes-caprae1	1.0000
rotation	Ipomoea pes-caprae1	0.9993
scale	Ipomoea pes-caprae2	0.7071
intensity	Polygonum hydropiperoides1	0.6303
scale	Ipomoea pes-caprae1	0.5943
scale	Polygonum densiflorum2	0.5858
rotation	Polygonum densiflorum2	0.5820
contrast	Polygonum densiflorum2	0.5537
intensity	Polygonum densiflorum2	0.5537
translation	Polygonum densiflorum2	0.5537

B.2.6 Ipomoea pes-caprae2

Transform Type	Species Image	R^2 Value
contrast	Ipomoea pes-caprae2	1.0000
intensity	Ipomoea pes-caprae2	1.0000
translation	Ipomoea pes-caprae2	1.0000
rotation	Ipomoea pes-caprae2	0.9959
scale	Ipomoea pes-caprae2	0.7421
scale	Acrostichum danaeifolium1	0.6326
intensity	Ipomoea pes-caprae1	0.5916
scale	Pteris vittata1	0.5814
contrast	Pteris vittata2	0.5602
translation	Pteris vittata2	0.5602

B.2.7 Phlebodium aureum1

Transform Type	Species Image	R^2 Value
contrast	Phlebodium aureum1	1.0000
intensity	Phlebodium aureum1	1.0000
translation	Phlebodium aureum1	1.0000
rotation	Phlebodium aureum1	0.9917
scale	Phlebodium aureum1	0.8421
contrast	Phlebodium aureum2	0.8287
intensity	Phlebodium aureum2	0.8287
translation	Phlebodium aureum2	0.8287
rotation	Phlebodium aureum2	0.8284
rotation	Schoenoplectus taberaemontani2	0.8035

B.2.8 Phlebodium aureum2

Transform Type	Species Image	R^2 Value
contrast	Phlebodium aureum2	1.0000
intensity	Phlebodium aureum2	1.0000
translation	Phlebodium aureum2	1.0000
rotation	Phlebodium aureum2	1.0000
contrast	Phlebodium aureum1	0.8287
intensity	Phlebodium aureum1	0.8287
translation	Phlebodium aureum1	0.8287
scale	Phlebodium aureum1	0.8223
intensity	Rhynchospora colorata1	0.8172
rotation	Phlebodium aureum1	0.8122

B.2.9 Pistia stratiotes1

Transform Type	Species Image	R^2 Value
contrast	Pistia stratiotes1	1.0000
intensity	Pistia stratiotes1	1.0000
translation	Pistia stratiotes1	1.0000
rotation	Pistia stratiotes1	0.9942
scale	Pistia stratiotes1	0.9171
scale	Blechnum serrulatum1	0.8641
contrast	Blechnum serrulatum1	0.8569
intensity	Blechnum serrulatum1	0.8569
translation	Blechnum serrulatum1	0.8569
rotation	Blechnum serrulatum1	0.8529

B.2.10 Pistia stratiotes2

Transform Type	Species Image	R^2 Value
contrast	Pistia stratiotes2	1.0000
translation	Pistia stratiotes2	1.0000
rotation	Pistia stratiotes2	0.9965
scale	Pistia stratiotes2	0.8112
rotation	Blechnum serrulatum1	0.7881
scale	Blechnum serrulatum1	0.7644
contrast	Pistia stratiotes1	0.7639
intensity	Pistia stratiotes1	0.7639
translation	Pistia stratiotes1	0.7639
rotation	Thelypteris kunthii1	0.7568

B.2.11 Polygonum densiflorum1

Transform Type	Species Image	R^2 Value
translation	Polygonum densiflorum1	1.0000
contrast	Polygonum densiflorum1	1.0000
rotation	Polygonum densiflorum1	0.9956
scale	Polygonum densiflorum2	0.8112
rotation	Polygonum densiflorum2	0.7960
contrast	Polygonum densiflorum2	0.7832
intensity	Polygonum densiflorum2	0.7832
translation	Polygonum densiflorum2	0.7832
scale	Polygonum densiflorum1	0.7566
intensity	Polygonum densiflorum1	0.7316

B.2.12 Polygonum densiflorum2

Transform Type	Species Image	R^2 Value
contrast	Polygonum densiflorum2	1.0000
intensity	Polygonum densiflorum2	1.0000
translation	Polygonum densiflorum2	1.0000
rotation	Polygonum densiflorum2	0.9901
scale	Polygonum densiflorum2	0.9032
contrast	Polygonum densiflorum1	0.7832
translation	Polygonum densiflorum1	0.7832
rotation	Polygonum densiflorum1	0.7832
translation	Pteris longifolia1	0.7139
contrast	Pteris longifolia1	0.7139

