

University of New Hampshire

## University of New Hampshire Scholars' Repository

---

Master's Theses and Capstones

Student Scholarship

---

Fall 2020

# RECURSIVE PATH FOLLOWING IN LOG POLAR SPACE FOR AUTONOMOUS LEAF CONTOUR EXTRACTION

Anna Claire Snarski

*University of New Hampshire, Durham*

Follow this and additional works at: <https://scholars.unh.edu/thesis>

---

### Recommended Citation

Snarski, Anna Claire, "RECURSIVE PATH FOLLOWING IN LOG POLAR SPACE FOR AUTONOMOUS LEAF CONTOUR EXTRACTION" (2020). *Master's Theses and Capstones*. 1399.

<https://scholars.unh.edu/thesis/1399>

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact [nicole.hentz@unh.edu](mailto:nicole.hentz@unh.edu).

RECURSIVE PATH FOLLOWING IN LOG POLAR SPACE FOR  
AUTONOMOUS LEAF CONTOUR EXTRACTION

BY

ANNA SNARSKI

Bachelor of Science in Electrical Engineering, University of New Hampshire, 2018

THESIS

Submitted to the University of New Hampshire  
in Partial Fulfillment of  
the Requirements for the Degree of

Master of Science  
in  
Electrical and Computer Engineering

September 2020

This thesis was examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering by:

Thesis Director, Dr. Richard A. Messner

Associate Professor, Electrical and Computer Engineering, University of New Hampshire

Dr. John LaCourse

Professor, Electrical and Computer Engineering, University of New Hampshire

Dr. Wayne Smith

Senior Lecturer, Electrical and Computer Engineering, University of New Hampshire

On June 25, 2020

Approval signatures are on file with the University of New Hampshire Graduate School.

# Acknowledgements

I have the deepest gratitude to my advisor, Professor Richard A. Messner, for providing guidance and feedback throughout my research. Thank you for not only being my advisor for the last six years but offering rewarding yet painfully challenging courses. I wish to thank my committee members, Professor Wayne Smith and Professor John LaCourse, for their time reviewing my thesis.

My graduate career would not have been possible without the support of my family. Words cannot express how thankful I am to my parents and brothers for the encouragement over the years. A special thank you to my mother for her selflessness and unconditional love.

I am grateful to my friends that encouraged me to enjoy my life when my research felt never ending. Thank you, JP, for the solidarity and support as we completed our graduate career together and for taking four-hour study breaks to play video games. Thank you, Leah, for the continuous friendship throughout college and always being there for me. Finally, a special thank you to Adam who has been by my side throughout graduate school and for making sure my coffee cup is always full.



# Table of Contents

Acknowledgements .....	iii
List of Figures .....	v
List of Tables .....	ix
Abstract .....	x
Introduction .....	1
Chapter 1 Background.....	4
1.1 LOG POLAR TRANSFORM.....	4
1.2 COLOR SPACE.....	8
1.3 CHAIN CODE .....	10
1.4 MAXIMAL SIMILARITY-BASED REGION MERGING.....	11
Chapter 2 Methods .....	13
2.1 IMAGE DATABASES.....	13
2.2 ALGORITHM DESCRIPTION.....	17
2.2.1 Log Polar Space.....	17
2.2.2 Extracted Edges Algorithm.....	20
2.2.3 Start Location Algorithm and Node Finder Algorithm .....	26
2.2.4 Path Finder Algorithm .....	29
2.2.5 Contour Finder Algorithm.....	33
2.3 CATEGORIZING RESULTS.....	38
Chapter 3 Results .....	40
3.1 ALGORITHM CONTOUR RESULTS.....	40
3.2 THRESHOLD VALUES .....	47
3.3 MSRM RESULTS.....	52
3.4 TEST IMAGE DATABASE RESULTS .....	55
3.5 LOG POLAR SPACE SIZE .....	69
3.6 CARTESIAN SPACE IMAGE SIZE.....	75
Chapter 4 Conclusion .....	81
Chapter 5 Future Improvements .....	82
References .....	84

# List of Figures

Figure 1.1: Cartesian Space to Log Polar Space.....	5
Figure 1.2: Scale and Rotation Tolerant.....	5
Figure 1.3: Schematic of RGB Color Cube [28] .....	8
Figure 1.4: RGB Color Cube [28].....	8
Figure 1.5: L*a*b* Color Space .....	9
Figure 1.6: 8-Neighborhood Connectivity .....	10
Figure 1.7: Chain Code Example.....	10
Figure 1.8: Bhattacharyya Coefficient Example.....	11
Figure 1.9: MSRM GUI .....	12
Figure 2.1: Peperomia Leaf .....	13
Figure 2.2: Swedish Ivy.....	13
Figure 2.3: Algerian Ivy .....	13
Figure 2.4: Peperomia Group 1 (P1).....	14
Figure 2.5: Peperomia Group 2 (P2).....	14
Figure 2.6: Peperomia Group 3 (P3).....	14
Figure 2.7: Swedish Ivy Group 1 (S1) .....	14
Figure 2.8: Swedish Ivy Group 2 (S2) .....	14
Figure 2.9: Swedish Ivy Group 3 (S3) .....	14
Figure 2.10: Algerian Ivy Group 1 (A1) .....	15
Figure 2.11: Algerian Ivy Group 2 (A2) .....	15
Figure 2.12: Algerian Ivy Group 3 (A3) .....	15
Figure 2.13: P1 in CS .....	18
Figure 2.14: P1 in LPS .....	18
Figure 2.15: P2 in CS .....	18
Figure 2.16: P2 in LPS .....	18
Figure 2.17: P3 in CS .....	18
Figure 2.18: P3 in LPS .....	18
Figure 2.19: S1 in CS .....	18
Figure 2.20: S1 in LPS .....	18
Figure 2.21: S2 in CS .....	18
Figure 2.22: S2 in LPS .....	18
Figure 2.23: S3 in CS .....	18
Figure 2.24: S3 in LPS .....	18
Figure 2.25: A1 in CS .....	19
Figure 2.26: A1 in LPS .....	19
Figure 2.27: A2 in CS .....	19
Figure 2.28: A2 in LPS .....	19
Figure 2.29: A3 in CS .....	19
Figure 2.30: A3 in LPS .....	19
Figure 2.31: Extracted Edges Algorithm Block Diagram .....	20

Figure 2.32: LP-P1 .....	21
Figure 2.33: LP-A3 .....	21
Figure 2.34: K-means Clustering of LP-P1 .....	21
Figure 2.35: K-means Clustering of LP-A3 .....	21
Figure 2.36: Edge Boundary Mask of LP-P1 .....	22
Figure 2.37: Edge Boundary Mask of LP-A3.....	22
Figure 2.38: Horizontal Derivative (HD) of LP-P1 .....	22
Figure 2.39: Horizontal Derivative (HD) of LP-A3 .....	22
Figure 2.40: Absolute Value and Threshold of LP-P1.....	23
Figure 2.41: Absolute Value and Threshold of LP-A3 .....	23
Figure 2.42: Extracted Edges of LP-P1.....	23
Figure 2.43: Extracted Edges of LP-A3.....	23
Figure 2.44: Identified Edges of LP-P1, Threshold = 12.....	24
Figure 2.45: Identified Edges of LP-P2, Threshold = 12.....	24
Figure 2.46: Identified Edges of LP-P3, Threshold = 7.....	24
Figure 2.47: Identified Edges of LP-S1, Threshold = 12.....	24
Figure 2.48: Identified Edges of LP-S2, Threshold = 12.....	24
Figure 2.49: Identified Edges of LP-S3, Threshold = 12.....	24
Figure 2.50: Identified Edges of LP-A1, Threshold = 4 .....	25
Figure 2.51: Identified Edges of LP-A2, Threshold = 11 .....	25
Figure 2.52: Identified Edges of LP-A3, Threshold = 11 .....	25
Figure 2.53: Start Location Algorithm Block Diagram .....	26
Figure 2.54: Node Finder Algorithm Block Diagram.....	26
Figure 2.55: Synthetic Image.....	27
Figure 2.56: Extracted Objects from Top Rows of Synthetic Image.....	27
Figure 2.57: Root Path Example .....	29
Figure 2.58: Path Finder Algorithm Block Diagram .....	30
Figure 2.59: Synthetic Image.....	32
Figure 2.60: Node Finder on Synthetic Image.....	32
Figure 2.61: Inner and Outer Contour.....	33
Figure 2.62: Contour Finder Algorithm Block Diagram.....	34
Figure 2.63: 8-Neighborhood Connectivity .....	35
Figure 2.64: Outer Contour of P1 .....	36
Figure 2.65: Inner Contour of P1 .....	36
Figure 2.66: Smoothed Outer Contour of P1.....	36
Figure 2.67: Smoothed Inner Contour of P1 .....	36
Figure 2.68: Contours of P1 .....	37
Figure 2.69: Contours of P2 .....	37
Figure 2.70: Contours of P3 .....	37
Figure 2.71: Contours of S1 .....	37
Figure 2.72: Contours of S2 .....	37
Figure 2.73: Contours of S3 .....	37
Figure 2.74: Contours of A1.....	37
Figure 2.75: Contours of A2.....	37
Figure 2.76: Contours of A3.....	37
Figure 2.77: Swedish Ivy Contour Failure .....	39

Figure 2.78: Swedish Ivy Contour Failure Close-up .....	39
Figure 3.1: Contour Success Examples .....	42
Figure 3.2: Contour Failure Examples .....	43
Figure 3.3: Contours Before Gaussian Filter .....	44
Figure 3.4: Contours After Gaussian Filter .....	44
Figure 3.5: Incomplete Examples .....	45
Figure 3.6: Threshold Distribution of Group 1 .....	47
Figure 3.7: Threshold Distribution of Group 2 .....	48
Figure 3.8: Threshold Distribution of Group 3 .....	48
Figure 3.9: Threshold Distribution of Peperomia Plant .....	49
Figure 3.10: Threshold Distribution of Swedish Ivy Plant .....	50
Figure 3.11: Threshold Distribution of Algerian Ivy .....	50
Figure 3.12: SLIC of S3 .....	52
Figure 3.13: MSRM of P1 .....	53
Figure 3.14: MSRM of P2 .....	53
Figure 3.15: MSRM of P3 .....	53
Figure 3.16: MSRM of S1 .....	53
Figure 3.17: MSRM of S2 .....	53
Figure 3.18: MSRM of S3 .....	53
Figure 3.19: MSRM of A1 .....	54
Figure 3.20: MSRM of A2 .....	54
Figure 3.21: MSRM of A3 .....	54
Figure 3.22: SLIC and MSRM Closeup of A2 .....	54
Figure 3.23: White Background Test Images in Success Group .....	60
Figure 3.24: White Background Test Images in Failure Group .....	61
Figure 3.25: White Background Test Images in Incomplete Group .....	61
Figure 3.26: Natural Background Test Images in Success Group .....	66
Figure 3.27: Natural Background Test Images in Failure Group .....	67
Figure 3.28: Natural Background Test Images in Incomplete Group .....	68
Figure 3.29: 25 x 50 LPS Edges and Contour Result (P1) .....	69
Figure 3.30: 50 x 100 LPS Edges and Contour Result (P1) .....	69
Figure 3.31: 150 x 250 LPS Edges and Contour Result (P1) .....	69
Figure 3.32: 200 x 350 LPS Edges and Contour Result (P1) .....	69
Figure 3.33: 25 x 50 LPS Edges and Contour Result (P2) .....	70
Figure 3.34: 50 x 100 LPS Edges and Contour Result (P2) .....	70
Figure 3.35: 150 x 250 LPS Edges and Contour Result (P2) .....	70
Figure 3.36: 200 x 350 LPS Edges and Contour Result (P2) .....	70
Figure 3.37: 25 x 50 LPS Edges and Contour Result (P3) .....	70
Figure 3.38: 50 x 100 LPS Edges and Contour Result (P3) .....	70
Figure 3.39: 150 x 250 LPS Edges and Contour Result (P3) .....	70
Figure 3.40: 200 x 350 LPS Edges and Contour Result (P3) .....	70
Figure 3.41: 25 x 50 LPS Edges and Contour Result (S1) .....	71
Figure 3.42: 50x 100 LPS Edges and Contour Result (S1) .....	71
Figure 3.43: 150 x 250 LPS Edges and Contour Result (S1) .....	71
Figure 3.44: 200 x 350 LPS Edges and Contour Result (S1) .....	71
Figure 3.45: 25 x 50 LPS Edges and Contour Result (S2) .....	71

Figure 3.46: 50 x 100 LPS Edges and Contour Result (S2).....	71
Figure 3.47: 150 x 250 LPS Edges and Contour Result (S2).....	71
Figure 3.48: 200 x 350 LPS Edges and Contour Result (S2).....	71
Figure 3.49: 25 x 50 LPS Edges and Contour Result (S3).....	72
Figure 3.50: 50 x 100 LPS Edges and Contour Result (S3).....	72
Figure 3.51: 150 x 250 LPS Edges and Contour Result (S3).....	72
Figure 3.52: 200 x 350 LPS Edges and Contour Result (S3).....	72
Figure 3.53: 25 x 50 LPS Edges and Contour Result (A1).....	72
Figure 3.54: 50 x 100 LPS Edges and Contour Result (A1).....	72
Figure 3.55: 150 x 250 LPS Edges and Contour Result (A1).....	72
Figure 3.56: 200 x 350 LPS Edges and Contour Result (A1).....	72
Figure 3.57: 25 x 50 LPS Edges and Contour Result (A2).....	73
Figure 3.58: 50 x 100 LPS Edges and Contour Result (A2).....	73
Figure 3.59: 150 x 250 LPS Edges and Contour Result (A2).....	73
Figure 3.60: 200 x 350 LPS Edges and Contour Result (A2).....	73
Figure 3.61: 25 x 50 LPS Edges and Contour Result (A3).....	73
Figure 3.62: 50 x 100 LPS Edges and Contour Result (A3).....	73
Figure 3.63: 150 x 250 LPS Edges and Contour Result (A3).....	73
Figure 3.64: 200 x 350 LPS Edges, Failed Path.....	73
Figure 3.65: Contour for P1 at Original Size.....	77
Figure 3.66: Contour for P1 at One-half Size.....	77
Figure 3.67: Contour for P1 at One-tenth Size.....	77
Figure 3.68: Contour for P2 at Original Size.....	77
Figure 3.69: Contour for P2 at One-half Size.....	77
Figure 3.70: Contour for P2 at One-tenth Size.....	77
Figure 3.71: Contour for P3 at Original Size.....	77
Figure 3.72: Contour for P3 at One-half Size.....	77
Figure 3.73: Contour for P3 at One-tenth Size.....	77
Figure 3.74: Contour for S1 at Original Size.....	78
Figure 3.75: Contour for S1 at One-half Size.....	78
Figure 3.76: Contour for S1 at One-tenth Size.....	78
Figure 3.77: Contour for S2 at Original Size.....	78
Figure 3.78: Contour for S2 at One-half Size.....	78
Figure 3.79: Contour for S2 at One-tenth Size.....	78
Figure 3.80: Contour for S3 at Original Size.....	78
Figure 3.81: Contour for S3 at One-half Size.....	78
Figure 3.82: Contour for S3 at One-tenth Size.....	78
Figure 3.83: Contour for A1 at Original Size.....	79
Figure 3.84: Contour for A1 at One-half Size.....	79
Figure 3.85: Contour for A1 at One-tenth Size.....	79
Figure 3.86: Contour for A2 at Original Size.....	79
Figure 3.87: Contour for A2 at One-half Size.....	79
Figure 3.88: Contour for A2 at One-tenth Size.....	79
Figure 3.89: Contour for A3 at Original Size.....	79
Figure 3.90: Contour for A3 at One-half Size.....	79
Figure 3.91: Contour for A3 at One-tenth Size.....	79

## List of Tables

Table 2.1: Image Distribution by Plant and Group.....	15
Table 3.1: Validation Image Database by Plant and Group .....	40
Table 3.2: Contour Successes by Group and Plant Type .....	40
Table 3.3: Contour Success Percentage .....	40
Table 3.4: Contour Failures by Group and Plant Type .....	42
Table 3.5: Contour Failure Percentage.....	42
Table 3.6: Incompletes by Group and Plant Type .....	44
Table 3.7: Percentages of Incompletes.....	45
Table 3.8: Contour Success Percentage Excluding Incompletes.....	46
Table 3.9: Contour Failure Percentage Excluding Incompletes .....	46
Table 3.10: Average Thresholds .....	49
Table 3.11: Test Image Results.....	55
Table 3.12: Contour Successes by Group and Image Size.....	75
Table 3.13: Contour Success Percentages by Group and Image Size.....	75
Table 3.14: Contour Failures by Group and Image Size.....	75
Table 3.15: Contour Failure Percentage by Group and Plant Size .....	75
Table 3.16: Incompletes by Group and Image Size .....	75
Table 3.17: Percentages of Incompletes by Group and Image Size .....	76

# **Abstract**

Recursive Path Following in Log Polar Space for Autonomous Leaf Contour Extraction

by

Anna Snarski

University of New Hampshire, September 2020

Use of image segmentation has caused agriculture advancement in species identification, chlorophyll measurements, plant growth and disease detection. Most methods require some level of manual segmentation as autonomous image segmentation is a difficult task. Methods with the highest segmentation precision use a priori knowledge obtained from user input which is time consuming and subjective. This research focuses on providing current segmentation methods a pre-processing model that autonomously extracts an internal and external contour of the leaf. The model converts the uniform Cartesian images to non-uniformly sampled images in log polar space. A recursive path following algorithm was designed to map out the leaf's edge boundary. This boundary is shifted inward and outward to create two contours; one that lies within the foreground and one within the background. The image database consists of 918 leaves from multiple plants and different background mediums. The model successfully created contours for 714 of the leaves. Results of the autonomously created contours being used in lieu of user-input contours for a current segmentation algorithm are presented.

# Introduction

Image processing has become a popular technique in agriculture advancement. Pérez-Patricio *et al.* [1] designed a vision-base model that uses reflectance and transmittance to estimate chlorophyll. Itakura *et al.* [2] estimated leaf area and inclination angle from 3D constructed images. Weizheng *et al.* [3] created a grading method that extracted discolored pixels in the image and ranked the disease level from the lesion to leaf area ratio. Species identification, chlorophyll measurements, plant growth, and disease detection have all benefited from some technique of image segmentation. Most methods require some level of manual segmentation as autonomous image segmentation is a difficult task. Image segmentation methods with the highest precision use a priori knowledge of leaf features, limit the bounds of captured image, or use user interactive input. This research focuses on providing current segmentation methods a pre-processing model that autonomously provides an initial contour of the foreground and background. The method proposed will help in the creation of leaf contour extraction algorithms that do not require initial human interaction.

Grand-Brochier *et al.* [4] compares segmentation methods from early thresholding techniques to Guided Active Contour (GAC) [5], [6] on tree leaves in natural images. Of the six methods that accept a priori knowledge, five had improved results from using color distance maps and user-input strokes. A stroke is defined as a hand-drawn marking the user adds to the image. The color distance maps are based on two assumptions; the leaf is at the center of the image and the background is at the corners. The input stroke allows the user to locate the leaf and draw the



general shape. For GAC and simple linear iterative clustering (SLIC) [7] the stroke is used as a priori knowledge on the leaf color, while the other methods use the stroke for contour initialization.

