AN EDGE BASED LEAF SEPARATION ALGORITHM

NICHA PIEMKAROONWONG

A THESIS REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTING IN ENGINEERING SYSTEMS
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2017
KMITL-2017-IC-M-11-02

AN EDGE BASED LEAF SEPARATION ALGORITHM

NICHA PIEMKAROONWONG

A THESIS REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTING IN ENGINEERING SYSTEMS
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2017
KMITL-2017-IC-M-11-02

KMITL-2017-IC-M-11-02

AN EDGE BASED LEAF SEPARATION ALGORITHM

NICHA PIEMKAROONWONG

A THESIS REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTING IN ENGINEERING SYSTEMS
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2017

THESIS TITLE	An edge based leaf separation algorithm
STUDENT NAME	Nicha Piemkaroonwong
STUDENT ID	59610038
DEGREE	Master of Engineering
PROGRAM	Computing in Engineering Systems (International Program)
ADVISOR	Dr. Ukrit Watchareeruetai

Abstract

This research proposes a method that separates the region of each leaf from an image of occluded leaves using the edge of the leaves as information, and then produces a set of single-leaf images. To indicate the regions, an intersection point and a direction field are used as edge information and a guideline to extract a leaf inner-edge, which is the boundary that divided occluded leaves. An intersection point, which is defined as a concave point between leaves, is used as the starting position of leaf inner-edge estimation process. A direction field, which describes the average direction of edges in a local area, is used to guide the estimation process. After all estimated regions are identified, they will be used as a set of predefined markers in a watershed transform. The watershed transform is an image segmentation which requires a marker set as an input in order to produce a single-leaf image set as an output.

There were four experiments included in the research. The first one was created for finding the appropriated parameters. The next two experiments were used to evaluate the intersection point estimation process and the proposed method. These experiments applied to the dataset captured in a controlled environment, which consists of 128 occluded leaf images with various species of leaves in the same image. The last

experiment was created to verify the generalization of the proposed method by using another dataset, which was captured in a natural environment, with a total of 30 images with occluded leaves. Experimental result of the proposed method showed that 76.19% of testing leaf images captured in the controlled environment were correctly separated from each other with a segmentation accuracy of 90.96%. When another dataset was applied, the proposed method produced 92.59% result accuracy with a segmentation accuracy of 92.42%.

Acknowledgments

To work on this research topic, it requires a lot of research and trial. Without much support for these people, this research will not become a reality.

First of all, the author would like to sincerely thank the advisor who supports the research, gives an idea, helps an implementation, and provides a lot of materials especially literature of related works, Dr. Ukrit Watchareeruetai. Without him, this proposed method would not be able to complete.

The other grateful supports are given by Asst.Prof.Dr. Chaiwat Nuthong for augmented knowledge and experience, Mathara Rojanamontien for assisting this study, Supanat Kamales for helping the image acquisition, Poomkawin Sihanartkathakul, Purit Thong-on and Chaiwat Wattanapaiboonsuk for proofreading, and Computer Vision Laboratory of International College, KMITL for supporting the research.

Nicha Piemkaroonwong

Contents

List of figures

List of tables

Chapter 1

Introduction

This chapter provides an introduction to this research in order to give the overall idea. It contains the motivation and problem description, the objective, and the scope. Furthermore, it explains the expected benefit and thesis structure of the research.

1.1 Motivation and problem description

Image processing and computer vision are used to change the representation of an image in order to improve its information for human interpretation or make it more suitable for a machine perception [11, 12]. Since, a machine is fast, accurate, and never let its judgment skew its perception. Moreover, it can repeat the same task without dropping computation time and does not have the limitation like a human vision. Thus, the machine with image processing and computer vision is mainly used for improving a quality of human life (e.g., more comfortable). With a lot of its benefits, many research fields adapt computer vision methods for solving their issues in security, medical, transportation, and agricultural fields.

Recently, computer vision techniques have been applied to solve many agricultural issues. For example, a computer vision based method for detecting weeds [21] was

proposed to help a gardener to find and eliminate weeds easily. Additionally, in [17, 23], image analysis methods for identifying the nutrient deficiency in a plant, which is a non-destructive method and treats a plant that requires any nutrient accurately for reducing a dead rate, were proposed. Furthermore, an automatic fruit harvesting method [6] was invented to reduce manual labor of a fruit gardener. Moreover, a computer vision can also be used to automatically identify plants' species based on their leaves [3, 16, 25]. These applications may support a botanist by saving a searching time.