B.2.13 Polygonum hydropiperoides1

Transform Type	Species Image	R^2 Value
contrast	Polygonum hydropiperoides1	1.0000
translation	Polygonum hydropiperoides1	1.0000
rotation	Polygonum hydropiperoides1	0.9965
scale	Polygonum hydropiperoides1	0.9655
rotation	Polygonum hydropiperoides2	0.6492
contrast	Polygonum hydropiperoides2	0.6481
intensity	Polygonum hydropiperoides2	0.6481
scale	Polygonum hydropiperoides2	0.6481
translation	Polygonum hydropiperoides2	0.6481
contrast	Thelypteris kunthii2	0.6423

B.2.14 Polygonum hydropiperoides2

Transform Type	Species Image	R^2 Value
translation	Polygonum hydropiperoides2	1.0000
contrast	Polygonum hydropiperoides2	1.0000
intensity	Polygonum hydropiperoides2	1.0000
scale	Polygonum hydropiperoides2	1.0000
rotation	Polygonum hydropiperoides2	0.9954
rotation	Polygonum densiflorum2	0.6831
scale	Polygonum densiflorum2	0.6714
contrast	Polygonum densiflorum2	0.6573
intensity	Polygonum densiflorum2	0.6573
translation	Polygonum densiflorum2	0.6573

B.2.15 Pteris longifolia1

Transform Type	Species Image	R^2 Value
contrast	Pteris longifolia1	1.0000
intensity	Pteris longifolia1	1.0000
translation	Pteris longifolia1	1.0000
rotation	Pteris longifolia1	0.9949
scale	Pteris vittata1	0.8089
scale	Polygonum densiflorum2	0.7580
rotation	Pteris vittata1	0.7503
contrast	Pteris vittata1	0.7491
intensity	Pteris vittata1	0.7491
translation	Pteris vittata1	0.7491

B.2.16 Pteris longifolia2

Transform Type	Species Image	R^2 Value
contrast	Pteris longifolia2	1.0000
translation	Pteris longifolia2	1.0000
rotation	Pteris longifolia2	0.9923
intensity	Pteris longifolia2	0.9840
scale	Pteris longifolia2	0.9773
rotation	Pteris vittata1	0.7136
rotation	Acrostichum danaeifolium2	0.7066
translation	Acrostichum danaeifolium2	0.7046
intensity	Acrostichum danaeifolium2	0.7046
contrast	Pteris vittata1	0.7020

B.2.17 Pteris vittata1

Transform Type	Species Image	R^2 Value
contrast	Pteris vittata1	1.0000
intensity	Pteris vittata1	1.0000
translation	Pteris vittata1	1.0000
rotation	Pteris vittata1	0.9886
rotation	Pteris longifolia1	0.7721
translation	Pteris longifolia1	0.7491
contrast	Pteris longifolia1	0.7491
intensity	Pteris longifolia1	0.7491
scale	Pteris vittata1	0.7323
scale	Acrostichum danaeifolium1	0.7193

B.2.18 Pteris vittata2

Transform Type	Species Image	R^2 Value
contrast	Pteris vittata2	1.0000
translation	Pteris vittata2	1.0000
rotation	Pteris vittata2	0.9947
scale	Pteris longifolia1	0.8279
intensity	Acrostichum danaeifolium2	0.7759
translation	Acrostichum danaeifolium2	0.7759
rotation	Acrostichum danaeifolium2	0.7667
scale	Pteris vittata2	0.7327
intensity	Polygonum densiflorum1	0.6777
scale	Acrostichum danaeifolium1	0.6730

B.2.19 Rhynchospora colorata1

Transform Type	Species Image	R^2 Value
contrast	Rhynchospora colorata1	1.0000
translation	Rhynchospora colorata1	1.0000
rotation	Rhynchospora colorata1	0.9998
scale	Rhynchospora colorata1	0.7835
scale	Pistia stratiotes2	0.7292
contrast	Schoenoplectus taberaemontani2	0.6993
intensity	Schoenoplectus taberaemontani2	0.6993
translation	Schoenoplectus taberaemontani2	0.6993
rotation	Schoenoplectus taberaemontani2	0.6981
contrast	Phlebodium aureum1	0.6899