Wang *et al.* [8] combines the Chan-Vese model and the Sobel operator to create a multistep algorithm to segment overlapping cucumber leaves. Background removal consists of thresholding in RGB (red, green, blue) color space; if green is the prominent color the pixel is considered part of the leaf region. The Chan-Vese model is an active contour region-based method that deals with energy minimization and needs an initialization contour to start [9]. The traditional two Sobel operators are 3 x 3 matrices that identify edges in the horizontal (0°) and vertical (90°) direction. To reduce the impact of noise and discontinuity, 5 x 5 gradient operators in eight directions were used. The algorithm had a mean error rate of 4.3% when compared to the ground truth images. To remain autonomous, Wang *et al.* uses the center point of the image for the Chan-Vese model contour initialization instead of user-input strokes. This research will allow Wang *et al.*'s algorithm to remain autonomous and begin with an initial contour for the Chan-Vese model.

User interactive input strokes are not restricted to leaf segmentation. Grady's [10] Random Walk (RW) algorithm uses user-defined labeled areas to calculate the probability that an unlabeled pixel belongs to a labeled area and was tested on medical and landscape images. Yang *et al.* [11] constrain the RW model to two labeled areas, foreground and background. Their model allows for soft constraint inputs where the boundary region should pass through and hard constraint inputs that must align with the boundary. Their image dataset contained wildlife, people and foliage. Rother *et al.* [12] extends the graph-cut [13] approach to require less interactive input through iterative estimation and incomplete labeling. Their algorithm, GrabCut, requires only a user-defined rectangle encompassing the main object to begin and was tested on similar types of images as contained in [11]. Ning *et al.* [14] created a maximal similarity-based region merging (MSRM)

algorithm. It uses mean shift [15] as the initial low-level segmentation method and user-input markings to label the main object and background areas.

User interactive input is subjective and time consuming. To date, such interactive input is necessary in order to obtain accurate segmentation results. Many of these algorithms mentioned are autonomous after the user-based input. The goal of this research is to create a pre-processing algorithm that can autonomously create a contour around the main object, a leaf. The database will consist of multiple plants and different background mediums. The hypothesis is the simpler leaves and background mediums will have a higher success rate.

The model converts the Cartesian images to log polar space (LPS) to find the contours. Although the goal is to be truly autonomous and avoid any user input, images used must have the center point inside the leaf which does require some level of user input for the acquired image to begin with. However, once this a priori knowledge is met, the algorithm is autonomous. The results of this thesis will allow other segmentation methods to be more fully autonomous and less subjective.

The hopes of creating fully autonomous segmentation algorithms will allow for real-time segmentation results. These results can then be combined with chlorophyll measurement algorithms, plant identification databases, or disease grading methods. All the end user will need to do is take a photo of the leaf and within seconds they would be given educated guesses of the type of leaf, if the plant is lacking chlorophyll, or what type of disease is killing the plant.

The remainder of the thesis is organized as follows. Chapter 1 presents background material. Chapter 2 discusses the image databases, algorithm breakdown, and how results are categorized. Chapter 3 presents the contour results, MSRM results, and LPS sizes. Chapter 4 concludes the thesis and Chapter 5 discusses future improvements to the model.

# Chapter 1 Background

## 1.1 LOG POLAR TRANSFORM

The log polar transform (LPT) is a conformal mapping which maps the Cartesian coordinates (x,y) to the logarithmic coordinates (u,v), representing the natural-log of the radius (r) and the angle of radius vector ( $\theta$ ). Mathematically we have the following:

$$r = \sqrt{x^2 + y^2} \quad (1.1)$$

$$\theta = \arctan\left(\frac{y}{x}\right) \quad (1.2)$$

$$u = \ln\left(\frac{r}{R_{min}}\right) \quad (1.3)$$

$$v = \theta \quad (1.4)$$

To produce this transform from a Cartesian space digital image Young's [16] method was implemented which overlays concentric circles onto the input image. The radial distances, or rings, constitute the number of circles used from  $R_{min}$  to  $R_{max}$ . The angle widths, or wedges, constitute the number of circles used from  $\theta = 0^\circ$  to  $\theta = 360^\circ$ . All pixels within a circle are averaged together and map to a single pixel in log polar space. An example is shown in Figure 1.1. In this case, the number of rings is five and wedges is sixteen, resulting in a mapped dimension (u,v) of 5 x 16 pixels.

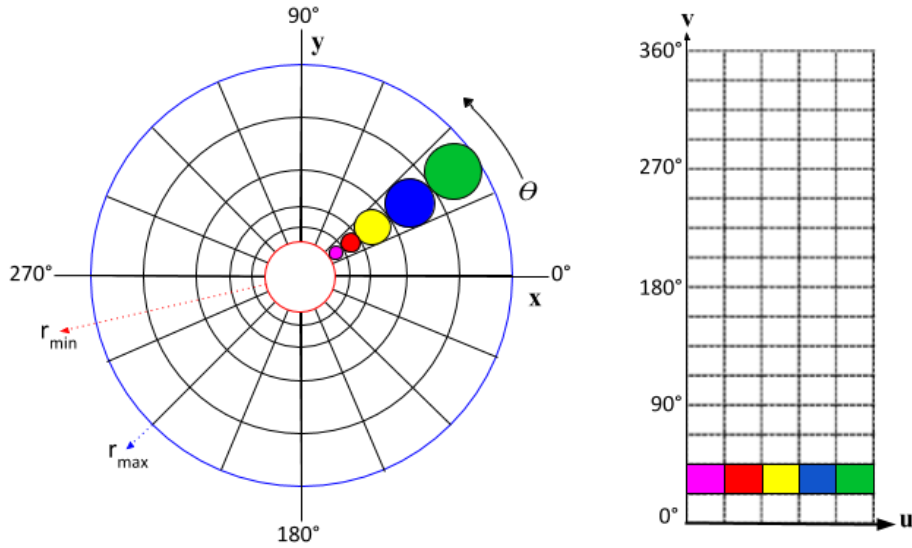


Figure 1.1: Cartesian Space to Log Polar Space

The log polar coordinate system is rotational and scale tolerant. Figure 1.2 depicts the rotation and scale tolerance. A scale in Cartesian space translates to a horizontal shift in LPS. A rotation in Cartesian space translates to a vertical shift in LPS. If  $R_{\min}$  is inside the leaf, the edge of the leaf will become a complete path from  $0 \leq v \leq \#$  of wedges in LPS. The start of the edge boundary can always be defined within the first row of the LPS image and the end of the edge boundary within the final row of the LPS image.

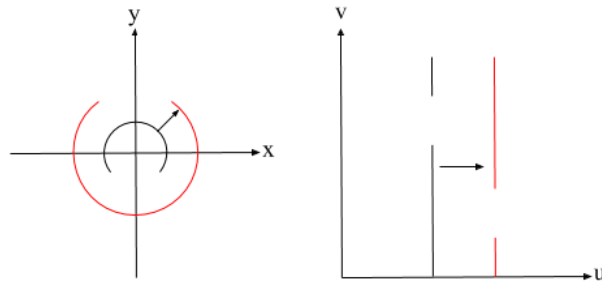


Figure 1.2: Scale and Rotation Tolerant

As the concentric circles get farther away from the center, the more pixels in Cartesian space will be averaged together to become one pixel in log polar space, causing non-uniform sampling. Information at the center of the Cartesian image will be preserved in log polar space while high frequency information is lost via the averaging of pixels the closer it is to  $R_{\max}$ . For the purpose of this research, this loss of information is an advantage. The foreground (leaf) will always be at the center of the image and the unimportant background (dirt, branches, other leaves) is always near the outermost rings in the transform space. Data loss and averaging of the background will blur edges and remove high frequencies allowing the edge of the leaf to become more prominent.

Research of log polar image representation dates to the 1970's starting with Schwartz [17] who showed that mammals' visual system can be represented by log polar mapping. Weiman and Chaikin [18] mathematically derived a logarithmic spiral grid to produce conformal mapping of Cartesian coordinates for image processing use. Schenker *et al.* [19] uses a similar method of polar exponential grid (PEG) for image representation and blob boundary detection. It has become a common transform to use for object recognition with a wide variety of computer vision and machine learning techniques. Incorporating Schwartz's and Schenker's research, Messner and Szu [20] designed in hardware the logarithmic coordinate mapping without calculation. The architecture successfully converts scale and rotations of the input image into translations in real time. Bishay *et al.* [21] used log polar mapping to transform the unknown edges that defined boundaries of a room into horizontal edges. Template-matching was then performed to identify the objects in each portion of the room. The system was computationally efficient as only one template per object was needed due to LPT being rotational and scale tolerant. Traver and Pla [22] investigate the problem of translation estimation in log polar images using two methods, gradient-

descent and projections into 1D arrays. Ellahyani and Ansari [23] combines the log polar mapping with mean shift clustering and random forests to detect the road sign by color and shape. Similarly, Koester [24] and Radhi [25] compared LPS images to Cartesian space images using a convolutional neural network to identify road images. The processing speed was significantly higher for log polar space images with relatively similar validation accuracy.

## 1.2 COLOR SPACE

Computer vision is comparable to human vision. The eyes are the receptors for human vision and a digital camera are the receptors for computer vision. The brain and computer would then be the interpretation system. To acquire colored images, most cameras are configured with a Bayer [26] color filter array. A square grid of photosensors are overlaid with red (R), green (G), and blue (B) color filters. When a digital RGB image is captured, each pixel is represented by three values, one for each RGB color value, ranging from 0 to 1 or 0 to 255 if 8-bit quantized as shown in Figure 1.3. Figure 1.4 displays these three primary colors are enough to produce any perceived color. The value of each RGB color needed to create a specific color is known at the tristimulus values [27].

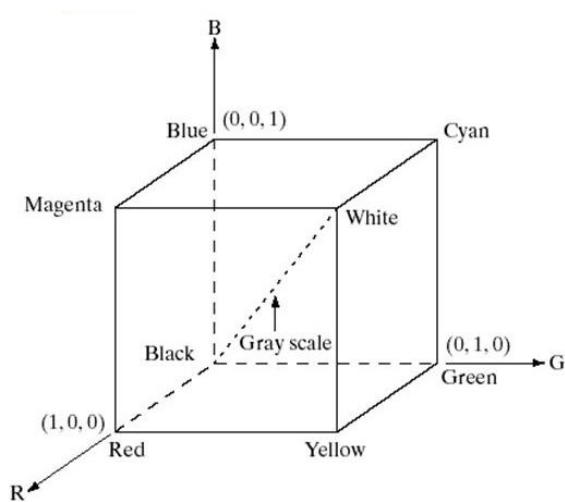


Figure 1.3: Schematic of RGB Color Cube [28]

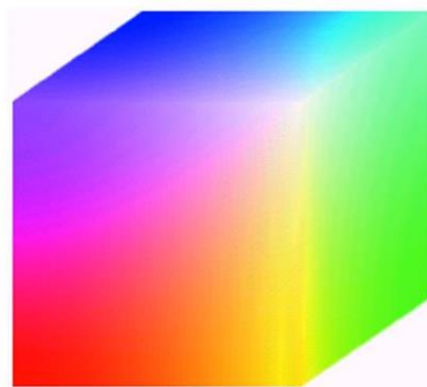


Figure 1.4: RGB Color Cube [28]

RGB is just one of many color spaces to represent an image. Humans observe color relating to hue, chroma, and brightness rather than tristimulus values. Brightness corresponds to the perceived lightness stimulated by the rods and cones in the retina. Hue quantifies a perceived color by how similar or different it is to the primary colors. Neutral colors (black, white and grey) do not have any hue. Chroma defines the level of colorfulness in relation to the amount of brightness.

Color spaces that quantify color like the human visual system are known as perceptual spaces. In image processing, it is common to convert images into a perceptual color space to make it easier for the human to describe color. Brightness is not numerically quantifiable; therefore, luminance, a measurement of light intensity, is used. Perceptual spaces are also categorized as luminance-chrominance based where one component represents the luminance and two represent the chrominance [27].

For this research, images were converted into  $L^*a^*b^*$  color space. As shown in Figure 1.5, luminance ( $L^*$ ) ranges from 0 (black) to 100 (white),  $a^*$  ranges from -128 (green) to 128 (red), and  $b^*$  ranges from -128 (blue) to 128 (yellow).  $L^*a^*b^*$  can be converted to LCH (luminance, chroma, hue) where chroma and hue are the polar coordinates of  $a^*$  and  $b^*$ . Chroma ranges from 0 at the center, unsaturated, to 100 at the edge of the circle, fully saturated. Hue is defined in angular degrees with red being  $0^\circ$ , green at  $120^\circ$ , and blue at  $240^\circ$ . Asmare *et al.* [29] compares numerous color spaces for conversion accuracy and image enhancement.  $L^*a^*b^*$  has a high accuracy conversion from RGB and was deemed an acceptable color space to use. The conversion between RGB and  $L^*a^*b^*$  can be found in [27] and [29].

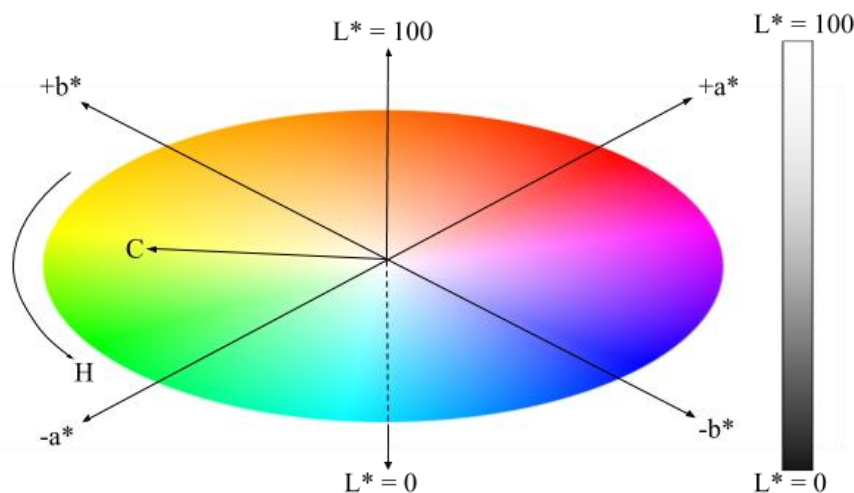


Figure 1.5:  $L^*a^*b^*$  Color Space



### 1.3 CHAIN CODE

For this research, chain code was used to perform a cost analysis to determine the best edge boundary path if more than one was found. Chain code is an algorithm that represents a pixel location of a boundary by its slope direction compared to the previous pixel [30]. It allows for a lossless compression, only the chain code and start location is needed to recreate the boundary of an object. The algorithm uses an 8-neighborhood connectivity map shown in Figure 1.6. Figure 1.7 is an example of the chain directions starting at the location (4,4). The chain code would then be 100775565433221. Chain code is typically used for a closed edge boundary. In this research every boundary is open. Luengo's [31] MATLAB code was modified to accept an open boundary and always given a start location at the top row of the image.

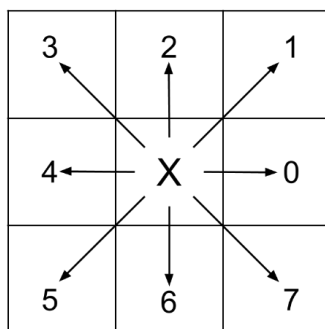


Figure 1.6: 8-Neighborhood Connectivity

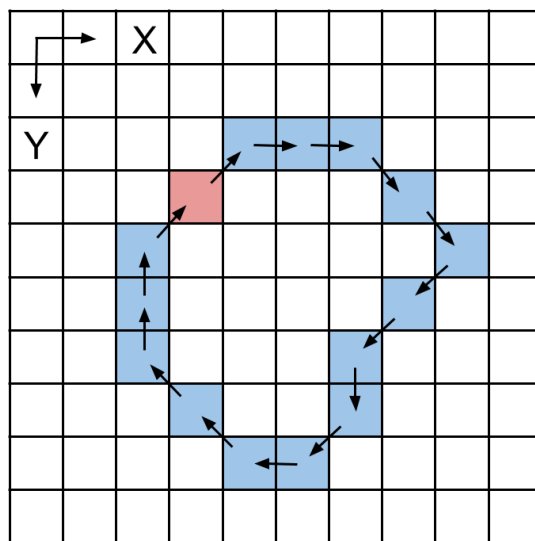


Figure 1.7: Chain Code Example

## 1.4 MAXIMAL SIMILARITY-BASED REGION MERGING

The outputs of this research are meant to replace user-input markings for current segmentation algorithms. The maximal similarity-based region merging (MSRM) algorithm by Ning *et al.* [14] will be used to test if this research's contours are successful. MSRM accepts user-input background and foreground strokes. To create homogeneous regions, an initial low-level segmentation algorithm is needed. Regions are categorized as containing background strokes ( $G_B$ ), foreground strokes ( $G_F$ ) and unlabeled ( $G_U$ ). For each region, the color channels are quantized into  $N$  bins and the normalized color histogram (Hist) is found. The histograms are used to calculate the Bhattacharyya coefficient [32] of adjacent regions, shown in Figure 1.8.

$$\rho(A, B) = \sum_{i=1}^N \sqrt{Hist_A^i \cdot Hist_B^i} \quad (1.5)$$

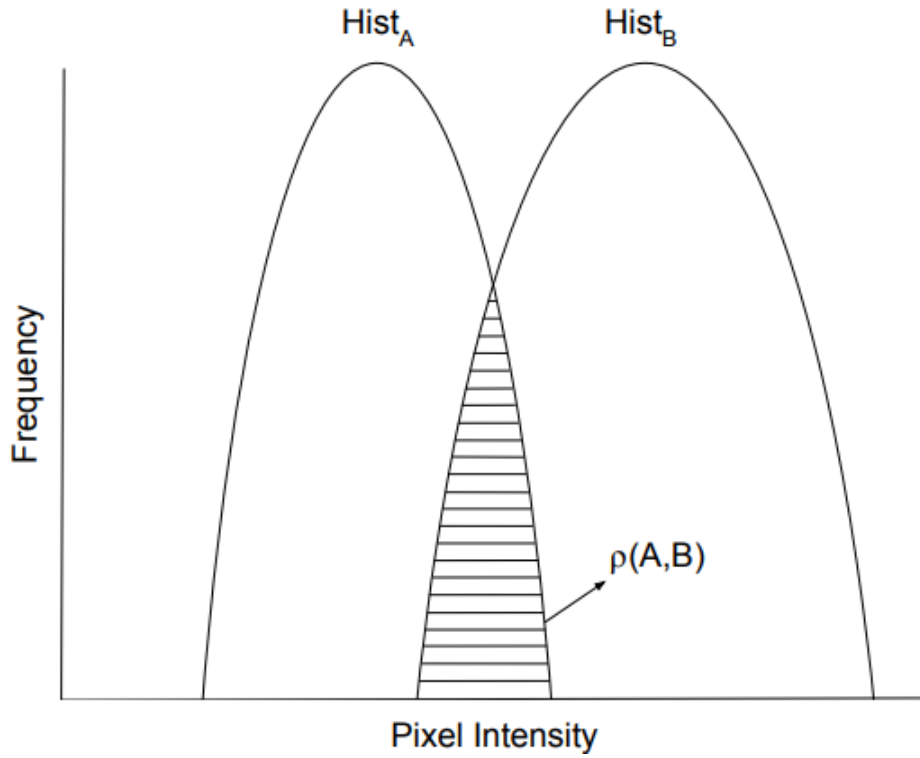


Figure 1.8: Bhattacharyya Coefficient Example

The Bhattacharyya coefficient is a statistical measure of similarity between two samples. The higher the coefficient the higher the similarity. The coefficient is calculated for all regions adjacent to A, then the region with the highest coefficient is merged with A. The algorithm is split into two stages, merging  $G_B$  and  $G_U$  regions then merging  $G_F$  and  $G_U$  regions.