Leaves are parts of a plant that have been extensively used in many of these applications because they contain unique and useful information. Leaves are easier to find, compared to other parts, such as flowers or fruits, which can be found only in a specific season. In order to identify the locations of all leaves in an image, a leaf detection process is required. It is also one of the earliest steps used by many application mentioned earlier.

For the vital process that is used as the early step in a lot of applications, many researchers proposed a leaf detection method. For example, Nguyen et al. [14] proposed a genetic programming based method for detecting rice leaves. Their research determines whether each pixel was rice or not, which means the approach is used to find only the whole area of rice in an image, not an individual. The next study is the research [10] by Liu et al. which describes a strategy to identify overlapped leaves in a surveillance video. The combination of color information and motion direction information is used as a feature map into a convolutional neural network. This research works only when an input is a video, so the weak point is a computation time. Also, Zhang et al.' research [24] identifies each leaf in an image by using the skeleton to estimate its position and direction. The position (i.e., base) indicates a location where a leaf connects with its stem, while the direction (i.e., the direction of midvein) indicates the main distribution from base to apex. Although their work does not separate leaves, it can identify the base and the midvein of each leaf in an image. Another one, the proposed method of Xia et

al. [22] detects leaf boundaries in a pepper's plant image within a controlled environment. It also applies active shape model, which is a deformable template model used to match the same object with different shape variations, to increase the performance of their detection method. This method works only with pepper's species, and the training process is required when a new species is applied.

However, only the result of a leaf detection method might not be enough to use as the early step of some applications which can handle only with a single-leaf image, not a multiple-leaves. Unfortunately, when a leaf photo is taken on a natural background, two or more leaves may be presented in the same input image, and they are sometimes occluded with each other. Therefore leaf separation, which is a process that extracts the region of each leaf from the others, is also required before further processes, in case more than one occluded leaves is found and a set of single-leaf images is demanded.

There is few pieces of research works on leaf separation methods. For example, Frolov et al.' study [7] proposed a method to separate and visualize wheat as a three-dimensional (3D) model. Their work captures wheat on a rotating tray and collects various data in order to reconstruct each wheat as a 3D model. Another one, Janssens et al.' research [9] automatically separates leaves' region from rosette plant. It locates stems' location in an image and eliminates the stems. Then, it extracts each leaf from the image. Unfortunately, both researches work with only a single specific species (i.e., Frolov et al. for wheat and Janssens et al. for rosette). Furthermore, every image in their dataset has to be captured with a specific location of cameras facing a specific angle. To be more specific, the first dataset was collected from three fixed cameras with different angles to generate the model, while another one was obtained from the top-view. Moreover, Frolov et al.' method spends a lot of time to work the whole process because it deals with 3D model. To manipulate their limitation, our proposed method is contributed. It can be used to separate occluded leaves with various plants' species in a 2D image. It can also handle when occluded leaves with different angles were applied.

This research proposes a method that separates leaves from a single input image using edge information of the leaves. The proposed method takes an image of occluded leaves as an input and produces a set of single-leaf images as an output. There are four main steps used in the proposed method. First, the outer contour of leaves' area is extracted. Second, the curvature of the contour is analyzed to locate intersection point(s) between leaves. An intersection point is used as a starting position for other processes. Third, the proposed method applies the intersection points to preprocesses the input image and then estimates its direction field. The direction field determines the average direction of edges in a local area. At this step, a part of each leaf can be roughly identified using the intersection points and the direction field. Finally, a watershed transform is used to segment the region of each leaf from the others, as shown in Fig. 1.1. In summary, this proposed method creates an initial marker set as the guideline for the watershed transform.