B.2.20 Rhynchospora colorata2

Transform Type	Species Image	R^2 Value
contrast	Rhynchospora colorata2	1.0000
intensity	Rhynchospora colorata2	1.0000
translation	Rhynchospora colorata2	1.0000
rotation	Rhynchospora colorata2	0.9929
scale	Rhynchospora colorata2	0.8337
scale	Thelypteris kunthii2	0.8153
rotation	Phlebodium aureum2	0.8093
contrast	Phlebodium aureum2	0.8092
intensity	Phlebodium aureum2	0.8092
translation	Phlebodium aureum2	0.8092

B.2.21 Schoenoplectus taberaemontani1

Transform Type	Species Image	R^2 Value
contrast	Schoenoplectus taberaemontani1	1.0000
translation	Schoenoplectus taberaemontani1	1.0000
rotation	Schoenoplectus taberaemontani1	0.9981
scale	Schoenoplectus taberaemontani1	0.9820
intensity	Schoenoplectus taberaemontani1	0.8482
scale	Schoenoplectus taberaemontani2	0.6297
rotation	Schoenoplectus taberaemontani2	0.5946
contrast	Schoenoplectus taberaemontani2	0.5845
intensity	Schoenoplectus taberaemontani2	0.5845
translation	Schoenoplectus taberaemontani2	0.5845

B.2.22 Schoenoplectus taberaemontani2

Transform Type	Species Image	R^2 Value
contrast	Schoenoplectus taberaemontani2	1.0000
intensity	Schoenoplectus taberaemontani2	1.0000
translation	Schoenoplectus taberaemontani2	1.0000
rotation	Schoenoplectus taberaemontani2	0.9935
scale	Phlebodium aureum1	0.8451
scale	Schoenoplectus taberaemontani2	0.8115
contrast	Acrostichum danaeifolium2	0.8075
rotation	Phlebodium aureum2	0.7983
contrast	Phlebodium aureum1	0.7982
intensity	Phlebodium aureum1	0.7982

B.2.23 Thelypteris kunthii1

Transform Type	Species Image	R^2 Value
contrast	Thelypteris kunthii1	1.0000
intensity	Thelypteris kunthii1	1.0000
translation	Thelypteris kunthii1	1.0000
rotation	Thelypteris kunthii1	0.9884
scale	Thelypteris kunthii1	0.8386
scale	Blechnum serrulatum2	0.8331
contrast	Thelypteris kunthii2	0.7990
intensity	Thelypteris kunthii2	0.7990
translation	Thelypteris kunthii2	0.7990
scale	Rhynchospora colorata2	0.7978

B.2.24 Thelypteris kunthii2

Transform Type	Species Image	R^2 Value
contrast	Thelypteris kunthii2	1.0000
intensity	Thelypteris kunthii2	1.0000
translation	Thelypteris kunthii2	1.0000
rotation	Thelypteris kunthii2	0.9837
rotation	Thelypteris kunthii1	0.8105
scale	Blechnum serrulatum2	0.8035
contrast	Thelypteris kunthii1	0.7990
intensity	Thelypteris kunthii1	0.7990
translation	Thelypteris kunthii1	0.7990
rotation	Phlebodium aureum2	0.7835

Appendix C

Source Code

C.1 Overview

This section contains a listing of the source code of the software used to explore the data set, generate the affine transformations and to conduct the bulk of the experiments in Chapter 4. The implementation of the FLLT is not included due to its excessive length and the original version by Monasse and Guichard (2000) is available from Megawave's website. Being a viewer, the shape explorer GUI's code is excluded and any classifier code present has an equivalent in the GNU Octave scripts below.