Ning *et al.* have a website [33] that contained MATLAB code for their algorithm. Modification of this code was necessary in order to be compatible with MATLAB R2019b. The code also had to be updated to handle images larger than 600 x 600 pixels. The code contains a graphical user interface (GUI) where the user draws the foreground and background markers or uploads predefined markers. Figure 1.9 shows the layout of the GUI with user-drawn markers.

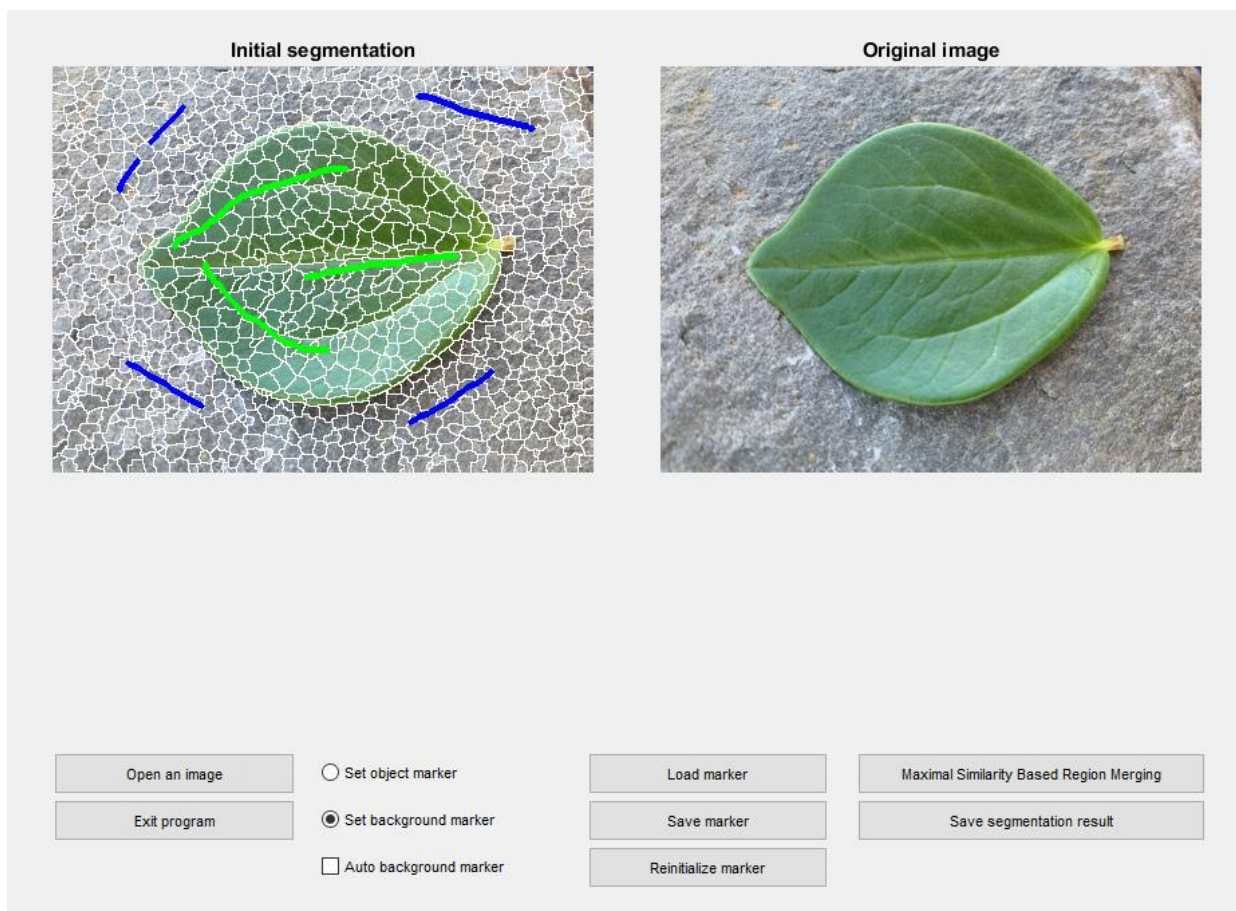


Figure 1.9: MSRM GUI

# Chapter 2 Methods

## 2.1 IMAGE DATABASES

To accurately test the program, an image database of different plant leaves and backgrounds was required. These images will be referred to as the validation image database. Images were captured in shade to limit reflection and shadows. The full leaf had to be included in the image and the center of the image must be part of the leaf. Three plants were used, peperomia, Swedish ivy and Algerian ivy.



*Figure 2.1: Peperomia Leaf*



*Figure 2.2: Swedish Ivy*



*Figure 2.3: Algerian Ivy*

Images were split into three groups by background medium. Group 1 is a simple background where leaves are placed on large stones. Group 2 consists of pebbles and sand while Group 3 was a mixture of dead and alive grass. The background in Group 1 lacks high frequency and shades of green. Group 2 has high frequency from the different colored pebbles and sand. Group 3 has high frequency and shades of green making it the most complicated background.



Peperomia leaves (Figure 2.4, Figure 2.5, Figure 2.6) are elliptic or obtuse shaped and have entire (smooth) edges.



*Figure 2.4: Peperomia Group 1 (P1)*



*Figure 2.5: Peperomia Group 2 (P2)*



*Figure 2.6: Peperomia Group 3 (P3)*

Swedish ivy leaves (Figure 2.7, Figure 2.8, Figure 2.9) have crenate (scallop) edges and are broadly ovate shaped with some leaves having an oblique base.



*Figure 2.7: Swedish Ivy Group 1 (S1)*



*Figure 2.8: Swedish Ivy Group 2 (S2)*



*Figure 2.9: Swedish Ivy Group 3 (S3)*



Algerian ivy leaves (Figure 2.10, Figure 2.11, Figure 2.12) are palmately lobed with entire edges.



Figure 2.10: Algerian Ivy Group 1 (A1)



Figure 2.11: Algerian Ivy Group 2 (A2)



Figure 2.12: Algerian Ivy Group 3 (A3)

The database contains 918 images and is shown broken down by group and plant in Table 2.1. Most of the images are 3024 x 4032 pixels, though some were cropped to move the leaf to the center of the image or rotated to create more variety between images. Leaves in Figure 2.4 through Figure 2.12 will be used throughout Chapter 2 to show results at each stage of the model. They will be referenced as the first letter of their plant name and what group they are in (i.e. P1, A2).

	Peperomia	Swedish Ivy	Algerian Ivy	Total
Group 1	130	82	49	261
Group 2	141	159	108	408
Group 3	83	99	67	249
Total	354	340	224	918

Table 2.1: Image Distribution by Plant and Group

Eighty additional images from ImageCLEF's [34] 2014 plant database are used to test the model. They will be referred to as the test image database. These leaves were not used while writing and debugging the algorithms developed. The aim of the test images is to demonstrate and

determine the level of robustness with respect to leaf shape and background noise in general. The images are split into two groups, white background and natural background. The images ranged in size from 340 x 350 pixels to 960 x 900 pixels. Images were chosen using the same restraints as the validation dataset. Leaves had to be simple (not compound), completely in the image, green, non-overlapping with another leaf, and with minimal reflection. These restrictions had to be relaxed to obtain forty background images. Some leaves in the database have dead spots, overlap other leaves, or are yellow.

The database contains elliptic, deltoid, acicular, reniform, ovate, orbicular, cuneate, cordate and palmately lobed shaped leaves. The edges are entire, dentate, spiny, serrate, and undulate. Please refer to pages 18-25 of [35] for an in-depth description of leaf architecture. The natural backgrounds contained dirt, leaves, pebbles, sticks, grass, and stone slabs.

## 2.2 ALGORITHM DESCRIPTION

### 2.2.1 Log Polar Space

Young's [16] MATLAB code was used for all log polar and inverse log polar transforms. For  $R_{\max}$  to be half the length of the longest image dimension, either the length or width of the image was padded to create a square image. 150 rings and 250 wedges were chosen creating 150 x 250 resolution images.  $R_{\min}$  was calculated using Equation 2.1 which is equivalent to Equation 2.2 given the constant ring and wedge values for all images. Figure 2.13 through Figure 2.30 show the leaves in Cartesian space (CS) and the results from the LPT.

$$R_{\min} = R_{\max} e^{\frac{-2\pi(rings-1)}{wedges}} \quad (2.1)$$

$$R_{\min} = 0.0236 R_{\max} \quad (2.2)$$





*Figure 2.13: P1 in CS*



*Figure 2.14: P1 in LPS*



*Figure 2.15: P2 in CS*



*Figure 2.16: P2 in LPS*



*Figure 2.17: P3 in CS*



*Figure 2.18: P3 in LPS*



*Figure 2.19: S1 in CS*



*Figure 2.20: S1 in LPS*



*Figure 2.21: S2 in CS*



*Figure 2.22: S2 in LPS*



*Figure 2.23: S3 in CS*



*Figure 2.24: S3 in LPS*



*Figure 2.25: A1 in CS*



*Figure 2.26: A1 in LPS*



*Figure 2.27: A2 in CS*



*Figure 2.28: A2 in LPS*



*Figure 2.29: A3 in CS*



*Figure 2.30: A3 in LPS*



### 2.2.2 Extracted Edges Algorithm

Next, the edges of the leaf need to be extracted from the image. Figure 2.31 breaks down the Extracted Edges algorithm which also includes converting to LPS. The input is the original Cartesian space image and the output is a LPS binary image of the edges found.

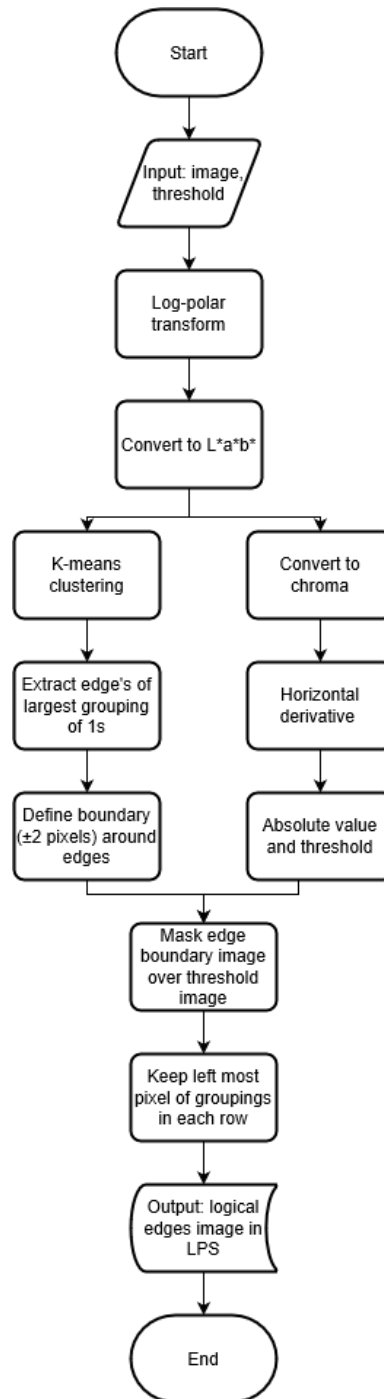


Figure 2.31: Extracted Edges Algorithm Block Diagram

Leaves P1 and A3 are used to show outputs at each stage of the algorithm. Figure 2.32 and Figure 2.33 reshown the leaves in LPS and will be referenced as LP-P1 and LP-A3. Once the LPS image is converted to  $L^*a^*b^*$  color space, k-means clustering is performed using  $a^*$  and  $b^*$  to partition the pixels into two clusters, foreground and background, as seen in Figure 2.34 and Figure 2.35.



*Figure 2.32: LP-P1*



*Figure 2.33: LP-A3*

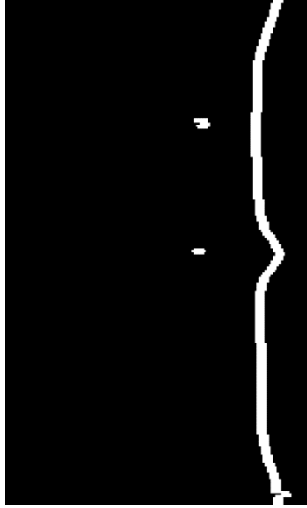


*Figure 2.34: K-means Clustering of LP-P1*



*Figure 2.35: K-means Clustering of LP-A3*

Only the edges of the largest grouping of ones is kept. Figure 2.36 and Figure 2.37 show the edge mask made by creating a boundary of two pixels to the left and right of each edge pixel.

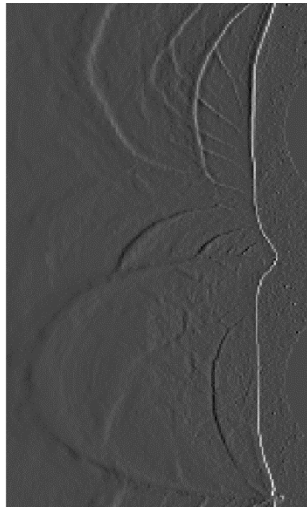


*Figure 2.36: Edge Boundary Mask of LP-P1*

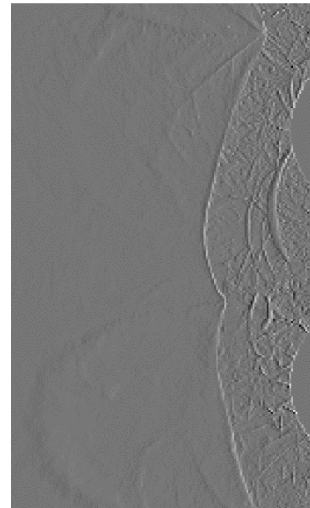


*Figure 2.37: Edge Boundary Mask of LP-A3*

Figure 2.38 and Figure 2.39 are the results of the horizontal value difference between adjacent pixels in chroma, the magnitude of  $a^*$  and  $b^*$ .



*Figure 2.38: Horizontal Derivative (HD) of LP-P1*



*Figure 2.39: Horizontal Derivative (HD) of LP-A3*

Next, the absolute value is taken and all pixels below the input threshold are changed to zero as shown in Figure 2.40 and Figure 2.41.



*Figure 2.40: Absolute Value and Threshold of LP-P1*



*Figure 2.41: Absolute Value and Threshold of LP-A3*

The boundary mask is overlaid on the threshold image to extract the real edge of the leaf. Lastly, to improve the path following algorithm only the left most pixel in horizontal groupings are kept. The result of the Extracted Edges algorithm is shown in Figure 2.42 and Figure 2.43.



*Figure 2.42: Extracted Edges of LP-P1*



*Figure 2.43: Extracted Edges of LP-A3*

To extract only the prominent edges, the Extracted Edges algorithm is first given a high threshold. If the Path Finder algorithm (section 2.2.4) cannot find a complete path, the threshold is lowered and both algorithms are rerun. If the threshold is at four, the lowest threshold allowed, and a path still cannot be found, the leaf is considered a failure. Figure 2.44 through Figure 2.52 show the identified edges from the Extracted Edges algorithm and the respective threshold value.



Figure 2.44: Identified Edges of LP-P1,  
Threshold = 12

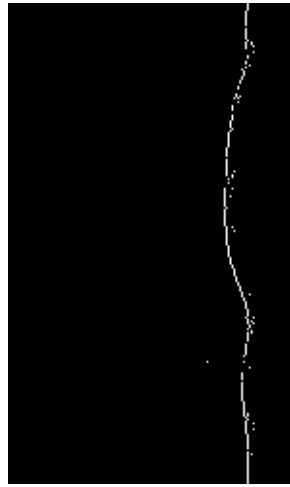


Figure 2.45: Identified Edges of LP-P2,  
Threshold = 12



Figure 2.46: Identified Edges of LP-P3,  
Threshold = 7



Figure 2.47: Identified Edges of LP-S1,  
Threshold = 12

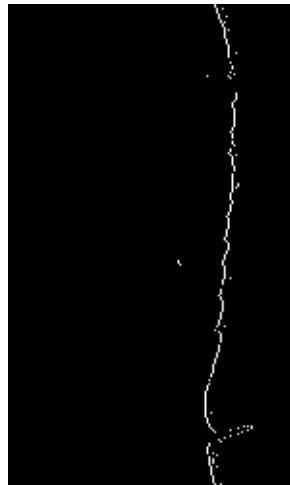


Figure 2.48: Identified Edges of LP-S2,  
Threshold = 12



Figure 2.49: Identified Edges of LP-S3,  
Threshold = 12



*Figure 2.50: Identified Edges of LP-A1,  
Threshold = 4*



*Figure 2.51: Identified Edges of LP-A2,  
Threshold = 11*



*Figure 2.52: Identified Edges of LP-A3,  
Threshold = 11*



### 2.2.3 Start Location Algorithm and Node Finder Algorithm

To begin following the edge path, a start location is needed. Figure 2.53 is a block diagram showing how the start location is chosen. The Start Location algorithm uses the Node Finder algorithm in Figure 2.54.