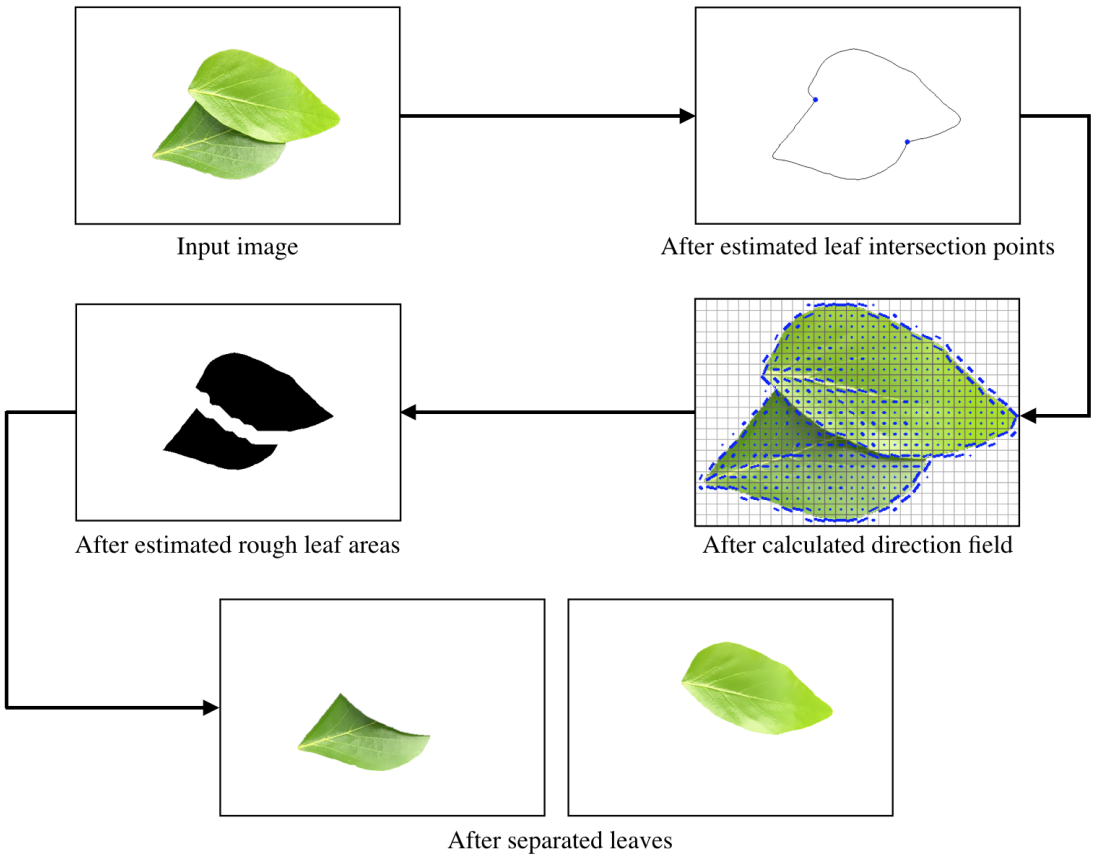


Figure 1.1: System overview

1.2 Objective and scope

The sole objective of this research is to study and develop an edge-based approach for an automatic leaf separation method that can reduce a simple routine task of people, such as manually segmenting occluded leaf images by hand.

This research focuses on separating occluded leaves with a plain background. Moreover, each leaf must be a simple leaf with non-palmate.

The expected benefit of the research is to assist an agricultural application that used computer vision techniques. After the system produced the output images, the images

can be used in many applications, which use a single leaf image with a plain background as an input such as plant nutrient deficiency detection, plant disease diagnosis, and leaf identification system. Moreover, the research can support people who want to use a segmented leaf image without manual labor.

1.3 Thesis structure

The remaining of the thesis is organized as follows. Chapter 2 determines all necessary background knowledge for understanding the system. It also provides related works. Chapter 3 describes the proposed method for leaves detection and segmentation in detail. Experimental results and discussion are explained in Chapter 4. It also includes the image acquisition. The final chapter, Chapter 5, concludes the research and gives some ideas for future work.

Chapter 2

Background knowledge and literature reviews

This chapter provides four topics as background knowledge, which are leaf structure, bilateral filter, morphological operator, and watershed transform. Additionally, it consists of four literature reviews. The first research is a 3D wheat's plant separation, while the second one is a leaf segmentation from a complicated background. The third one is to extract a subjective contour, and the last one is a rosette leaf separation.

2.1 Background knowledge

2.1.1 Leaf structure

Leaves are vital organs in a plant because they are mainly responsible for food production using photosynthesis. Leaves can also be found in most of the plants and almost every season. Moreover, they are significant to organisms for the reason that they can be a source of foods and medicines. Thus, many leaves are benefiting, but some leaves are toxics. Therefore, it is essential to distinguish the species of a plant and know a shared

knowledge of the structure of a leaf because its leaf can be used to identify the plant's species.