C.2 Code

C.2.1 Fourier Descriptor Analysis

This GNU Octave script creates centroid distance Fourier descriptors from a set of boundary points and then performs an all-pairs comparison for all shapes in the input file:

```
%
% Input
```

```
%   boundaries input file, output file path, sample size
% Output
%   All-pairs CSV file for R2 similarity
%
function [] = analyze_bound(INFILE, OUTFILE, sample_size)
  in = csvread(INFILE);
  [m n] = size(in);


  lgss = log(sample_size/2)/log(2);
  %K = 2.^(1:lgss);
  K = 2:2:2^lgss;
  k_len = length(K);
  MAX_K = max(K);

  %
  % Generate Fourier Descriptors
  %
    data = zeros(m, MAX_K);
  for i=1:m
    data(i,:) = compute_fd(in(i,:), MAX_K, sample_size);
  end

    %
  % Calculate R2 Values for each dimension K
  %
    for d=1:k_len
    k = K(d);

    % Determine space required
    r2_vals_m = ceil(factorial(m) / (2*factorial(m-2)));
    r2_vals = zeros(r2_vals_m, 7);

    % List sample size and k value
```

```
    r2_vals(:,1) = ones(r2_vals_m, 1)*sample_size;
    r2_vals(:,2) = ones(r2_vals_m, 1)*k;

    % Perform all-pairs comparison
    pos = 1;
    for i=1:m
      for j=(i+1):m
        % Import Image ID
        r2_vals(pos, 3) = in(i, 1);
        r2_vals(pos, 4) = in(j, 1);

        % Import Group ID
        r2_vals(pos, 5) = in(i, 2);
        r2_vals(pos, 6) = in(j, 2);

        % Calculate cosine similarity
        r2_vals(pos, 7) = cos_sim(data(i, 1:k), data(j, 1:k));

        pos = pos + 1;
      end
    end

    % Sort results by decreasing R2 value
     r2_vals = sortrows(r2_vals, -7);

    % SAMPLE SIZE | K | ID 1 | ID 2 | GROUP 1 | GROUP 2 | R2
    dlmwrite(OUTFILE, r2_vals, '-append', 'delimiter', ',', 'precision',
        '%1.4f');
  end
end

%
% input:
```

```
%   Inline:   line from CSV:  ID, Group ID, Boundary_Length, Boundary_0
    ...Boundary_n
%   sample_size:  Number of samples from the boundary%
% output:
%   fourier descriptor (minus DC component)
%
function [fd] = compute_fd(inLine, MAX_K, sample_size)

  [m n] = size(inLine);
  boundary_len = inLine(3);

  % Adjust n to match the end of the boundary
  n = 3 + 2*boundary_len;

  x = zeros(1, boundary_len);
  y = zeros(1, boundary_len);

  % Extract line into coordinate vectors
  pos = 1;
  for i=4:2:n
    x(pos) = inLine(i);
    pos = pos + 1;
  end

  pos = 1;
  for i=5:2:n
    y(pos) = inLine(i);
    pos = pos + 1;
  end

  % mean center (centroid at coordinate 0,0)
  mean_x = mean(x);
  x = x - mean_x;
```

```
    mean_y = mean(y);
    y = y - mean_y;

    % create interpolant
    xx = angle(x+sqrt(-1)*y);

    % Determine radius
    r = abs(x+sqrt(-1)*y);

    % Scale radii
    r_norm = norm(r);
    r = r / r_norm;

    % Sample the iterpolated boundary
    % polarx = 0:(2*pi/boundary_len):2*pi*(1-1/boundary_len);
    % sampley = spline(polarx,r,(0:(2*pi/sample_size):2*pi*(1-1/
        sample_size)));
    samplex3 = 0:(2*pi/boundary_len):(6*pi-2*pi/boundary_len);
    sampley3 = spline(samplex3,[r r r],(0:(2*pi/sample_size):(6*pi-2*pi/
        sample_size)));
    sampley = sampley3((floor(length(sampley3)/3)+1):floor(length(sampley3
        ))/3+sample_size);
    % samplex = 0:(2*pi/boundary_len):(2*pi*(1-1/boundary_len));

    % Get magnitude, discard DC component
    fd = abs(fft(sampley));

    fd = fd(2:(MAX_K+1));

end

% Compute cosine similarity between two vectors
function [r2] = cos_sim(v1, v2)
  m = length(v1);
```

```
  % Multiply by index
  ind = 1:m;

  v1 = v1 .* ind;
  v2 = v2 .* ind;

  v_dot = dot(v1, v2);
  v1_mag = norm(v1);
  v2_mag = norm(v2);

  # Compute DOT Product
  r2 = v_dot / (v1_mag*v2_mag);

  # R2
  r2 = r2 * r2;
end
```