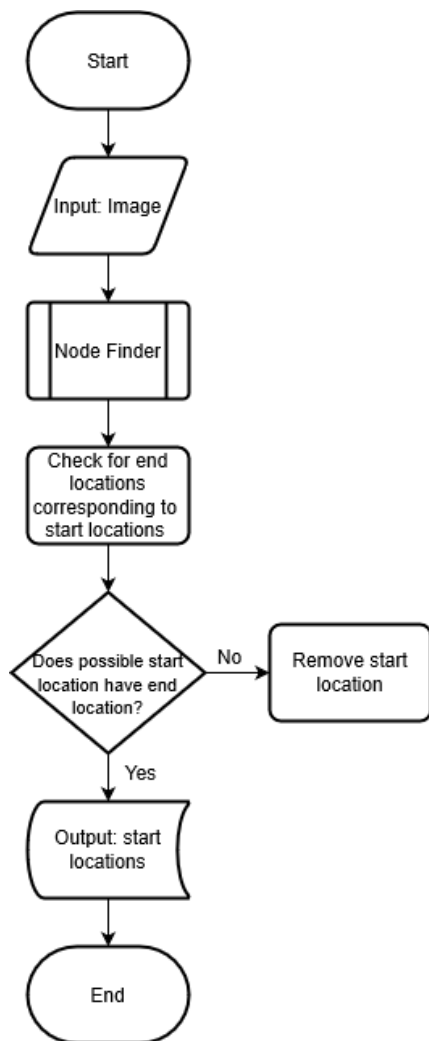


Figure 2.53: Start Location Algorithm Block Diagram

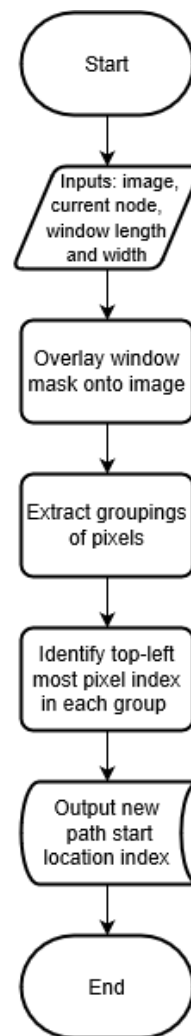
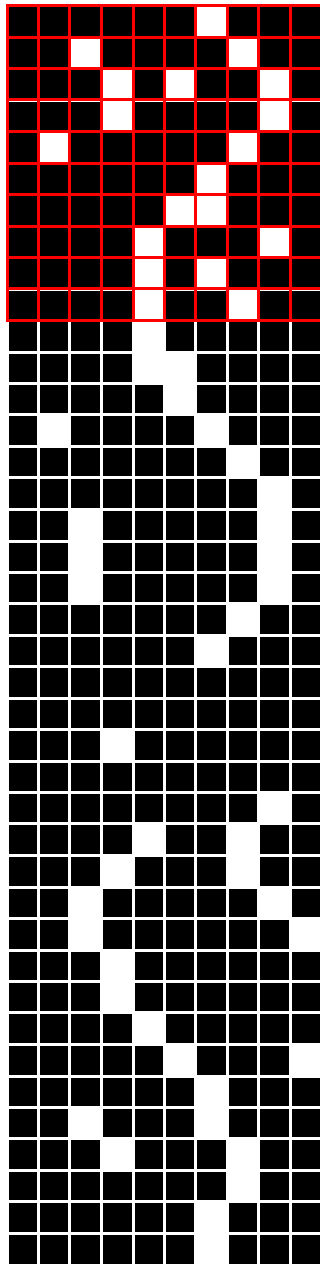


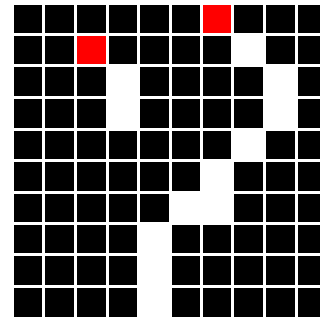
Figure 2.54: Node Finder Algorithm Block Diagram

The term node is used to describe a pixel with value one (white). The input window length and width define how large of a window the nodes will be searched in. When used to find the start location, window length is set to ten and window width is the entire row. The algorithm extracts objects larger than two pixels from the first ten rows and chooses the top left most pixel of each grouping as the output.

To demonstrate both algorithms, a synthetic input image of 10 x 40 pixels is shown in Figure 2.55. The Start Location algorithm will call the Node Finder algorithm to extract the top ten rows (highlighted in red). Figure 2.56 shows the window size (10 x 10) and the extracted objects. The two pixels identified as possible starting locations are shown in red.



*Figure 2.55: Synthetic Image*

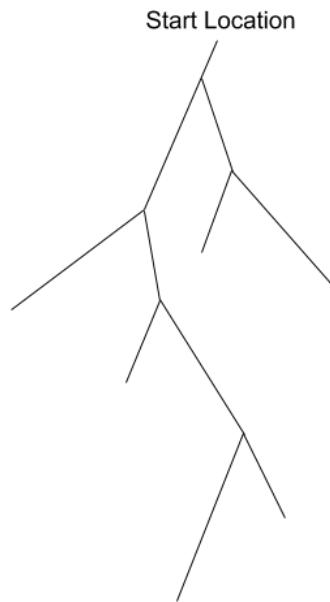


*Figure 2.56: Extracted Objects from Top Rows of Synthetic Image*

The Start Location algorithm identifies the possible start locations and checks to see if nodes exist within the same few columns at the bottom of the image. A benefit of the LPT is the edge leaf will be at the same radius ( $u$ ) at  $v = 0^\circ$  (bottom of image) and  $v = 360^\circ$  (top of image). Therefore, the true start location of the edge will have a matching end location. The Node Finder algorithm is also used to search the image for potential paths when a large enough break occurs. An example of this is shown in the next section.

### 2.2.4 Path Finder Algorithm

The Path Finder algorithm is the main part of the autonomous model. The paths can be thought of as tree roots as seen in Figure 2.57. The algorithm is designed to search for the deepest root. When a root splits into multiple paths, the algorithm stores the split location, then recursively follows each root to its end. Figure 2.58 shows the block diagram for the algorithm.



*Figure 2.57: Root Path Example*

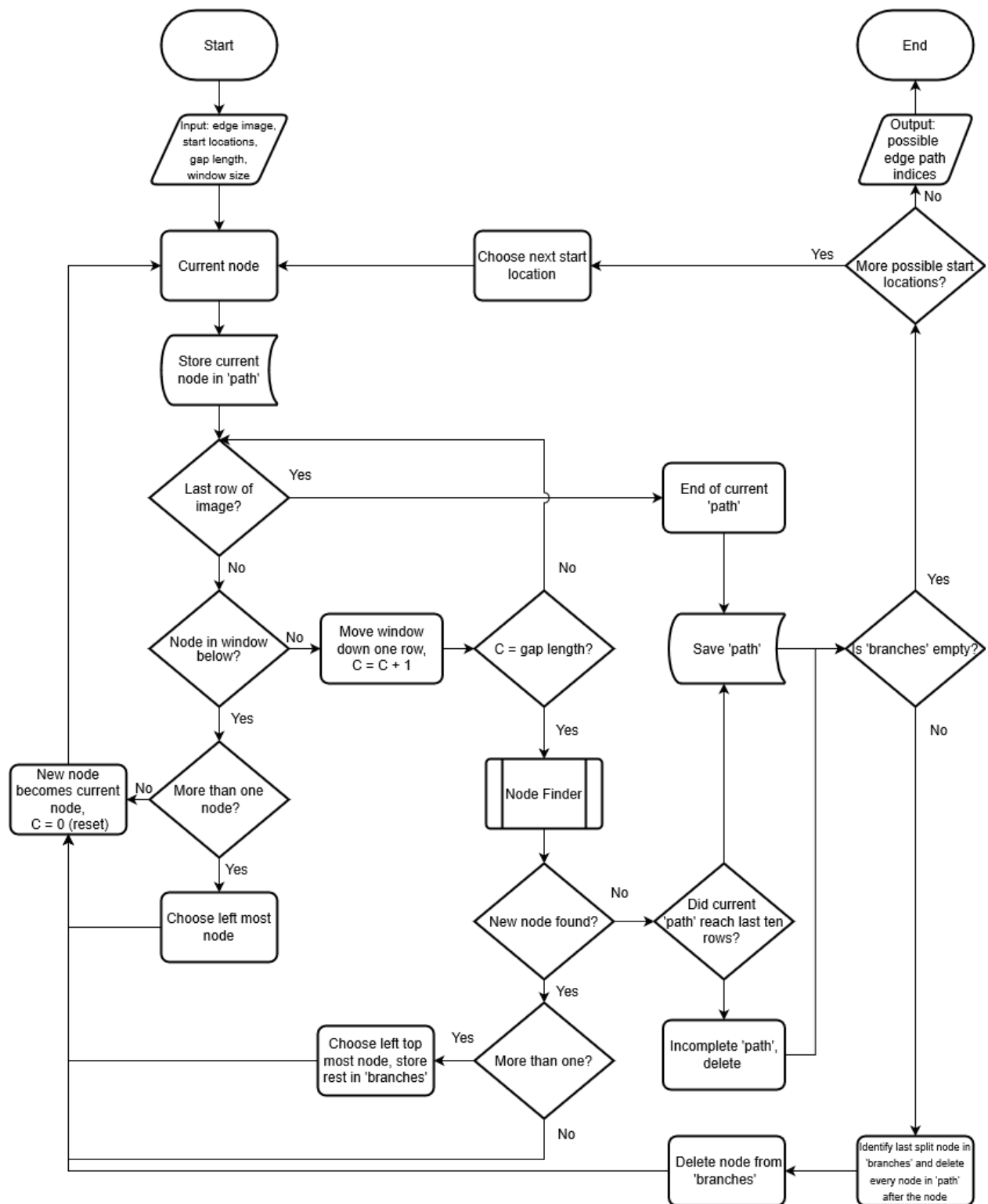


Figure 2.58: Path Finder Algorithm Block Diagram

The inputs to the algorithm are the start location(s), the LPS binary edge image from Extracted Edges, the window size, and the gap length. The start location index becomes the current node and is stored in the temporary array called 'path'. The window size defines how many pixels to extract from the row below the current node. A window size of two means extract two pixels to the left and right making the window length five. Typically, the window size is either three or five. If there are one or more nodes in the window, the left most node is chosen by default and becomes the new current node. If there is not a node in the window, the window moves down a row and continues the search. If the number of consecutive rows with zero nodes reaches the gap length value, the Node Finder algorithm is called.

Given a synthetic image as shown in Figure 2.59, it is clearly seen that there is a break in the path starting at row twenty-one. If the window size is two (window length is then five) and the gap length is five, then no node would be found. The Node Finder algorithm will extract a large window, Figure 2.60, with the top centered at the current node (red) and identify path continuation nodes (blue). The Path Finder algorithm will make the left blue node the new current node and store the right one in the cell array called 'branches'.

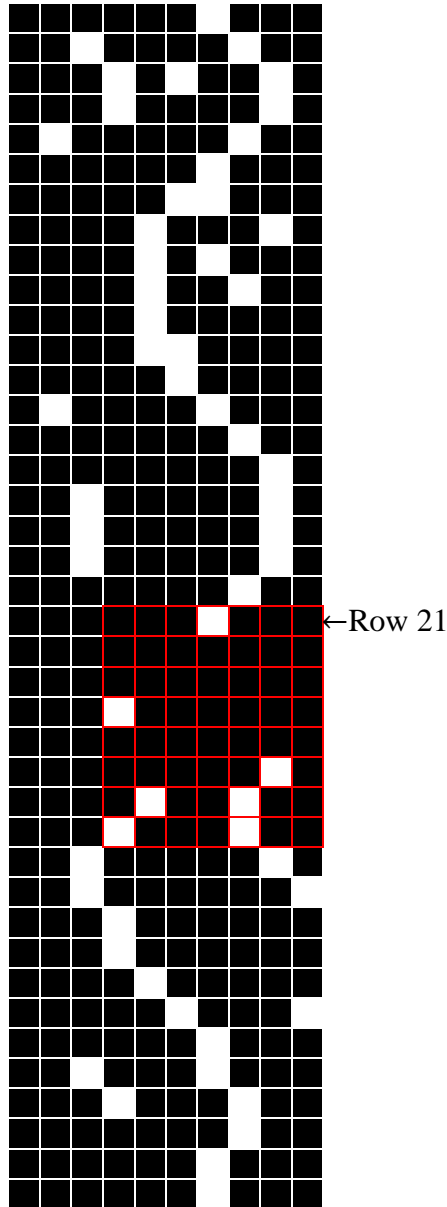


Figure 2.59: Synthetic Image

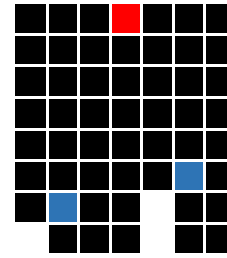
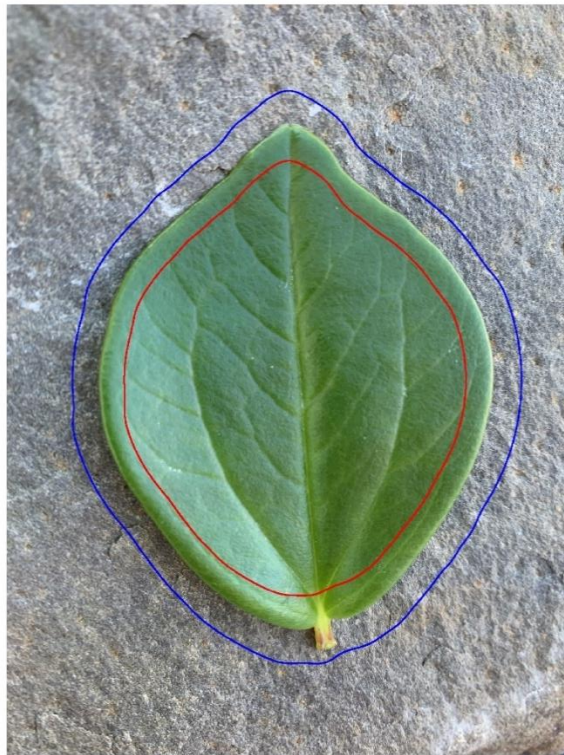


Figure 2.60: Node Finder on Synthetic Image

An edge path is considered complete if it makes it to the bottom ten rows of the image and is saved permanently. The algorithm recursively prunes as it finds paths that are incomplete. Once a path is done, the algorithm checks if ‘branches’ has any split nodes not followed and if there were any more starting locations. Only after this does the algorithm output the possible edge path indices.

### 2.2.5 Contour Finder Algorithm

The Contour Finder algorithm is the complete autonomous system that takes in a Cartesian space image and outputs two contours, one inside the leaf and one outside. The contours are one pixel thick and cannot be seen without zooming in. Figure 2.61 shows the outer (blue) and inner (red) contours with a ten-pixel thickness for visualization.



*Figure 2.61: Inner and Outer Contour*

Figure 2.62 is the block diagram describing how the contours are created. As mentioned in section 2.2.2,  $T$ , the threshold, starts high to only extract prominent edges. If a path cannot be found the threshold is lowered, edges are redefined, and the Path Finder algorithm is rerun.



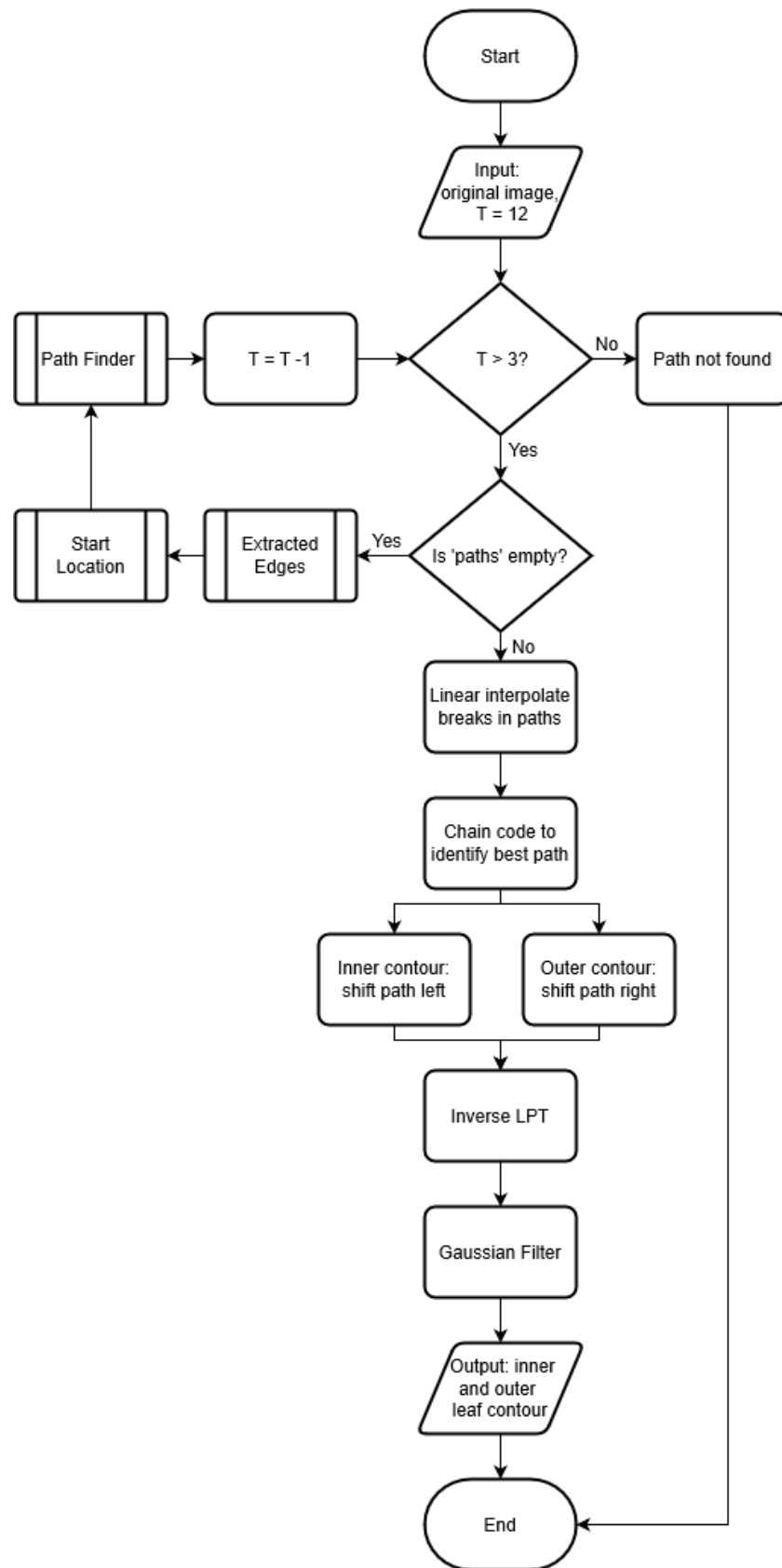


Figure 2.62: Contour Finder Algorithm Block Diagram

Once possible edge paths are found, linear interpolation is used for breaks in the path. Typically, with images in Group 2 and 3, multiple paths are found. Chain codes are generated for each path with the start location in the top row. Each number of the code is weighted differently. Codes 1, 2, 3 are weighted heavier than 5, 6, 7, because the true edge boundary typically has a downward slope. The path with the smallest cost is the chosen path to create the contours from.