There are five main features applied for the species' indication, which are base, apex, veins, blade, and margin. Figure 2.1 illustrates the components of a simple leaf [4, 5]. The base is the part of the leaf that attaches to a stem, while the apex is the opposite part of the base where the midvein ends. If the base is the lowest area that closes to the stem, the apex is the highest area that farthest to the stem. The midvein or primary vein is the main structure of veins that is analogous to the trunk of a tree. It commonly runs from the base to the apex. Secondary veins are analogous to a fork of the tree. They ordinarily run from the base or midvein toward the margin, which is the outer boundary of the lamina. The lamina, also called blade, is leaf's surface. It is the major part of photosynthetic in which its color normally is green (the dark green in Fig. 2.1).

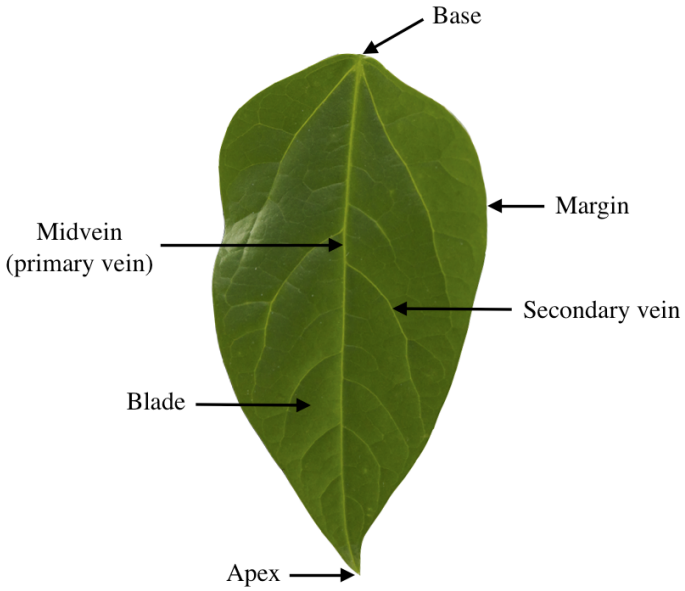


Figure 2.1: Structure of a leaf

This research uses information from the shape of a simple leaf with non-palmate. The simple leaf denotes a leaf that contains a single lamina attached to a stem. When

the simple leaves are occluded, an intersection point(s) between leaves should be found. Thus, this research applies this knowledge as a guideline to separate occluded leaves.

2.1.2 Bilateral filter

Bilateral filter [19] is a blur filter that reduces noise without destroying edges. Figure 2.2 represents an example when convoluted the filter.

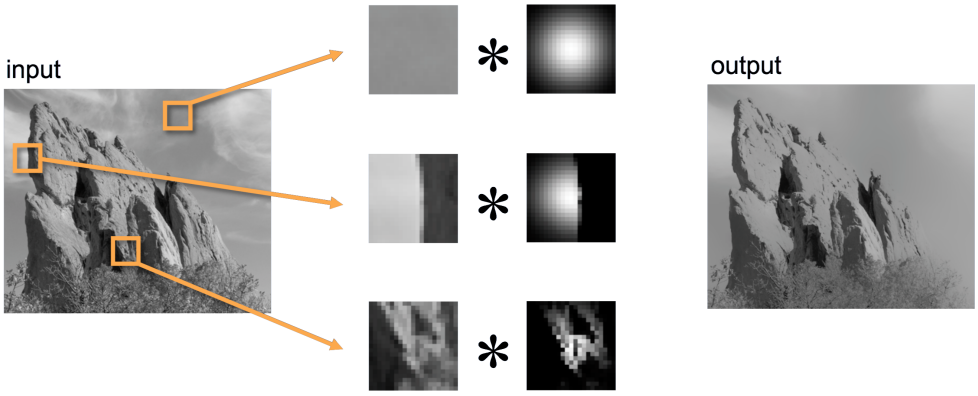


Figure 2.2: Bilateral filter example (source: [15])

It is designed based on the Gaussian filter in which each pixel's intensity is replaced by the weighted average intensity of neighbor's pixels. The difference between Gaussian blur filter and bilateral blur filter is that Gaussian does not concern about edge preservation, while bilateral has no averaging across an edge. The formula of bilateral is given by:

$$\begin{aligned} BF[I]_{\mathbf{p}} &= \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |) I_{\mathbf{q}}, \\ W_{\mathbf{p}} &= \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |), \end{aligned} \tag{2.1}$$

where $W_{\mathbf{p}}$ denotes a normalization factor in a range of 0 to 1, S denotes the window centered in $\mathbf{q}$, G_{σ_s} denotes a spatial kernel (i.e., size of considered neighborhood), G_{σ_r}

denotes a range kernel (i.e., minimum amplitude of an edge), $\mathbf{I_p}$ denotes an intensity of pixel $\mathbf{p}$, and $BG[I]_\mathbf{p}$ denotes the new intensity value of $\mathbf{p}$ after applied bilateral filter, as shown in Fig. 2.3.