This GNU Octave script evaluates different sample sizes and dimension retention for Fourier descriptors:

```
function eval_k_ss(BOUNDARIES_FILE, R2_FILE, RES_PATH, SAMPLE_SIZE)
  ss_len = length(SAMPLE_SIZE);



  delete(R2_FILE);

  for i=1:length(SAMPLE_SIZE)
    s = SAMPLE_SIZE(i);
    analyze_bound(BOUNDARIES_FILE, R2_FILE, s);
  end

  r2 = csvread(R2_FILE);

  imgs = unique([r2(:, 3) r2(:, 4)]);
  imgs_len = length(imgs);
```

```
grps = unique([r2(:, 5) r2(:, 6)]);
grps_len = length(grps);

summary = [];
% Traverse Sample Sizes
for ss = 1:ss_len
  s = SAMPLE_SIZE(ss);

  % determine K size
  lgss = log(s/2)/log(2);
  %K = 2.^(1:lgss);
  K = 2:2:2^lgss;
  k_len = length(K);

  % limit results to this group
  lim = r2(:, 1) == s;
  r2_ss = r2(lim,:);

  for kk=1:k_len
    k = K(kk);
    grp_tally = zeros(grps_len, 4);

    grp_tally(:,1) = ones(grps_len, 1) * s;
    grp_tally(:,2) = ones(grps_len, 1) * k;

    lim = r2_ss(:, 2) == k;
    r2_k = r2_ss(lim,:);
    % Traverse images
    for ii=1:imgs_len
      id = imgs(ii);

      grp_tally = calc_tally(r2_k, id, grp_tally);
      % Limit to proper ss, k and id number
    end
```

```
        summary = [summary; grp_tally];
      end
    end
    summary(:, 3:4) = summary(:, 3:4) ./ 6;

    plot_res(summary, RES_PATH);
end

function [grp_tally] = calc_tally(r2, id, grp_tally)
    lim_ida = r2(:,3) == id;
    lim_idb = r2(:,4) == id;

    lim = (lim_ida | lim_idb);
    img = r2(lim,:);
    img = img(1:3,:);

    if img(1, 3) == id
      grp = img(1, 5);
    else
      grp = img(1, 6);
    end

    if img(1,5) == img(1,6)
      grp_tally(grp,3) = grp_tally(grp,3) + 1;
    end

    maj = 0;
    for i=1:3
      if img(i,5) == img(i,6)
        maj = maj + 1;
      end
    end

    if maj >= 2
```

```
    grp_tally(grp,4) = grp_tally(grp,4) + 1;
  end
end

function plot_res(summary, RES_PATH)

  % Plot the results for K
  K_CSV_PATH = strcat(RES_PATH, '/k_summary.csv');
  K_PLOT_PATH = strcat(RES_PATH, '/k_summary.eps');
  % Plot sample size results
  s = 256;
  lim = summary(:,1) == s;
  res = summary(lim, :);
  K = unique(res(:,2));
  k_len = length(K);

  summary_k = zeros(k_len, 5);

  delete(K_CSV_PATH);
  for i=1:k_len
    k = K(i);
    lim = res(:,2) == k;
    sum_k = res(lim,:);

    mean_hit = mean(sum_k(:, 3));
    mean_maj = mean(sum_k(:, 4));

    worst_hit = min(sum_k(:, 3));
    worst_maj = min(sum_k(:, 4));

    summary_k(i, 1) = k;
    summary_k(i, 2) = mean_hit;
    summary_k(i, 3) = mean_maj;
    summary_k(i, 4) = worst_hit;
```

```
  summary_k(i, 5) = worst_maj;
end
dlmwrite(K_CSV_PATH, summary_k, '-append', 'delimiter', ',', '
    precision', '%1.4f');

hold on;
axis equal;
%plot(summary_k(:,1), summary_k(:,2), '--rs','MarkerSize',7);
plot(summary_k(:,1), summary_k(:,3), '--gs','MarkerSize',7);
%plot(summary_k(:,1), summary_k(:,4), '--bs','MarkerSize',7);
plot(summary_k(:,1), summary_k(:,5), '--ms','MarkerSize',7);
title('Dimensionality evaluation for morphological group
    classification (256-Sample Size)');
xlabel('Dimensions Retained');
ylabel('Accuracy');
%legend('Average Hit', 'Average Majority', 'Minimum Hit', 'Minimum
    Majority', 'Location', 'South');
legend('Average', 'Minimum', 'Location', 'South');
hold off;

print(K_PLOT_PATH);
close all;

% Plot the results for Sample Size
S_CSV_PATH = strcat(RES_PATH, '/s_summary.csv');
S_PLOT_PATH = strcat(RES_PATH, '/s_summary.eps');

S = unique(summary(:,1));
s_len = length(S);


summary_s = zeros(s_len, 5);

delete(S_CSV_PATH);
```

```
for i=1:s_len
  s = S(i);

  lim_s = summary(:,1) == s;
  lim_k = summary(:,2) == floor(s/4);
  lim = lim_s & lim_k;

  sum_s = summary(lim,:);

  mean_hit = mean(sum_s(:, 3));
  mean_maj = mean(sum_s(:, 4));

  worst_hit = min(sum_s(:, 3));
  worst_maj = min(sum_s(:, 4));

  summary_s(i, 1) = s;
  summary_s(i, 2) = mean_hit;
  summary_s(i, 3) = mean_maj;
  summary_s(i, 4) = worst_hit;
  summary_s(i, 5) = worst_maj;
end

dlmwrite(S_CSV_PATH, summary_s, '-append', 'delimiter', ',', '
    precision', '%1.4f');

hold on;
axis equal;
%plot(summary_s(:,1), summary_s(:,2), '--rs','MarkerSize',7);
plot(summary_s(:,1), summary_s(:,3), '--gs','MarkerSize',7);
%plot(summary_s(:,1), summary_s(:,4), '--bs','MarkerSize',7);
plot(summary_s(:,1), summary_s(:,5), '--ms','MarkerSize',7);
title('Sample size evaluation for morphological group classification')
    ;
```

```
  xlabel('Sample Size');
  ylabel('Accuracy');
  %legend('Average Hit', 'Average Majority', 'Minimum Hit', 'Minimum
      Majority', 'Location', 'South');
  legend('Average', 'Minimum', 'Location', 'South');
  hold off;

  print(S_PLOT_PATH);
  close all;
end
```