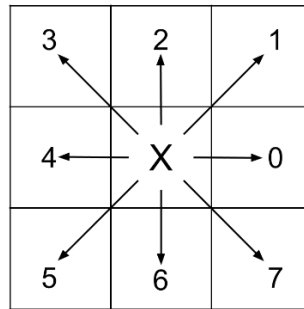
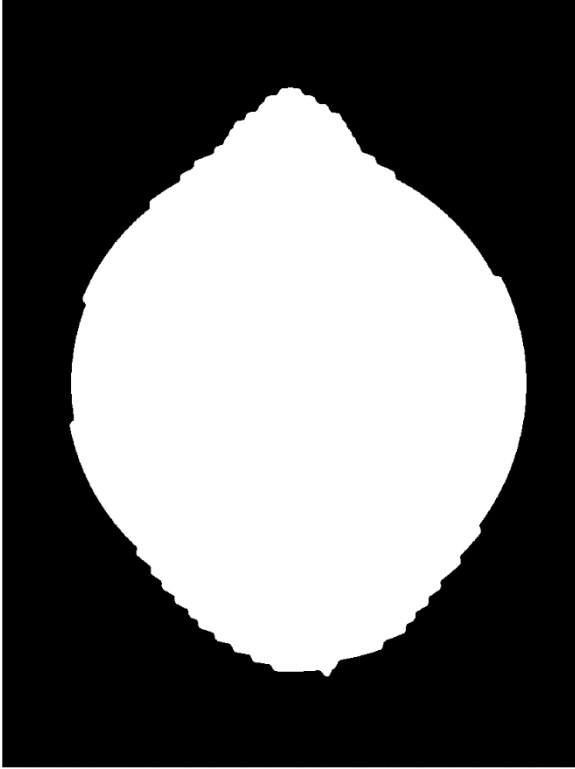
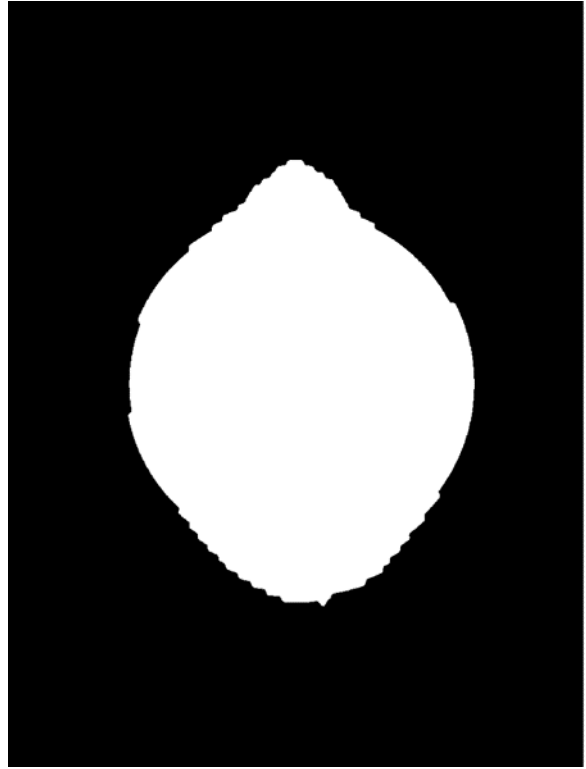


Figure 2.63: 8-Neighborhood Connectivity

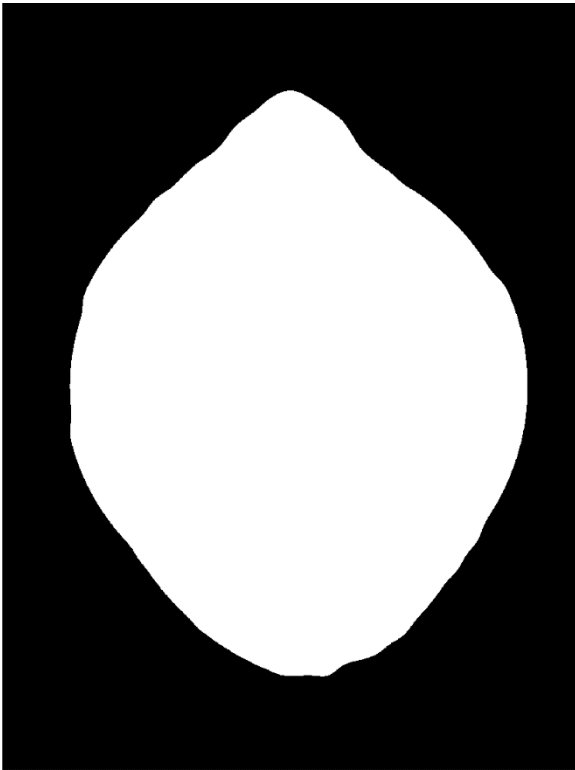
To create a contour inside the leaf, the edge path is shifted left, to a smaller radius. The outer contour is created from a shift to the right. A scale translation in Cartesian space is a shift in LPS as shown in Figure 1.2. At this stage, the images are converted back into Cartesian space through Young's [16] inverse LPT code. A low-pass Gaussian filter is applied to smooth the edges of the contour. Figure 2.64 and Figure 2.65 show the original contours while Figure 2.66 and Figure 2.67 are the resulting images following the application of the Gaussian filter. Figure 2.68 through Figure 2.76 are the final contours overlaid on the original images.



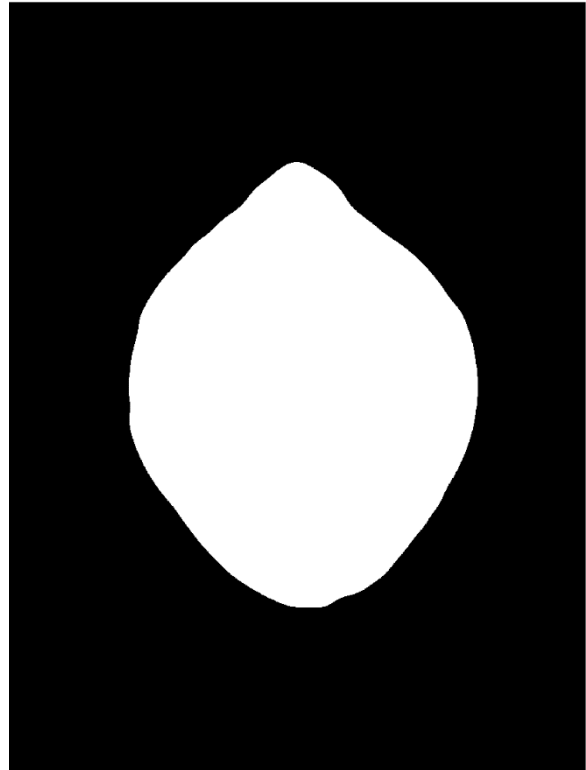
*Figure 2.64: Outer Contour of P1*



*Figure 2.65: Inner Contour of P1*



*Figure 2.66: Smoothed Outer Contour of P1*



*Figure 2.67: Smoothed Inner Contour of P1*



*Figure 2.68: Contours of P1*



*Figure 2.69: Contours of P2*



*Figure 2.70: Contours of P3*



*Figure 2.71: Contours of S1*



*Figure 2.72: Contours of S2*



*Figure 2.73: Contours of S3*



*Figure 2.74: Contours of A1*



*Figure 2.75: Contours of A2*



*Figure 2.76: Contours of A3*

## 2.3 CATEGORIZING RESULTS

A common method to validate image segmentation results is to test against the manually segmented ground truth image. Pixels that are part of the foreground are given a value of one and background pixels are given a value of zero. By calculating the precision (positive predicted value), sensitivity (true positive rate), and accuracy one can determine how successful the segmentation algorithm is. Unfortunately, this type of validation would produce inaccurate results for what was done in this research. Between the LPT and gaussian filter, the contours found give only a rough outline of the leaf. The goal of this algorithm is to produce contours inside and outside the leaf, not to extract the true leaf boundary.

Each image can be categorized into three groups; success, failure, and incomplete. A successful image will have no part of the inner or outer contour crossing the leaf boundary. Even if a small portion of the contour touches the leaf edge, it is a failure. Figure 2.77 and Figure 2.78 show examples of this. An incomplete is defined as when the algorithm fails to find an edge path and thus no contour is created.



*Figure 2.77: Swedish Ivy Contour Failure*



*Figure 2.78: Swedish Ivy Contour Failure Close-up*



## Chapter 3 Results

### 3.1 ALGORITHM CONTOUR RESULTS

Results are summarized in the tables below. Table 3.1 displays the validation image database distribution by group and plant type. Table 3.2 displays the number of successful contours found by group and plant type. Table 3.3 shows the same data but given in percentage. The hypothesis was the simpler the leaf and background were, the higher the success rate. Therefore, peperomia in Group 1 should have the highest success rate while Algerian ivy in Group 3 should have the lowest success rate. The results in Table 3.3 validate the hypothesis. Each plant had its highest success rate in Group 1 and lowest in Group 3.

	Group 1	Group 2	Group 3	Total
Peperomia	130	141	83	354
Swedish Ivy	82	159	99	340
Algerian Ivy	49	108	67	224
Total	261	408	249	918

*Table 3.1: Validation Image Database by Plant and Group*

	Group 1	Group 2	Group 3	Total
Peperomia	126	134	75	335
Swedish Ivy	72	135	70	277
Algerian Ivy	35	51	16	102
Total	233	320	161	714

*Table 3.2: Contour Successes by Group and Plant Type*

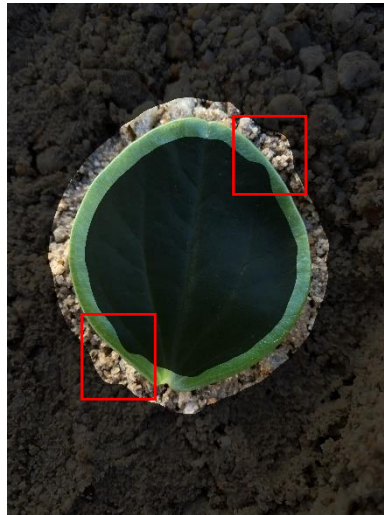
	Group 1	Group 2	Group 3	Average
Peperomia	96.9	95.0	90.4	94.6
Swedish Ivy	87.8	84.9	70.7	81.5
Algerian Ivy	71.4	47.2	23.9	45.5
Average	89.3	78.4	64.7	77.8

*Table 3.3: Contour Success Percentage*

Figure 3.1(a-i) show examples of contours that were classified successful. A contour does not need to be perfect to be successful as the goal is to find a rough outline. Figure 3.1(b) and Figure 3.1(i) both have mistakes in the contour where the contour does not follow exactly the leaf edge outline. This is acceptable if the error does not force the contour to touch the true leaf edge.



*a*



*b*



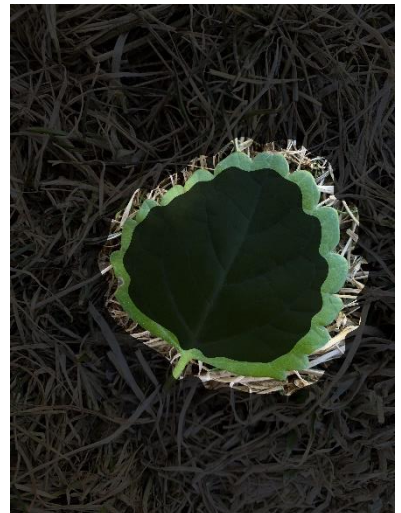
*c*



*d*



*e*



*f*





Figure 3.1: Contour Success Examples

Table 3.4 and Table 3.5 give the failed contour distribution. Including the incomplete group in the calculation, peperomia has a 2.3% failure rate, Swedish ivy has 16.8% and Algerian ivy with 32.6%. Of the 918 images, 138 or 15% were considered a failure.

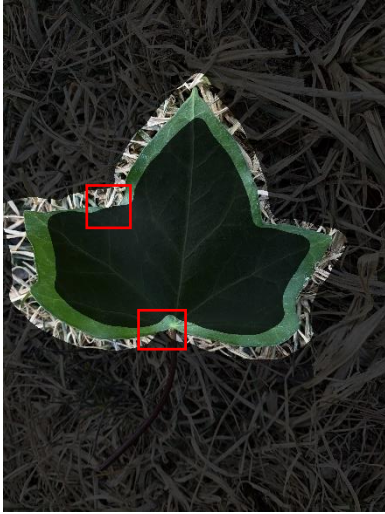
	Group 1	Group 2	Group 3	Total
Peperomia	0	1	7	8
Swedish Ivy	9	22	26	57
Algerian Ivy	9	28	36	73
Total	18	51	69	138

Table 3.4: Contour Failures by Group and Plant Type

	Group 1	Group 2	Group 3	Average
Peperomia	0.0	0.7	8.4	2.3
Swedish Ivy	11.0	13.8	26.3	16.8
Algerian Ivy	18.4	25.9	53.7	32.6
Average	6.9	12.5	27.7	15.0

Table 3.5: Contour Failure Percentage

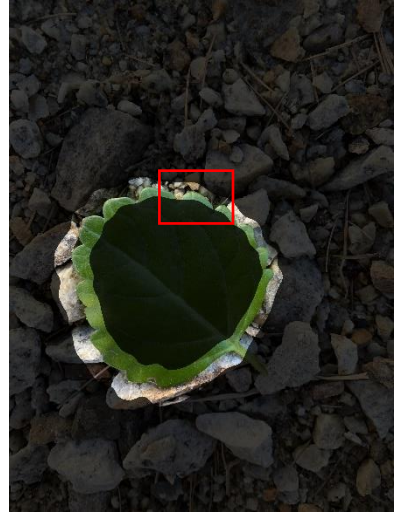
Figure 3.2(a-i) are examples of failed contours. Some contours, Figure 3.2(a-c), are relatively accurate contours but contain one or two areas that touch the leaf boundary. Many of the images in the failure group would be considered successful if the inner and outer contour were shifted one more pixel while still in LPS.



*a*



*b*



*c*



*d*



*e*



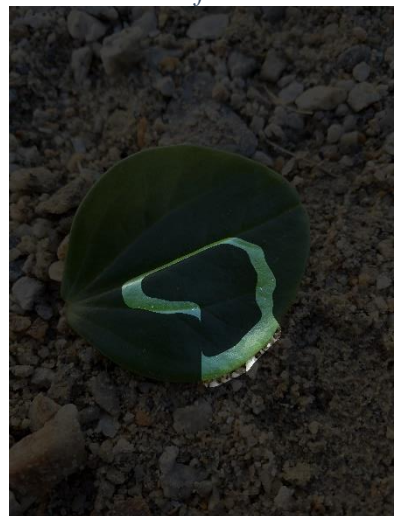
*f*



*g*



*h*



*i*

*Figure 3.2: Contour Failure Examples*



Failures are caused by either following the wrong path or the Gaussian filter. The Gaussian filter can over smooth the edges in the contour. For Swedish ivy it is typically leaves that have larger crenate edges that the inner contour touches the leaf edge after the Gaussian filter. Figure 3.3 and Figure 3.4 show an example of how the Gaussian filter causes an image to be a failure. The inner contour only crosses the leaf edge boundary after the smoothing filter.



Figure 3.3: Contours Before Gaussian Filter



Figure 3.4: Contours After Gaussian Filter

Incompletes are the true failures of the algorithm. For these images, the algorithm was not able to find a complete path to follow and therefore a contour could not be found. Swedish ivy is the only plant that follows the trend of having results becoming worse as the background becomes more cluttered. Table 3.6 and Table 3.7 give the incomplete image distribution. Overall, the model could not find paths for 66, or 7.2%, of the images.

	Group 1	Group 2	Group 3	Total
Peperomia	4	6	1	11
Swedish Ivy	1	2	3	6
Algerian Ivy	5	29	15	49
Total	10	37	19	66

Table 3.6: Incompletes by Group and Plant Type

	Group 1	Group 2	Group 3	Average
Peperomia	3.1	4.3	1.2	3.1
Swedish Ivy	1.2	1.3	3.0	1.8
Algerian Ivy	10.2	26.9	22.4	21.9
Average	3.8	9.1	7.6	7.2

Table 3.7: Percentages of Incompletes

Figure 3.5(a-f) show LPS edge images where paths could not be found. Either breaks in the path were too large, the path curved up, or a starting position was not found.

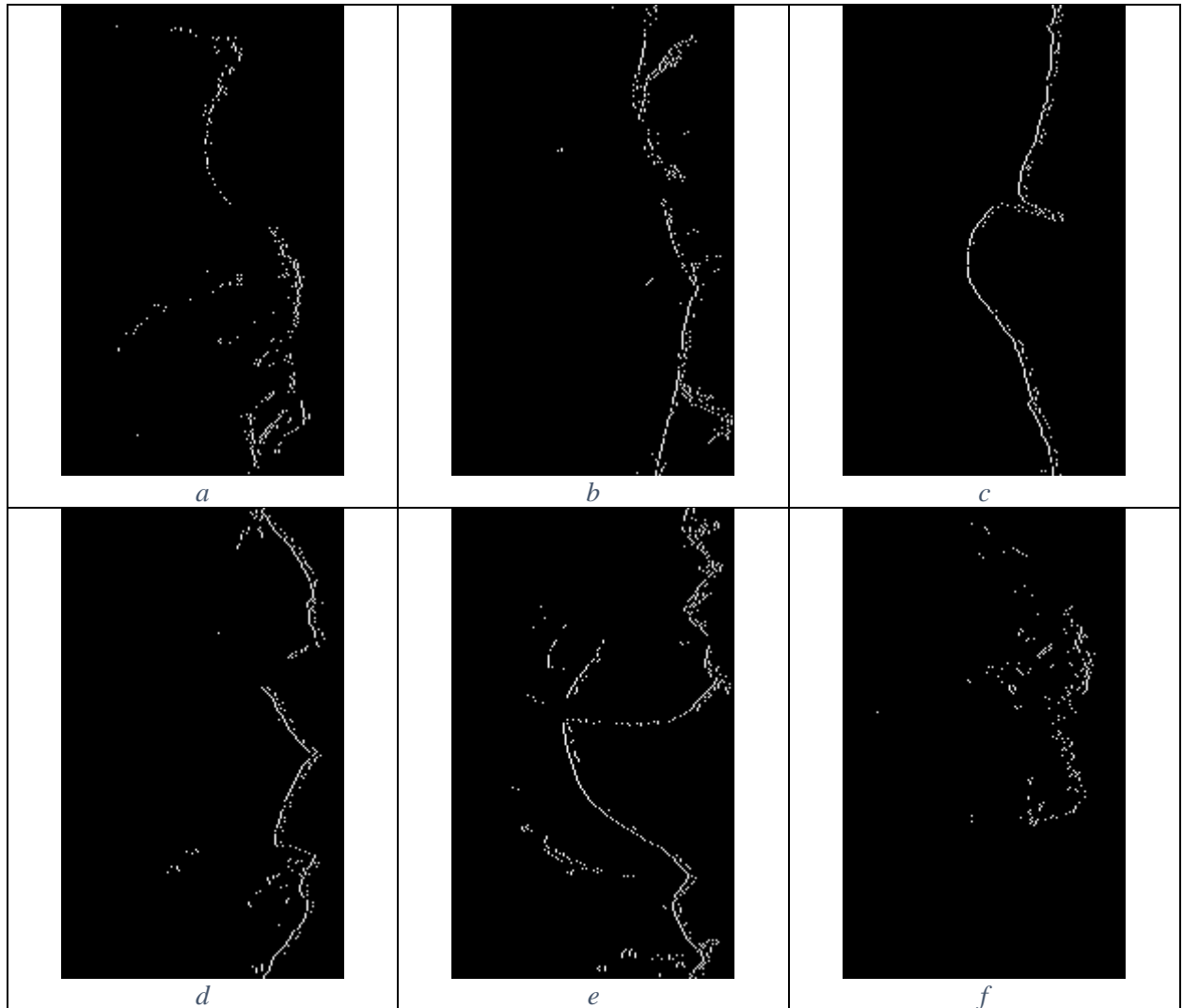


Figure 3.5: Incomplete Examples

Table 3.8 and Table 3.9 give a better distribution of contour success and failures when excluding the incomplete group. Peperomia plants in Group 1 now have a 100% success rate. Overall the success rate goes from 77.8% to 83.8% and the failure rate increases 1.2%.

	Group 1	Group 2	Group 3	Average
Peperomia	100.0	99.3	91.5	97.7
Swedish Ivy	88.9	86.0	72.9	82.9
Algerian Ivy	79.5	64.6	30.8	58.3
Average	92.8	86.3	70.0	83.8

*Table 3.8: Contour Success Percentage Excluding Incompletes*

	Group 1	Group 2	Group 3	Average
Peperomia	0.0	0.7	8.5	2.3
Swedish Ivy	11.1	14.0	27.1	17.1
Algerian Ivy	20.5	35.4	69.2	41.7
Average	7.2	13.7	30.0	16.2

*Table 3.9: Contour Failure Percentage Excluding Incompletes*

## 3.2 THRESHOLD VALUES

The threshold value used to define the LPS edges is a key part of the algorithm. If the threshold is set too high, then Path Finder will not be able to find a path. If set too low the algorithm is slow and there is a higher chance of choosing an incorrect path. The threshold value is dependent on the leaf type and background clutter. Figure 3.6, Figure 3.7, and Figure 3.8 show the threshold distribution of each background group to give insight on leaf type and threshold value. Success and failed images are included in the plots. Images grouped as incomplete did not have a viable path at the lowest threshold (four) and therefore were not included in the distribution.