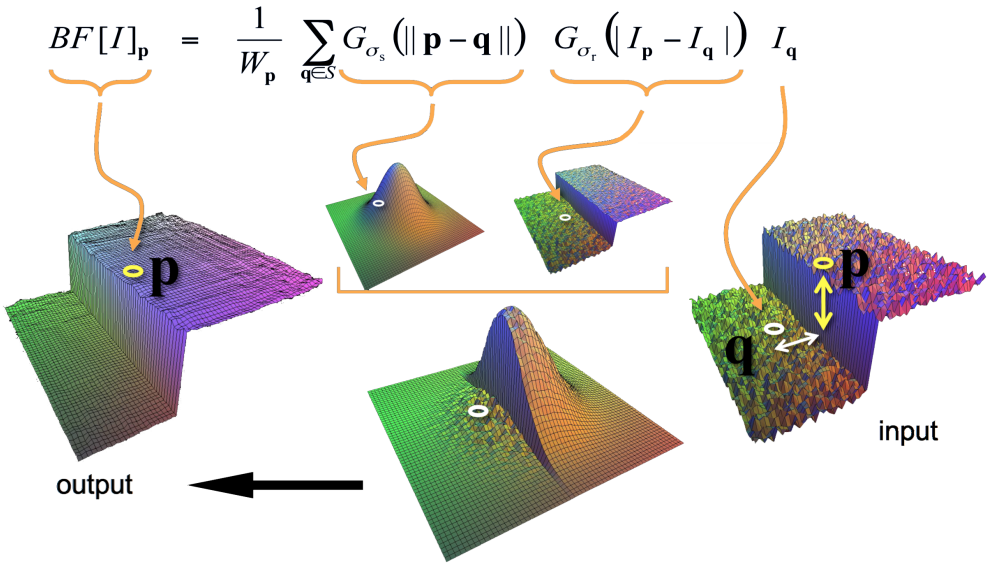


Figure 2.3: Bilateral filter explanation (source: [15])

2.1.3 Dilation and erosion morphological operators

Morphological operator [8] is usually used in a digital image which transforms a neighborhood into different forms such as thinning, pruning, or thickening. Most standard operators in morphology are dilation and erosion morphological operators. These operators are a binary operator.

Dilation is applied for thickening every foreground region in an image. A dilation of an image I with the structure element H is computed as follow:

$$I \oplus H = \{(\mathbf{p}+\mathbf{q}) | \mathbf{p} \in I, \mathbf{p} \in H\}, \tag{2.2}$$

where $+$ presents a translation operation.

Figure 2.4 represents an image with dilation morphological operator. Pixels around the region are expanded with the four neighborhood. Noted that a yellow pixel is a transformed neighbor's pixel.

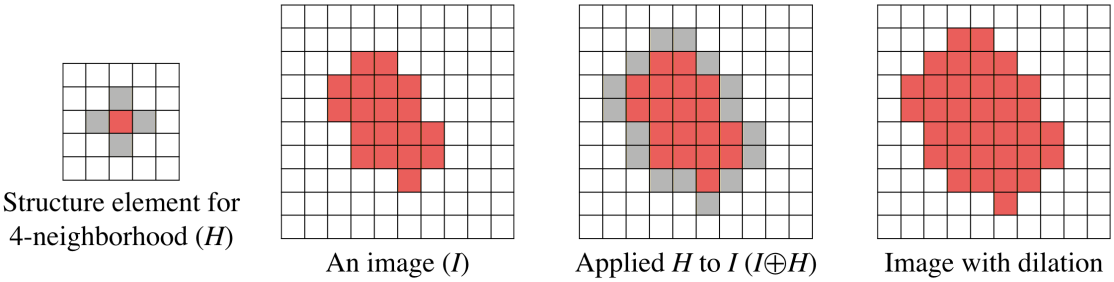


Figure 2.4: Result image when applied dilation with 4-neighborhood

Erosion is applied for shrinking every foreground region in an image. An erosion of an image I with the structure element H is denoted by:

$$I \ominus H = \{\mathbf{p} \in \mathbf{Z}^2 | (\mathbf{p}+\mathbf{q}) \in I, \text{for every } \mathbf{q} \in H \} \tag{2.3}$$

Figure 2.5 represents an image with erosion morphological operator. Border's pixels in the region is eroded with the four neighborhood.