C.2.2 TOS Metrics

This GNU Octave script is used to help determine optimal TOS Metric Phase 1 constraints using a set of known image classes:

```
function analyze_tos(INPUT_FILE, RES_PATH)
  in = csvread(INPUT_FILE);
  [m n] = size(in);

  % Read TOS Metrics
  % image_id, group_id, child_sum_area_ratio, int_ratio,
      boundary_size_ratio, dc
  n = 6;
  tos = zeros(m, n);
  tos(:, 1:2) = in(:, 1:2);
  for i=1:m
    boundary_len = in(i, 3)*2;

    tos(i, 3:6) = in(i, (4+boundary_len):(7+boundary_len));
  end

  % Get image IDs
  imgs = unique(tos(:,1));
  imgs_len = length(imgs);
```

```
  % Get group IDs
  grps = unique(tos(:,2));
  grps_len = length(grps);

  % Set Weights

  weights = [3,4,3,5.2];
  summary
  for i=1:m
    % Apply constraints
    lim = tos(:,1) ~= i;
    res_tos = tos(lim,:);
    for j=3:6
      lim_u = res_tos(:, j) <= tos(i, j) * weights(j-2);
      lim_l =  res_tos(:, j) >= tos(i, j) / weights(j-2);
      lim = lim_u & lim_l;
      res_tos = res_tos(lim,:);
    end

  end

  plot_summary(RES_PATH, grps, grps_len, tos)

end

function plot_summary(RES_PATH, grps, grps_len, tos)


  %image_id, group_id, child_sum_area_ratio, int_ratio,
      boundary_size_ratio, dc
  summary = zeros(grps_len, 13);
  summary(:,1) = grps;
  for i = 1:grps_len
```

```
  lim = tos(:,2) == grps(i);
  sum_g = tos(lim,:);

  [m n] = size(sum_g);
  for j = 1:n
    sum_g(:,j) = sum_g(:,j) / norm(sum_g(:,j));
  end

  for j = 1:n
    sum_g(:,j) = sum_g(:,j) - mean(sum_g(:,j));
  end

  for j = 0:3
    summary(i, j*4+2) = max(sum_g(:,3+j));
    summary(i, j*4+3) = median(sum_g(:,3+j));
    summary(i, j*4+4) = min(sum_g(:,3+j));
    summary(i, j*4+5) = std(sum_g(:,3+j));
  end
end

colors=['r', 'g', 'b','k'];
titles={'Child Sum Area Ratio', 'Intensity Ratio', 'Boundary Size
    Ratio', 'DC'};
save_name={'csar_plot', 'int_plot', 'bs_plot', 'dc_plot'};

for i = 0:3
  figure
  hold on;
  title(char(titles(i+1)));
  plot(1:4, summary(:, i*4+2)', '--rs', 'MarkerSize', 7);
  plot(1:4, summary(:, i*4+3)', '--gs', 'MarkerSize', 7);
  plot(1:4, summary(:, i*4+4)', '--bs', 'MarkerSize', 7);
  axis([0.5 4.5 -.65 .65]);
  legend('Maximum', 'Median', 'Minimum');
```

```
    set(gca,'XTickLabel',{'Inaperturate','Monolete','Periporate', '
        Trilete'});
    xlabel('Pollen Morphological Groups');
    ylabel('Values (normalized, mean-centered)');
    hold off;
    save_path = strcat(RES_PATH, '/', char(save_name(i+1)), '.eps');
    print (save_path);
  end
end
```