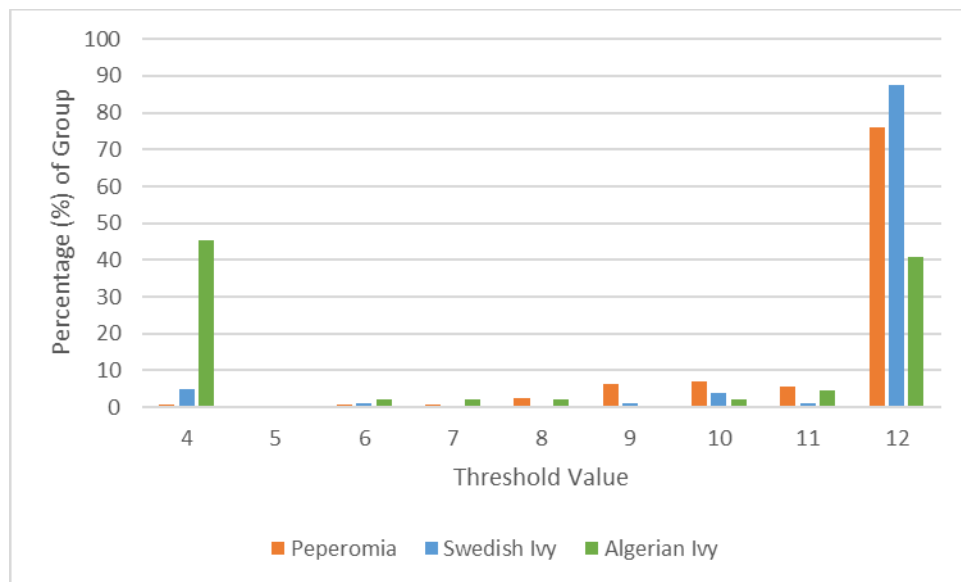


Figure 3.6: Threshold Distribution of Group 1

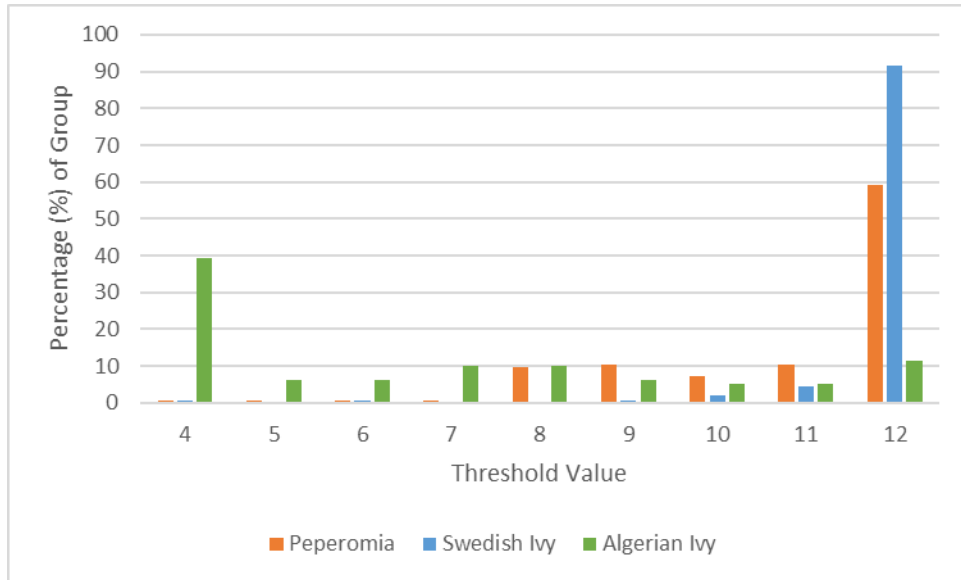


Figure 3.7: Threshold Distribution of Group 2

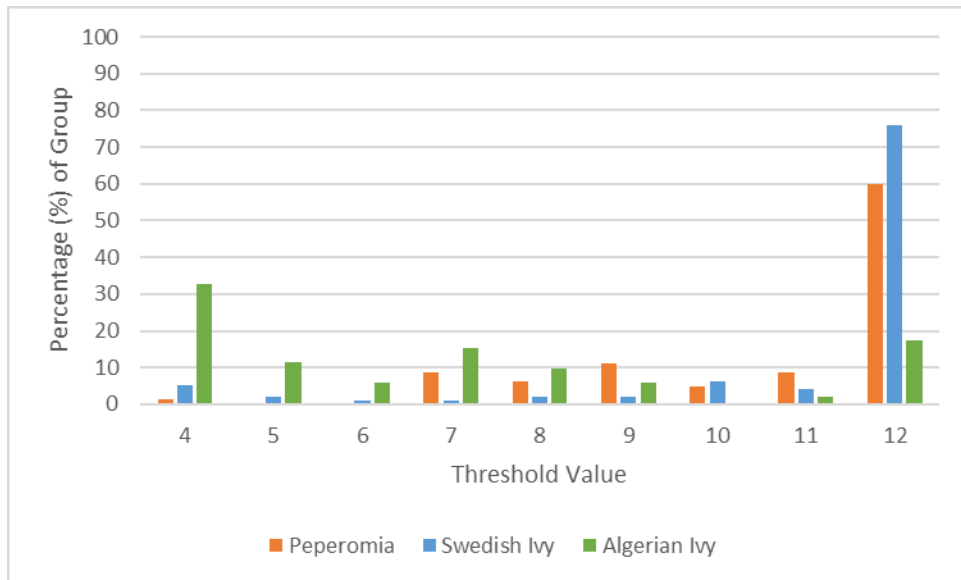


Figure 3.8: Threshold Distribution of Group 3

For all three groups, Swedish Ivy has the highest percentage of images at threshold 12. This could be for two reasons. Visually they are brighter green than Algerian Ivy and Peperomia, giving the leaf edges more contrast against the background. Though more likely it is due to the crenate edges. Swedish ivy will have a larger perimeter than a peperomia leaf of the same area. Since there are more edge pixels, the chance of enough edges being chosen at a higher threshold is better.

Similar to the contour results, Algerian ivy did significantly worse for threshold values compared to peperomia and Swedish ivy. However, in each graph Algerian ivy has a substantial percentage increase at a threshold of four compared to five. Since this is across all groups, it is not the level of noise in the background, but the lower chroma contrast between background and Algerian ivy leaf edges that cause a lower threshold to be needed. Table 3.10 displays the average threshold for each group and plant. Figure 3.9, Figure 3.10, and Figure 3.11 display the same information but show the threshold distribution of each plant across the groups.

	Group 1	Group 2	Group 3
Peperomia	11.37	10.86	10.72
Swedish Ivy	11.41	11.81	11.01
Algerian Ivy	7.93	6.78	6.88

Table 3.10: Average Thresholds

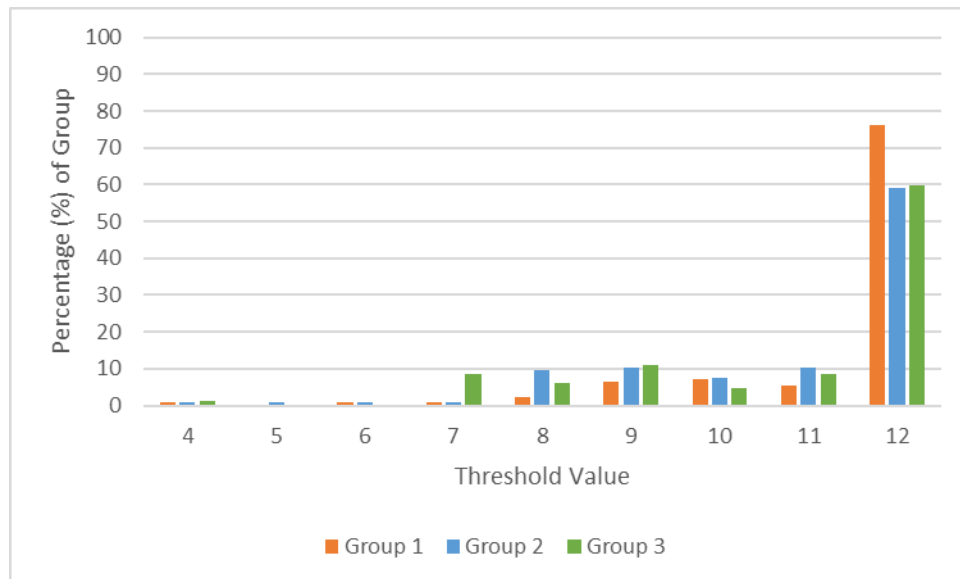


Figure 3.9: Threshold Distribution of Peperomia Plant



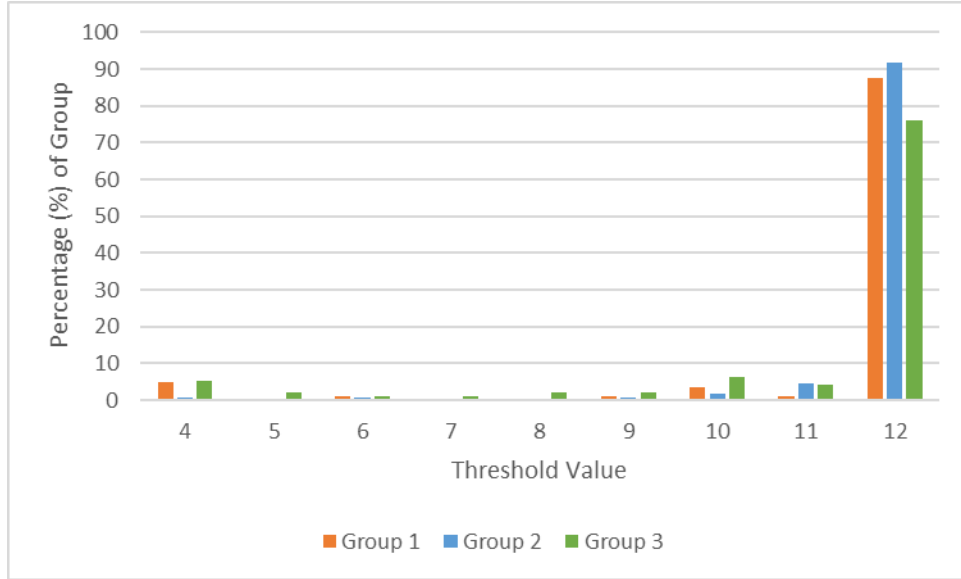


Figure 3.10: Threshold Distribution of Swedish Ivy Plant

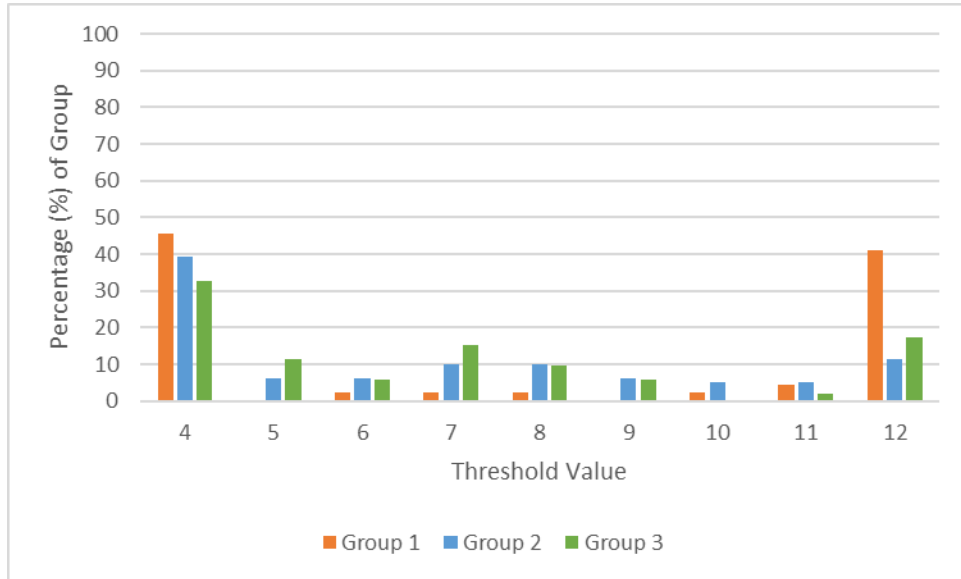


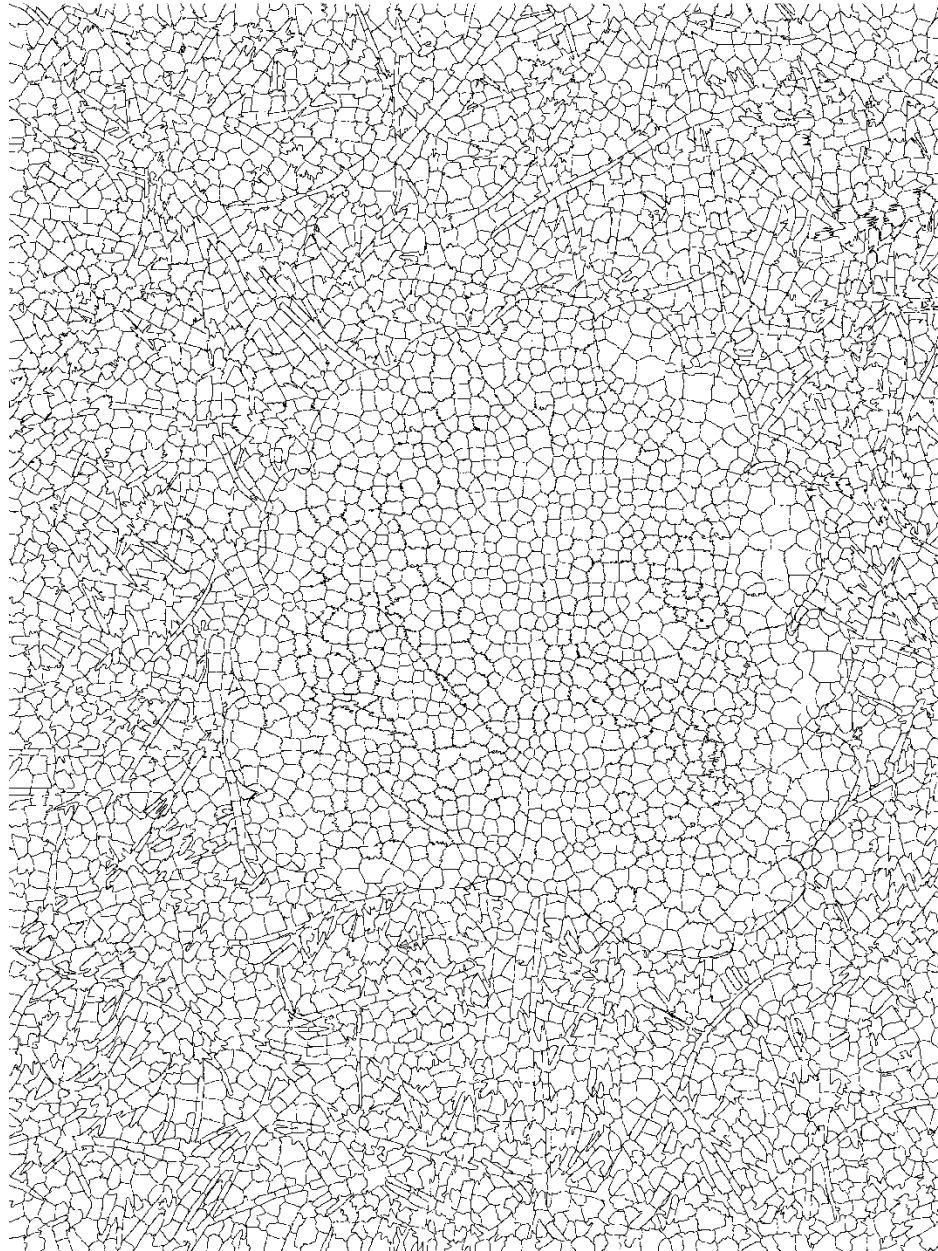
Figure 3.11: Threshold Distribution of Algerian Ivy

Peperomia (Figure 3.9) and Swedish ivy (Figure 3.10) have similar distributions over all groups. With most images having a threshold of 12 regardless of the background, the algorithm should be started at a higher threshold. This may help with false edge paths from background edges. In Figure 3.11 Algerian ivy has its highest percentage, 41%, of images at a threshold of 12

compared to Group 2 and 3. This illustrates the level of background clutter influences the minimum threshold needed to obtain an edge path in LPS.

### 3.3 MSRM RESULTS

MSRM is used to demonstrate that the model is a successful option in lieu of user-input strokes. To produce low-level segmentation regions, simple linear iterative clustering (SLIC) was used. Figure 3.12 shows the result of SLIC on S3.



*Figure 3.12: SLIC of S3*

The outer contour is used as the background marker and inner contour as the foreground marker for MSRM. Please refer to section 1.3 for a description of MSRM. Figure 3.13 through Figure 3.21 are results of MSRM.



*Figure 3.13: MSRM of P1*



*Figure 3.14: MSRM of P2*



*Figure 3.15: MSRM of P3*



*Figure 3.16: MSRM of S1*



*Figure 3.17: MSRM of S2*



*Figure 3.18: MSRM of S3*





*Figure 3.19: MSRM of A1*

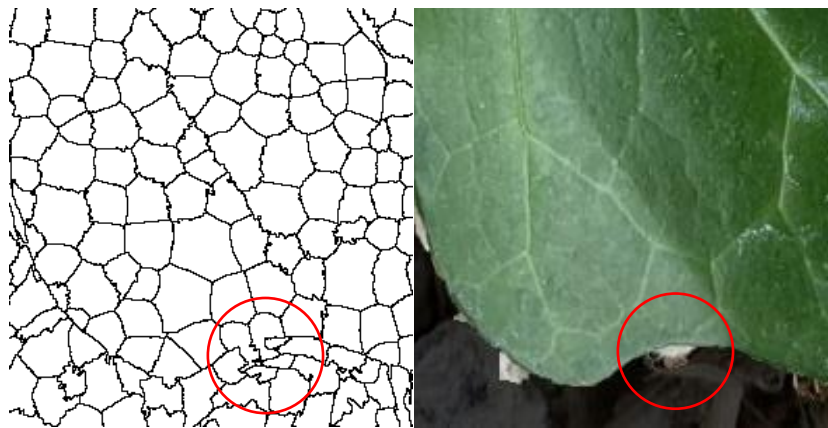


*Figure 3.20: MSRM of A2*



*Figure 3.21: MSRM of A3*

The success of MSRM heavily relies on the input SLIC image. Figure 3.22 is a closeup of A2 after SLIC and MSRM. The red circle on the SLIC image shows clusters that lie on the leaf edge. Anytime this occurs the MSRM algorithm either counts this cluster as background or foreground, either way it will not create a clean extraction. With that in mind, the outer and inner contours used as input strokes were enough to guide MSRM into correctly segmenting the leaves.