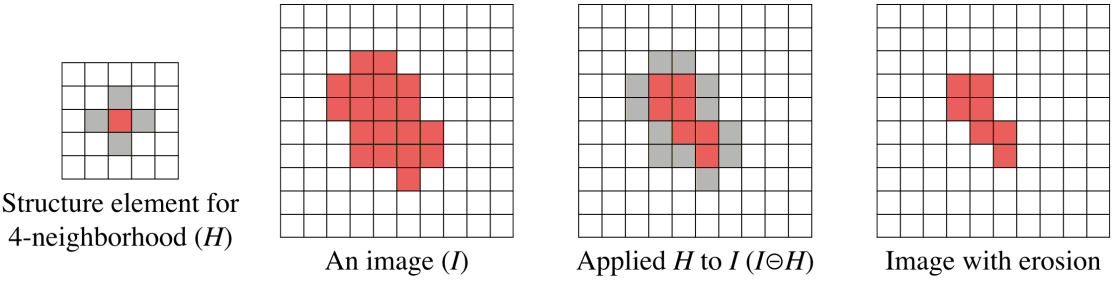


Figure 2.5: Result image when applied erosion with 4-neighborhood

2.1.4 Watershed transform

Watershed transform [2] was originally proposed by Beucher et al. in 1979. It is a morphological method used to extract the contours from an image. It is also one of the well-known algorithms for image segmentation because it does not require any threshold values. In geography, watershed transform means an area of land that divided waters flowing to the different river systems. The watershed transform tries to find two things: 1) catchment basins and 2) watershed ridgelines.

The concept of the watershed transform is to find the watershed lines. Assumed that water is flooded from each deepest region in a topographic surface with the same constant rate so that the water level will be increased in the area surface uniformly. When the rising water in the different regions is going to merge, the dam is built to prevent blending water. After the level of water reached all dam boundaries, the flood is stopped. Each remaining boundary is the watershed line. Each water area is the catchment basin. Applied with image processing, an image can be defined as the topographic surface, catchment basin corresponds image region, and watershed line corresponds region boundary.

This method works well when it is applied with a plain image. However, oversegmentation is appeared after applied with more complex image, as shown in Fig. 2.6.