C.2.3 Affine Transformations

This GNU Octave script is used to evaluate the R^2 values of original images with their affine transformations:

```
close all;
clc;
clear all;

K = 32;
SAMPLE_SIZE = 256;

BOUNDARIES_FILE = 'results/transform_boundaries.csv';
R2_FILE = 'results/transform_r2.csv';

delete(R2_FILE);

analyze_bound(BOUNDARIES_FILE, R2_FILE, SAMPLE_SIZE);

% Read input data from CSV files (omit sample size and K)
in = csvread(R2_FILE);
[m n] = size(in);

% Only use specified K value
lim = in(:,2) == K;
```

```
r2 = in(lim,3:n);

% Determine number of images
transform_type_m = length(unique([r2(:,1) r2(:,2)]));

% Locate IDs for each type
con_ids = 1:6:transform_type_m;
int_ids = 2:6:transform_type_m;
orig_ids = 3:6:transform_type_m;
rot_ids = 4:6:transform_type_m;
sca_ids = 5:6:transform_type_m;
tran_ids = 6:6:transform_type_m;

orig_m = length(orig_ids);

% Get R2 values containing an original image
orig_r2a = ismember(r2(:,1), orig_ids);
orig_r2b = ismember(r2(:,2), orig_ids);
orig_r2_not = orig_r2a & orig_r2b; % remove original comparisons with
    each other
orig_r2 = (orig_r2a | orig_r2b) & ~orig_r2_not;
orig_r2 = r2(orig_r2,:);

orig_r2_m = length(orig_r2);

con_hit = 0;
tran_hit = 0;
int_hit = 0;
sca_hit = 0;
rot_hit = 0;
unrelated_hit = 0;

summary = [];
for i = 1:orig_m
```

```
% limit results to a single image id
r2_lim_a = orig_r2(:,1) == orig_ids(i);
r2_lim_b = orig_r2(:,2) == orig_ids(i);
r2_lim = r2_lim_a | r2_lim_b;
r2_i = orig_r2(r2_lim,:);

% Place original image in left most slot
for j = 1:10
  if r2_i(j, 1) ~= orig_ids(i)
    tmp = r2_i(j, 1);
    r2_i(j, 1) = r2_i(j, 2);
    r2_i(j, 2) = tmp;

    tmp = r2_i(j, 3);
    r2_i(j, 3) = r2_i(j, 4);
    r2_i(j, 4) = tmp;
  end
end
summary = [summary; r2_i(1:10,:)];

% limit it to > .95 r2
r2_lim = r2_i(:,5) > .95;
r2_i = r2_i(r2_lim,:);
[r2_i_m foo] = size(r2_i);

for j=1:r2_i_m
  if r2_i(j,3) ~= r2_i(:,4)
    unrelated_hit = unrelated_hit + 1;
    continue;
  end

  if ismember(r2_i(j, 1), con_ids) || ismember(r2_i(j, 2), con_ids)
    con_hit = con_hit + 1;
  end
```

```
    if ismember(r2_i(j, 1), tran_ids) || ismember(r2_i(j, 2), tran_ids)
      tran_hit = tran_hit + 1;
    end

    if ismember(r2_i(j, 1), int_ids) || ismember(r2_i(j, 2), int_ids)
      int_hit = int_hit + 1;
    end

    if ismember(r2_i(j, 1), sca_ids) || ismember(r2_i(j, 2), sca_ids)
      sca_hit = sca_hit + 1;
    end

    if ismember(r2_i(j, 1), rot_ids) || ismember(r2_i(j, 2), rot_ids)
      rot_hit = rot_hit + 1;
    end

  end
end

csvwrite(R2_FILE, summary);

% Compute accuracy and plot results
con_hit = con_hit / orig_m;
tran_hit = tran_hit  / orig_m;
int_hit = int_hit / orig_m;
sca_hit = sca_hit / orig_m;
rot_hit = rot_hit / orig_m;

figure
bar([con_hit tran_hit int_hit sca_hit rot_hit]);
set(gca,'XTickLabel',{'', 'Contrast', 'Translation', 'Intensity', '
    Scaling', 'Rotation', ''});
title('Affine transformation invariance test results');
```

```
xlabel('Transformation');
ylabel('Correctly Mattched');

print -deps 'results/af_fd_summary_plot.eps';
```