*Figure 3.22: SLIC and MSRM Closeup of A2*

### 3.4 TEST IMAGE DATABASE RESULTS

Of the eighty test images, the model was able to produce successful contours for sixty, or 75% of them. The results are shown in Table 3.11. However, to say the algorithm is 75% successful would be not a reliable metric for overall performance. The goal of the test image database is to determine the robustness only against leaf shape, background clutter and what scenarios cause the algorithm to fail. Figure 3.23(1-31), Figure 3.24(1-5), and Figure 3.25(1-4) display all 40 white background images by success, failure, and incomplete groups respectively. Figure 3.26(1-29), Figure 3.27(1-7), and Figure 3.28(1-4) display the natural background images by group.

	Success	Failure	Incomplete
White Background	31	5	4
Natural Background	29	7	4
Total	60	12	8

*Table 3.11: Test Image Results*





3



4



5



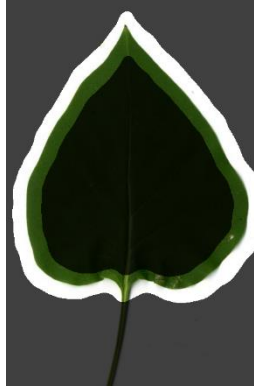
6



7



8



9



10



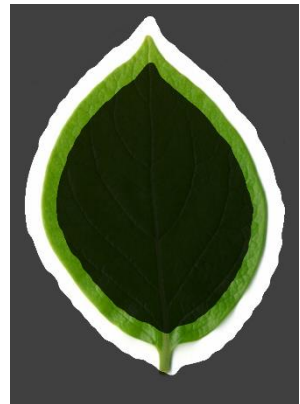
11



12



13



14





15



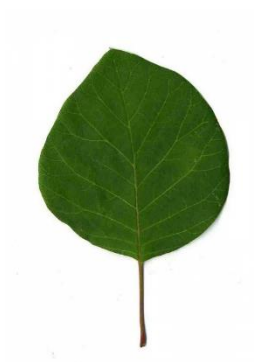
16



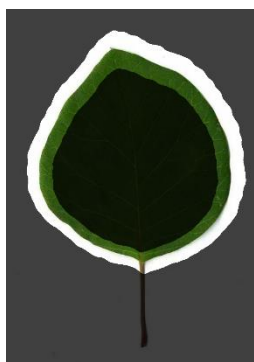
17



18



19



20





21



22



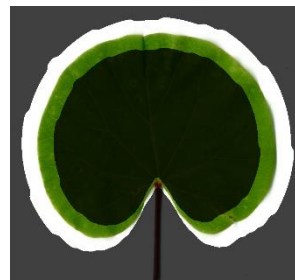
23



24



25



26





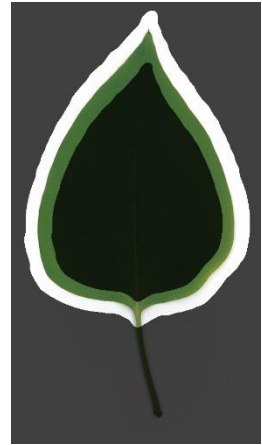
27



28



29

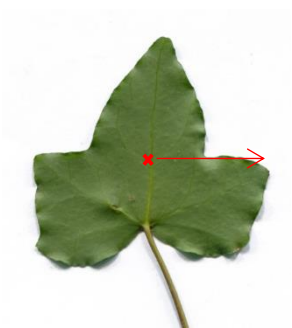


30



31

Figure 3.23: White Background Test Images in Success Group



1



2



Figure 3.24: White Background Test Images in Failure Group

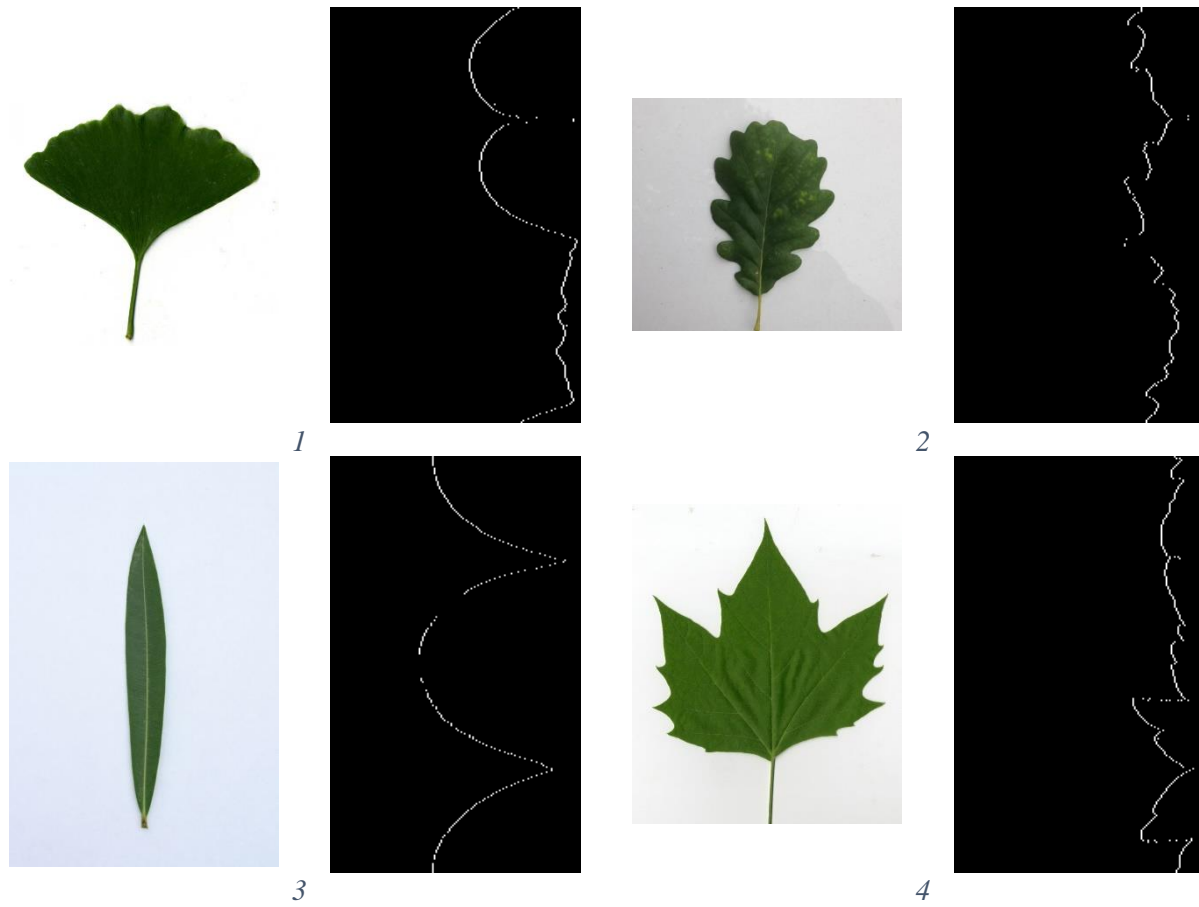
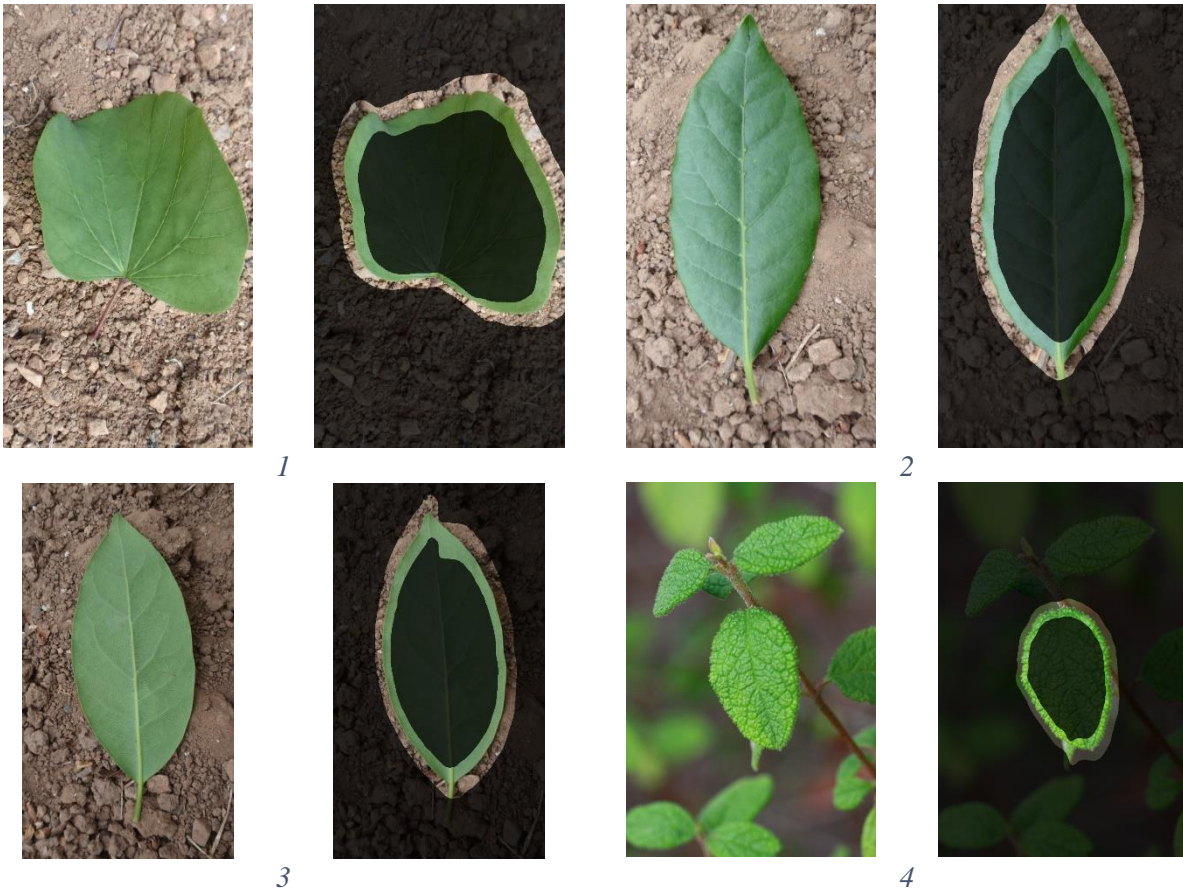
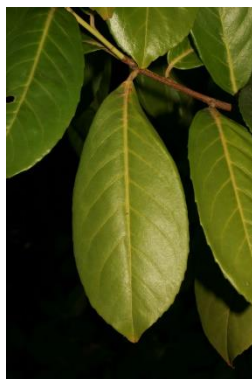


Figure 3.25: White Background Test Images in Incomplete Group



Within the failure group, Figure 3.24(1,3-5) are similar to leaves in the success group. This gives no conclusion as to what causes the model to fail. The center of leaf 1 in Figure 3.24 is marked by a red cross. When performing the LPT, the edge of the leaf is aligned radially at the red arrow. This can cause portions of the leaf to be misrepresented in LPS. Leaf 2 has a dead spot at the edge of the leaf that was most likely categorized as background during k-mean clustering. Leaf 3 has a cordate base shape that the algorithm ignored. Leaves 4 and 5 have acute apex's that were cut off. Leaves in Figure 3.25 all had definitive start locations in their binary LPS edge image though each contained large gaps. Leaves with an acute apex, acute base, or large serrate edges are more difficult to find a contour over obtuse apex shaped leaves.





5

6



7

8



9

10



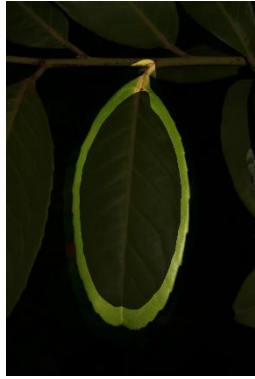
11

12





13



14



15



16



17



18





19



20



21



22



23



24



25



26





27



28



29



Figure 3.26: Natural Background Test Images in Success Group



1



2



3



4





5

6

7

Figure 3.27: Natural Background Test Images in Failure Group

The algorithm was able to successfully extract contours from twenty-nine of the forty natural background images. Of the seven leaves in the failure group, five had outer contours that cut off the tip of the leaf. Although the contours of leaf 1 of Figure 3.27 look successful, it crossed through a leaf spike. The overlapping leaves in leaf 4 was too much of a noisy background for the model.



1

2

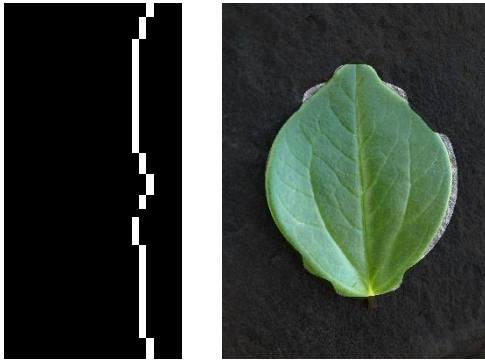


3  
*Figure 3.28: Natural Background Test Images in Incomplete Group*  
 4

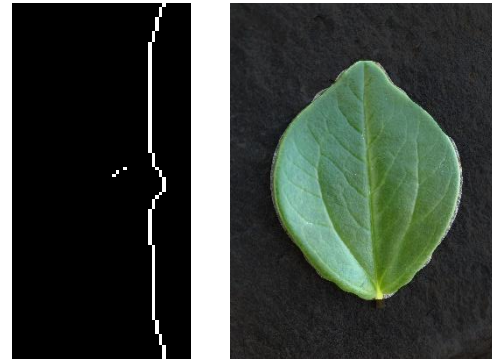
For the leaves in Figure 3.28, three had simple natural backgrounds that were expected to produce successful contours. Leaf 1 has a discolored portion that was most likely grouped as background during k-means clustering, causing a gap in the binary LPS edge image. Leaf 2 is like many images that were successful, but with some reflection and shadowing. Leaf 3's stem is green and is cropped out of the image, causing there to not be any start or stop location. Leaf 4 has discolored parts and reflection producing an inaccurate binary LPS edge image.

### 3.5 LOG POLAR SPACE SIZE

Deciding on 150 rings and 250 wedges (150 x 250 pixels) for the LPT was done by trial and error before the algorithm was developed. The goal was to decimate the original image as much as possible and visually retain a clear edge boundary. Once the algorithm was constructed, different LPT sizes were tested and evaluated. Figure 3.29 through Figure 3.64 show the binary LPS edges at 25 x 50, 50 x 100, 150 x 250, and 200 x 350 pixels. The contours shown were not shifted to create an outer or inner contour but to show how accurate the algorithm segments the leaf at each LPT size.



*Figure 3.29: 25 x 50 LPS Edges and Contour Result (P1)*



*Figure 3.30: 50 x 100 LPS Edges and Contour Result (P1)*



*Figure 3.31: 150 x 250 LPS Edges and Contour Result (P1)*



*Figure 3.32: 200 x 350 LPS Edges and Contour Result (P1)*





Figure 3.33: 25 x 50 LPS Edges and Contour Result (P2)



Figure 3.34: 50 x 100 LPS Edges and Contour Result (P2)



Figure 3.35: 150 x 250 LPS Edges and Contour Result (P2)



Figure 3.36: 200 x 350 LPS Edges and Contour Result (P2)

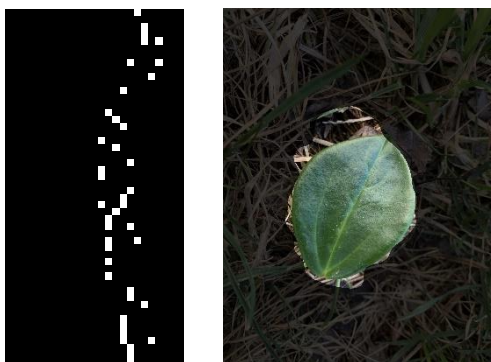


Figure 3.37: 25 x 50 LPS Edges and Contour Result (P3)



Figure 3.38: 50 x 100 LPS Edges and Contour Result (P3)



Figure 3.39: 150 x 250 LPS Edges and Contour Result (P3)



Figure 3.40: 200 x 350 LPS Edges and Contour Result (P3)

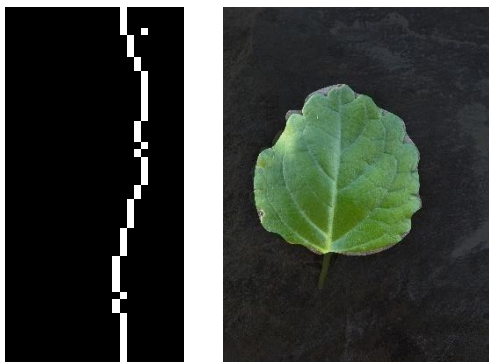


Figure 3.41: 25 x 50 LPS Edges and Contour Result (S1)



Figure 3.42: 50 x 100 LPS Edges and Contour Result (S1)



Figure 3.43: 150 x 250 LPS Edges and Contour Result (S1)



Figure 3.44: 200 x 350 LPS Edges and Contour Result (S1)



Figure 3.45: 25 x 50 LPS Edges and Contour Result (S2)



Figure 3.46: 50 x 100 LPS Edges and Contour Result (S2)



Figure 3.47: 150 x 250 LPS Edges and Contour Result (S2)



Figure 3.48: 200 x 350 LPS Edges and Contour Result (S2)

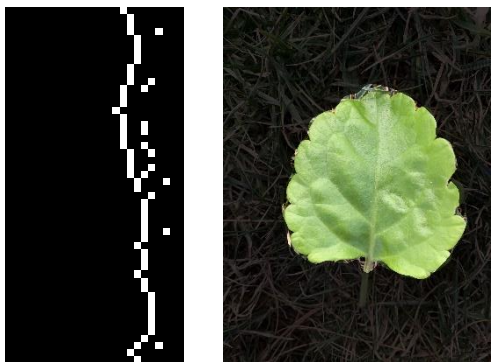


Figure 3.49: 25 x 50 LPS Edges and Contour Result (S3)



Figure 3.50: 50 x 100 LPS Edges and Contour Result (S3)

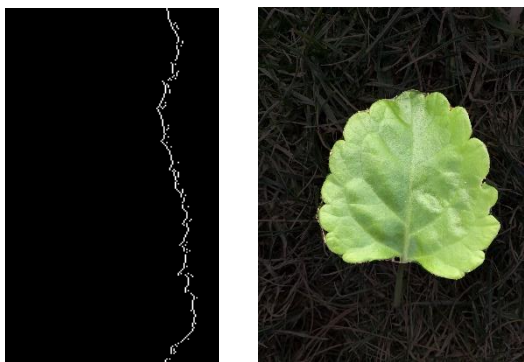


Figure 3.51: 150 x 250 LPS Edges and Contour Result (S3)



Figure 3.52: 200 x 350 LPS Edges and Contour Result (S3)

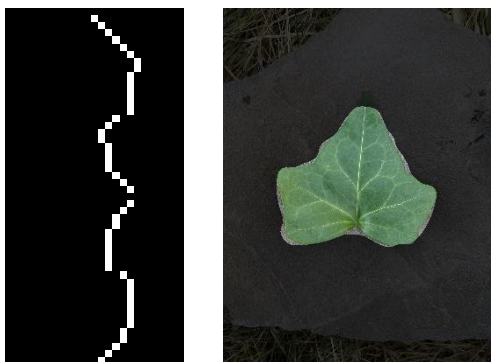


Figure 3.53: 25 x 50 LPS Edges and Contour Result (A1)