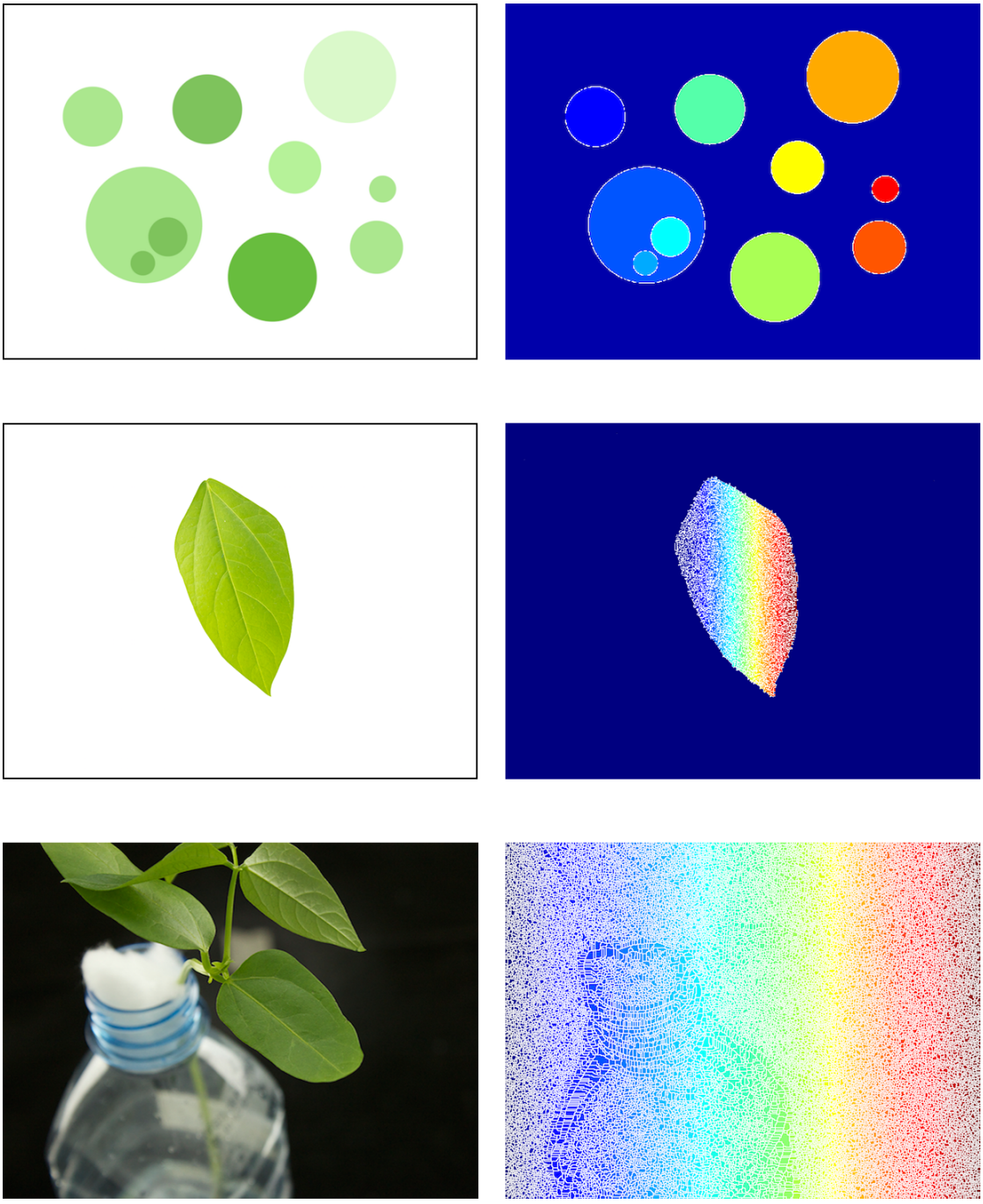


Figure 2.6: Input images (left) and result images of watershed transform (right). White color boundaries represent watershed lines. Color areas represent catchment basins.

In 1992, Meyer introduced an improved version of watershed transform [13] to avoid

over-segmentation by adding more input information called markers. Supposed that objects of interest and background are known, then the holes are bored only these positions before flooding. Each hole depicts a marker which is the deepest region in a topographic surface. After all markers were predefined, the first step of the watershed transform can be applied. In the first row of Fig. 2.7, two markers are applied with an image, one for leaf and one for background, to produce flood yielding the perfect result. However, the second row image is not correct because of predefined markers. Thus, the appropriate markers are significant for watershed transform by Meyer.