This Perl script, using Imagemagick Studio's PerlMagick, creates affine transformations from an original set of images:

```
#!/usr/bin/perl

use constant IN_PATH => '/home/qhaas/Documents/thesis/paper/figures/test
    ';
use constant OUT_PATH => '/home/qhaas/Desktop/fd_experiment/
    test_transforms';

use warnings;
use File::Find;
use Image::Magick;
use Math::Round;

sub rands();

# Get list of Morphological groups
opendir(DIR, IN_PATH);
my @morph_groups = readdir(DIR);
closedir(DIR);

foreach my $group (@morph_groups){
  next if ($group eq '.' or $group eq '..' or $group eq '.svn'); # omit
      basic file markers

  opendir(DIR, IN_PATH.'/'.$group);
  my @images = readdir(DIR);
  closedir(DIR);

  # Get list of images in morphological groups
```

```
foreach my $image (@images){
  next if ($image eq '.' or $image eq '..' or $image eq '.svn'); #
      omit basic file markers

  my ($imgPath, $outPath, $img, $af_img, $r, $rx, $ry, $width, $height
      );

  $imgPath = IN_PATH.'/'.$group.'/'.$image;
  $image =~ s/.png//g;
  $outPath = OUT_PATH.'/'.$image.'/';

  print "Transforming:\t$image... ";

  mkdir($outPath);

  # Load original image and reframe it (prevent shattering)
  $img = new Image::Magick;
  $img->read($imgPath);
  $img->Border('width'=>15,'height'=>15, 'bordercolor'=>'white');
  ($width, $height) = $img->Get('width', 'height');
  $img->write($outPath.'original.png');

  # Apply contrast change
  $af_img = $img->clone();
  $r = round(rand(2)) + 1;
  $af_img->SigmoidalContrast('contrast'=>$r, 'sharpen'=>round(rand()))
      ;
  $af_img->write($outPath.'contrast.png');

  # Apply intensity change
  $af_img = $img->clone();
  $r = .85 + round(rand(30))*.01;
  $af_img->Evaluate('value'=>$r, 'operator'=>'multiply');
  $af_img->write($outPath.'intensity.png');
```

```
    # Scale Image
    $af_img = $img->clone();
    $r = .85 + round(rand(30))*.01;
    $af_img->AdaptiveResize('width'=>($width*$r), 'height'=>($height*$r)
        , 'blur'=>0);
    $af_img->write($outPath.'scale.png');

    # Rotate Image
    $af_img = $img->clone();
    $af_img->AffineTransform('affine'=>[-1,0,0,rands(),0,0]);
    $af_img->write($outPath.'rotation.png');

    # Translate Image
    $af_img = $img->clone();
    $rx = round(rand(round($width*.1)));
    $ry = round(rand(round($height*.1)));
    $af_img->Extent('width'=>($width + $rx),'height'=>($height + $ry), '
        x'=> $rx, 'y'=> $ry, 'bordercolor'=>'white');
    $af_img->write($outPath.'translation.png');

    print "done\n";
  }
}

# Returns a random sign
sub rands() {
  if (round(rand()) > 0) {
    return 1;
  } else {
    return -1;
  }
}
```

Vita

The author, Quent Haas, hails from a small town in a South Mississippi, a little less than a couple of hours drive from New Orleans. The county's soil carried more oil and clay than nutrients, thus oil wells and manufacturing plants provided work for its inhabitants rather than the large soybean and cotton fields that dot the landscape in the western part of the state. His father worked in a data center for the company that invented Masonite and still bears its name, a biologist turned information technology specialist. His mother is a local artist whose exquisite paintings lined the walls of their house and the houses of their family. From the day his father brought home a Gateway 2000 386 PC, the young Quent grew quite fond of computing.

In his middle school, he learned the basics of programming and wrote software to help his mother, an art teacher, keep track of her student's grades. Graduating as the top ranking male in his high school class with highest honors and entering South Jone's High School's Hall of Fame, he also received an award for the highest achievements in his mathematics and science classes in 2004. His interest in computers lead him to major in Computer Science with minors in mathematics and biology at The University of Southern Mississippi, one of the largest universities in Mississippi. At Southern Miss, he won first place in two consecutive NASA sponsored Innovative Design Competitions and completed an undergraduate thesis on computer virus detection. He graduated Southern Miss with Summa Cum Laude honors and a 4.0 GPA in 2009.

After a summer internship at Redstone Arsenal in Hunstville, Alabama with The US Army's Aviation and Missile Research Development and Engineering Center (AMRDEC), he entered graduate school at The University of Tennessee in Knoxville to pursue a Master of Science in Computer Science, not far from where his family vacationed routinely as a child. Under the guidance of his supervising professors, Dr. Douglas Birdwell and Dr. Tse-Wei Wang, his knowledge and skills expanded quickly, resulting in invention disclosures. After graduation, he will start his new career as a software engineer at research and development powerhouse Alphatech in Burlington, MA, a spin-off of The Massachusetts Institute of Technology that was purchased by BAE Systems to form BAE Systems Advanced Information Technologies.