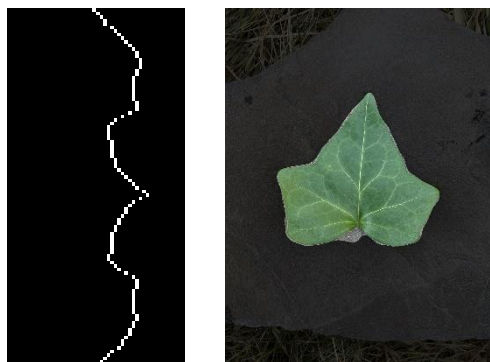


Figure 3.54: 50 x 100 LPS Edges and Contour Result (A1)



Figure 3.55: 150 x 250 LPS Edges and Contour Result (A1)



Figure 3.56: 200 x 350 LPS Edges and Contour Result (A1)



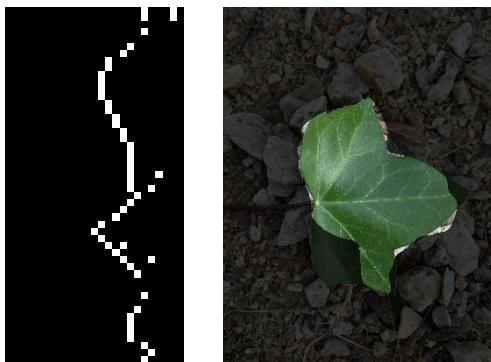


Figure 3.57: 25 x 50 LPS Edges and Contour Result (A2)

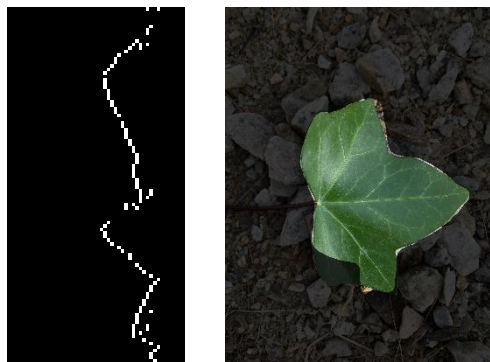


Figure 3.58: 50 x 100 LPS Edges and Contour Result (A2)

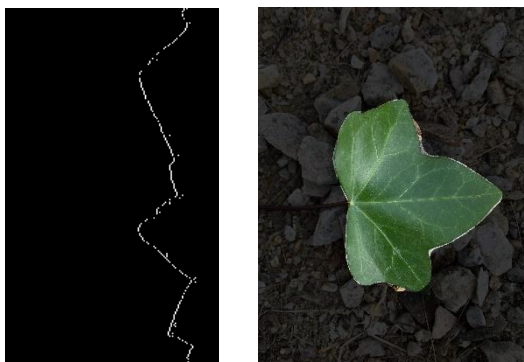


Figure 3.59: 150 x 250 LPS Edges and Contour Result (A2)



Figure 3.60: 200 x 350 LPS Edges and Contour Result (A2)

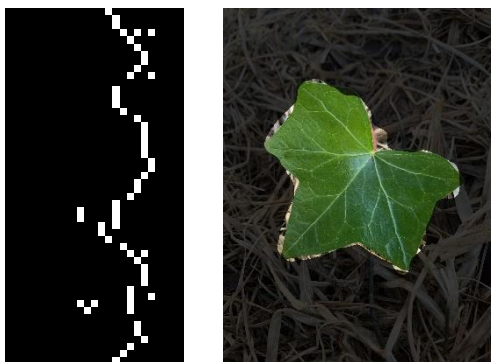


Figure 3.61: 25 x 50 LPS Edges and Contour Result (A3)

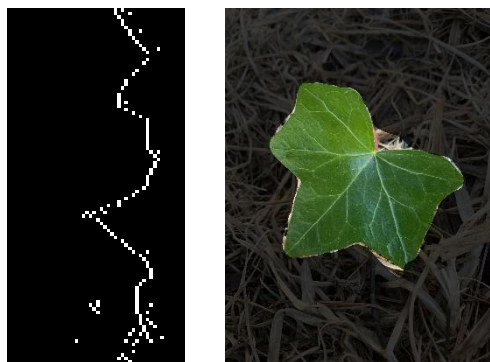


Figure 3.62: 50 x 100 LPS Edges and Contour Result (A3)



Figure 3.63: 150 x 250 LPS Edges and Contour Result (A3)

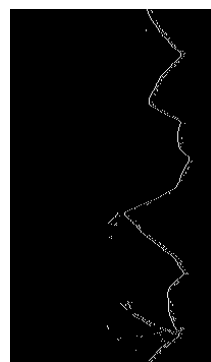


Figure 3.64: 200 x 350 LPS Edges, Failed Path



At 25 x 50 and 50 x 100 LPS sizes, Swedish ivy leaves visually had the most accurate contour. As expected, Group 1 (stone background) has better results for all three leaves. The results of 50 x100 and 150 x 250 LPS sizes are drastically different. In Figure 3.50 the LPS edge image does not show the crenate edges, while Figure 3.51 does. The contours are not any better with 200 x 350 LPS edges than 150 x 250. These results demonstrate that using 150 rings and 250 wedges is of sufficient resolution from the original 3024 x 4032 Cartesian images.

### 3.6 CARTESIAN SPACE IMAGE SIZE

To test the algorithm against image size, the validation images were decimated by one-half and one-tenth, becoming roughly 2000 x 1500 pixels and 400 x 300 pixels respectively. Table 3.12 through Table 3.17 display the distribution of contour successes, failures, and incompletes over each group and image size. Figure 3.65 through Figure 3.91 are the nine leaves (P1 through A3) at each image size.

Image Size	Group 1	Group 2	Group 3	Total
Original	233	320	161	714
One-half	230	317	161	708
One-tenth	214	317	157	688

Table 3.12: Contour Successes by Group and Image Size

Image Size	Group 1	Group 2	Group 3	Average
Original	89.3	78.4	64.7	77.8
One-half	88.1	77.7	64.7	77.1
One-tenth	82.0	77.7	63.1	74.9

Table 3.13: Contour Success Percentages by Group and Image Size

Image Size	Group 1	Group 2	Group 3	Total
Original	18	51	69	138
One-half	19	55	65	139
One-tenth	29	54	63	146

Table 3.14: Contour Failures by Group and Image Size

Image Size	Group 1	Group 2	Group 3	Average
Original	6.9	12.5	27.7	15.0
One-half	7.3	13.5	26.1	15.1
One-tenth	11.1	13.2	25.3	15.9

Table 3.15: Contour Failure Percentage by Group and Plant Size

Image Size	Group 1	Group 2	Group 3	Total
Original	10	37	19	66
One-half	12	36	23	71
One-tenth	18	37	29	84

Table 3.16: Incompletes by Group and Image Size

Image Size	Group 1	Group 2	Group 3	Average
Original	3.8	9.1	7.6	7.2
One-half	4.6	8.8	9.2	7.7
One-tenth	6.9	9.1	11.6	9.2

*Table 3.17: Percentages of Incompletes by Group and Image Size*

Table 3.13 shows as the image size gets smaller the average success rate goes down. This is expected due to down sampling the image significantly. For each group the success rate either stays the same or goes down as the image gets smaller. Interestingly, in Table 3.14 the number of failed contours increase in Group 1 but decrease in Group 3. For Algerian ivy leaves in Group 3, 36 leaves have failed contours at original size. Of those 36 leaves, at one-tenth image size nine became successes, four become incompletes and the rest remained failed contours. Group 3 has the hardest background to segment the leaf from. Presumably the decrease of failed contours at one-tenth image size is from the background being decimated enough to become blurred and obtain a higher chroma contrast compared to the leaf.



*Figure 3.65: Contour for P1 at Original Size*



*Figure 3.66: Contour for P1 at One-half Size*



*Figure 3.67: Contour for P1 at One-tenth Size*



*Figure 3.68: Contour for P2 at Original Size*



*Figure 3.69: Contour for P2 at One-half Size*



*Figure 3.70: Contour for P2 at One-tenth Size*



*Figure 3.71: Contour for P3 at Original Size*



*Figure 3.72: Contour for P3 at One-half Size*



*Figure 3.73: Contour for P3 at One-tenth Size*





*Figure 3.74: Contour for S1 at Original Size*



*Figure 3.75: Contour for S1 at One-half Size*



*Figure 3.76: Contour for S1 at One-tenth Size*



*Figure 3.77: Contour for S2 at Original Size*



*Figure 3.78: Contour for S2 at One-half Size*



*Figure 3.79: Contour for S2 at One-tenth Size*



*Figure 3.80: Contour for S3 at Original Size*



*Figure 3.81: Contour for S3 at One-half Size*



*Figure 3.82: Contour for S3 at One-tenth Size*





*Figure 3.83: Contour for A1 at Original Size*



*Figure 3.84: Contour for A1 at One-half Size*



*Figure 3.85: Contour for A1 at One-tenth Size*



*Figure 3.86: Contour for A2 at Original Size*



*Figure 3.87: Contour for A2 at One-half Size*



*Figure 3.88: Contour for A2 at One-tenth Size*



*Figure 3.89: Contour for A3 at Original Size*



*Figure 3.90: Contour for A3 at One-half Size*



*Figure 3.91: Contour for A3 at One-tenth Size*

Contours at original size are almost identical to the contours at one-half size. 25 of the 27 contours are successful. A1 (Figure 3.85) and A3 (Figure 3.91) have inner contours that touch the leaf edge. While contours are cleaner with large Cartesian space images, the algorithm can handle low-resolution images.



## Chapter 4 Conclusion

Designing a recursive path following algorithm in LPS is an innovative approach to reducing user input for segmentation algorithms. The objective was to create a pre-processing algorithm that can autonomously create contours around the main object of an image. This will allow other segmentation methods to be more fully autonomous and less subjective. The hypothesis, which was proved true, was the simpler leaves and simpler background mediums would have a higher success rate. While this research focused on extracting contours of leaves, the algorithm could be updated and improved to use on other images.

The model outputs two bounded contours; one inside and one outside the main object. Of the 918 images in the validation database, 714 had successfully extracted contours. The failures occurred either from the algorithm not finding a complete edge path in LPS or the wrong path was followed and created a bad contour. While the successful results are promising, the failed and incomplete contours give insight of how to improve the model. As a pilot study in LPS recursive path following the model is only at the beginning stages. Chapter 5 will discuss portions of the algorithm that can be improved and future objectives.

To demonstrate that the model is a successful option in lieu of user-input interaction, the MSRM algorithm was used which needs user-input foreground and background markers to begin. The inner contour was used as the foreground marker and the outer contour as the background marker. MSRM successfully segmented the leaves from the background as was demonstrated in section 3.3.

## Chapter 5 Future Improvements

As a preliminary study in LPT recursive path following, this research shows excellent results for creating autonomous contours in lieu of user-input a priori knowledge. To improve upon this pre-processing system, the Extracted Edges algorithm should calculate the horizontal and vertical pixel value difference. Leaves that are narrow or have acute apex's do not perform well with only horizontal edges identified. Leaf 3 depicted in Figure 3.25 is extremely narrow. This causes gaps between the edge nodes in the LPS representation which are problematic. Including the vertical pixel value difference before taking the absolute value and thresholding could improve the model's robustness of certain leaf types.

While analyzing failed and incomplete leaves, images were cropped and rotated, then tested again. Unfortunately, the algorithm implemented does not know the difference of a failed versus successful contour. Either the algorithm finds a complete path (contour found) or no path is found (incomplete). When a path is not found, autonomously rotating the image then running the model again would reduce the number of incomplete images. Rotating the image will not change the center point or cause loss of information. A method of autonomous cropping of the image would be beneficial, however, there runs a chance of cutting off the leaf edge which would not be desirable. This idea should be researched further. If all leaves were required to be in the inner two-thirds of the image this problem would be mitigated. There will always be a tradeoff between how robust the model is versus how many restraints are given.

It is extremely unlikely to create one algorithm that can extract leaves from every background medium without using a priori knowledge of the background. If the background

medium was known, the Extracted Edges algorithm could be tuned for different mediums. A neural network could be trained to determine the type of background (grass, leaves, stone, dirt, etc.) This would reduce background restrictions and allow the algorithm to remain autonomous for the end user.

Once the algorithm is robust enough that it can handle extracting leaves from all levels of background clutter, the next step is to reduce leaf restrictions. Damaged and discolored leaves should be included. It is expected that new challenges will arise when the leaf is not green. Currently the Extracted Edges algorithm uses only chroma when calculating adjacent pixel difference. A multi-colored leaf will likely have false edges in chroma color space. Although it will add many challenges, it is believed that defining the LPS leaf edges using luminance, chroma and hue color space would significantly improve the performance.

The need for a contour extractor model is not limited to image segmentation for leaf extraction. Within the medical field, having a system that extracts tumor boundaries could improve the timing and accuracy of reading an x-ray or MRI scan. It could be used to identify cavities within dental x-rays. Satellite images could use contour extraction to locate bodies of water. For farming, diseased or discolored fruit could be identified and removed from the harvest. While this model is tailored toward leaf extraction, the idea of log polar space recursive path following for contour extraction can be used in many areas of application.

## References

- [1] M. Perez-Patricio, J. L. Camas-Anzueto, A. Sanchez-Alegria, A. Aguilar-Gonzalez, F. Gutierrez-Miceli, E. Escobar-Gomez, Y. Voisin, C. Rios-Rojas and R. Grajales-Coutino, "Optical Method for Estimating the Chlorophyll Contents in Plant Leaves".
- [2] K. Itakura and F. Hosoi, "Automatic Leaf Segmentation for Estimating Leaf Area and Leaf Inclination Angle in 3D Plant Images," *Sensors*, vol. 18, 2018.
- [3] S. Weizheng, W. Yachun, C. Zhanliang and W. Hongda, "Grading Method of Leaf Spot Disease Based on Image Processing," in *International Conference on Computer Science and Software Engineering*, 2008.
- [4] M. Grand-Brochier, A. Vacavant, G. Cerutti, C. Kurtz, J. Weber and L. Tougne, "Tree Leaves Extraction in Natural Images: Comparative Study of Preprocessing Tools and Segmentation Methods," *IEEE Transactions on Image Processing*, vol. 24, no. 5, pp. 1549-1560, 2015.
- [5] G. Cerutti, L. Tougne, J. Mille, A. Vacavant and D. Coquin, "Understanding leaves in natural images - A model-based approach for tree species identification," *Computer Vision and Image Understanding*, vol. 117, no. 10, pp. 1482-1501, 2013.
- [6] G. Cerutti, L. Tougne, A. Vacavant and D. Coquin, "A Parametric Active Polygon for Leaf Segmentation and Shape Estimation," in *International Symposium on Visual Computing*, 2011.
- [7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Susstrunk, "SLIC Superpixels," EPFL Technical Report, 2010.
- [8] Z. Wang, K. Wang, F. Yang, S. Pan and Y. Han, "Image segmentation of overlapping leaves based on Chan-Vese model and Sobel operator," *Information Processing in Agriculture*, vol. 5, pp. 1-10, 2018.
- [9] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266-277, 2001.
- [10] L. Grady, "Random Walks for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, 2006.
- [11] W. Yang, J. Cai, J. Zheng and J. Luo, "User-Friendly Interactive Image Segmentation Through Unified Combinatorial-User Inputs," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2470-2479, 2010.
- [12] C. Rother, V. Kolmogorov and A. Blake, "GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309-314, 2004.
- [13] Y. Boykov and V. Kolmogorov, "Computing Geodesics and Minimal Surfaces via Graph Cuts," in *International Conference on Computer Vision*, Nice, France, 2003.
- [14] J. Ning, L. Zhang and C. Wu, "Interactive image segmentation by maximal similarity based region merging," *Pattern Recognition*, vol. 43, pp. 445-456, 2010.

- [15] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790-799, 1995.
- [16] D. Young, "Log-polar image sampling," MATLAB Central File Exchange, 2010.
- [17] E. L. Schwartz, "Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception," *Biological Cybernetics*, vol. 25, pp. 181-194, 1977.
- [18] C. F. R. Weiman and G. Chaikin, "Logarithmic Spiral Grids for Image Processing and Display," *Computer Graphics and Image Processing*, vol. 11, pp. 197-226, 1979.
- [19] P. S. Schenker, K. M. Wong and E. G. Cande, "Fast Adaptive Algorithms for Low-Level Scene Analysis: Applications Of Polar Exponential Grid (PEG) Representation To High-Speed, Scale-And-Rotation Invariant Target Segmentation," *Techniques and Applications of Image Understanding*, vol. 281, 1981.
- [20] R. A. Messner and H. H. Szu, "An Image Processing Architecture for Real Time Generation of Scale and Rotation Invariant Patterns," *Computer Vision, Graphics, and Image Processing*, vol. 31, pp. 50-66, 1985.
- [21] M. Bishay, R. A. Peters and K. Kawamura, "Object detection in indoor scene using log-polar mapping," in *International Conference on Robotics and Automation*, 1994.
- [22] V. J. Traver and F. Pla, "Dealing with 2D translation estimation in log-polar imagery," *Image and Vision Computing*, vol. 21, pp. 145-160, 2003.
- [23] A. A. Ellahyani, "Mean shift and log-polar transform for road sign detection," *Multimedia Tools and Applications*, vol. 76, pp. 24495-24513, 2017.
- [24] E. Koester, "A Human Visual System Inspired Feature Recognition Method Using Convolutional Neural Networks," M.S, University of New Hampshire, 2018.
- [25] N. Radhi, "A Biological Based Model of the Human Visual System Incorporating Lateral Subtractive Inhibition with Non-Uniform Sampling and Multiple Spatial Frequency Filters," MS, University of New Hampshire, 2016.
- [26] B. E. Bayer, "COLOR IMAGING ARRAY". United States Patent 3,971,065, 20 July 1976.
- [27] L. Busin, N. Vandenbroucke and L. Macaire, "Color Spaces and Image Segmentation," *Advances in Imaging and Electron Physics*, vol. 151, pp. 65-168, 2008.
- [28] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson, 2018.
- [29] M. H. Asmare, V. S. Asirvadam and L. Iznita, "Color Space Selection for Color Image Enhancement Applications," in *International Conference on Signal Acquisition and Processing*, 2009.
- [30] H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Transactions on Electronic Computers*, vol. 10, no. 2, pp. 260-268, 1961.
- [31] C. Luengo, "Cris' Image Analysis Blog," 27 September 2010. [Online]. Available: <https://www.crisluengo.net/archives/324>. [Accessed 2020].
- [32] T. Kailath, "The Divergence and Bhattacharyya Distance Measures in Signal Selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52-60, 1967.
- [33] J. Ning, L. Zhang, D. Zhang and C. Wu, "Interactive Image Segmentation by Maximal Similarity Based Region Merging," 12 July 2010. [Online]. Available: [http://www4.comp.polyu.edu.hk/~cslzhang/MSRM/PR\\_MSrm\\_website.htm](http://www4.comp.polyu.edu.hk/~cslzhang/MSRM/PR_MSrm_website.htm). [Accessed 2020].

- [34] H. Goeau, A. Joly and P. Bonnet , "ImageCLEF/LifeCLEF - Multimedia Retrieval in CLEF, Plant Task," 2014. [Online]. Available: <https://www.imageclef.org/2014/lifeclef/plant>. [Accessed 2020].
- [35] A. Ash, B. Ellis, L. Hickey, K. Johnson, P. Wilf and S. Wing, Manual of Leaf Architecture - Morphological description and categorization of dictyledonous and net-veined monocotyledonous angiosperms, Smithsonian Institution, 1998.