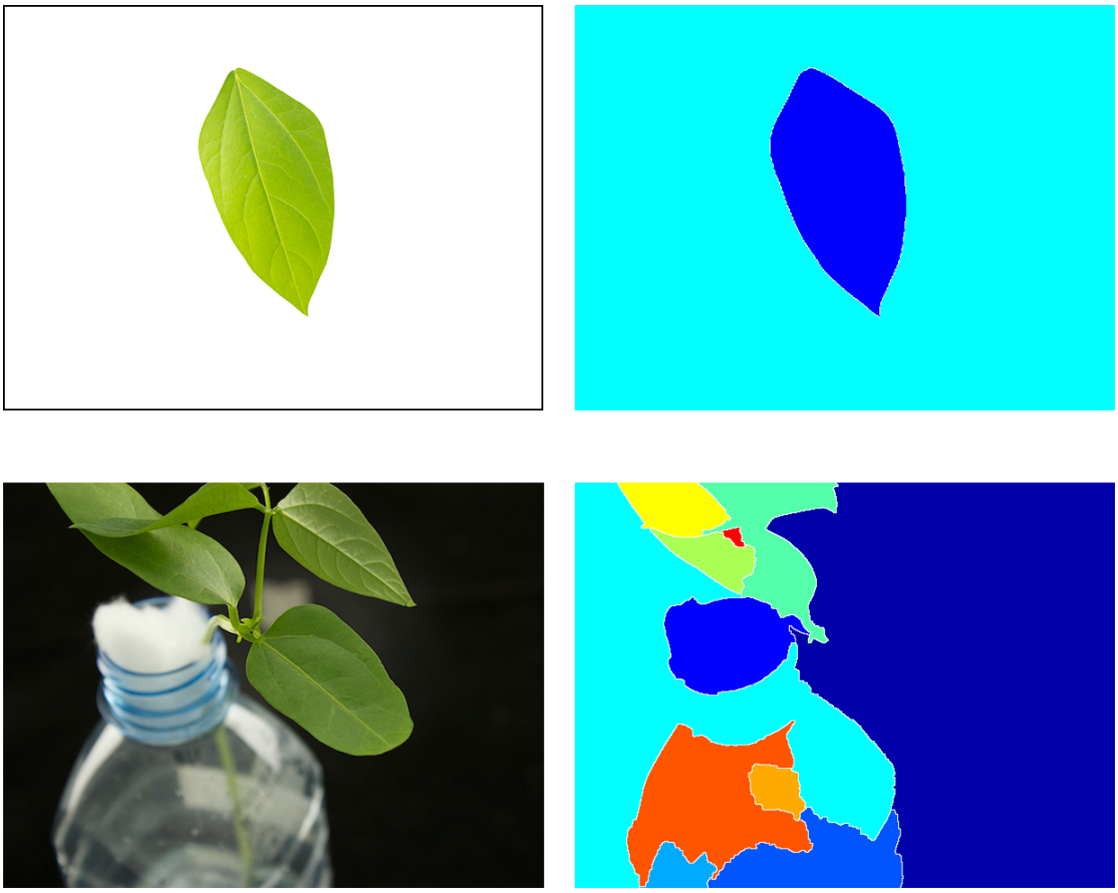


Figure 2.7: Input images (left) and result images of watershed transform by Meyer (right)

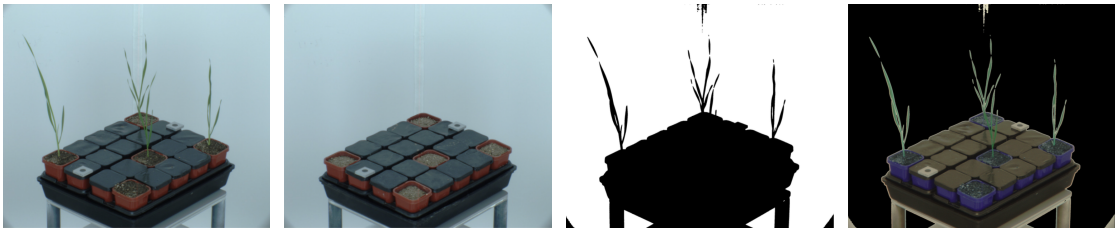
2.2 Literature reviews

This section is covered on the related work, which is about leaf separation, and other similar fields. Recalled that leaf separation is a process used to divide occluded leaves to a set of single leaves from an image.

2.2.1 3D wheat's plant separation

Frolov et al. proposed an automated wheat's plant separation in 3D [7]. There are three cameras used in this work: left side, right side, and top side, which automatically captured an empty tray and the tray with three to 10 wheat, totally 200 trays, and rotated every three degrees (360 images per tray). Other data, such as multi-wavelength, optical-imaging sensor, and light detection and ranging (LiDAR), were collected at the same time.

3D plant reconstruction begins with background subtraction using silhouette extraction. The tray with plants are subtracted from its empty tray, and then a set of masks is produced, as shown in Fig. 2.8.



(a) A wheat image (b) Its tray image (c) Background mask (d) Non background

Figure 2.8: Result of silhouette extraction from the top camera (source: [7])

Next is to generate a 3D convex hull volume from each mask with space carving algorithm. The convex hull is defined as a set of points in a plane, while the space carving algorithm is an algorithm that map images with different angle into a 3D convex hull volume. However, this algorithm is only produced a 3D discrete scalar field, but

Frolov el at.' research creates an application with 3D polygon mesh. Therefore, a standard marching-cube algorithm is performed to obtain the 3D polygonal mesh from a 3D discrete scalar field (3D convex hull volume). Then, the 3D mesh set is combined and converted to a triangular mesh. The tray with plants mesh is subtracted by the empty tray mesh to obtain only wheat's mesh using a mesh subtraction (see Fig. 2.9).

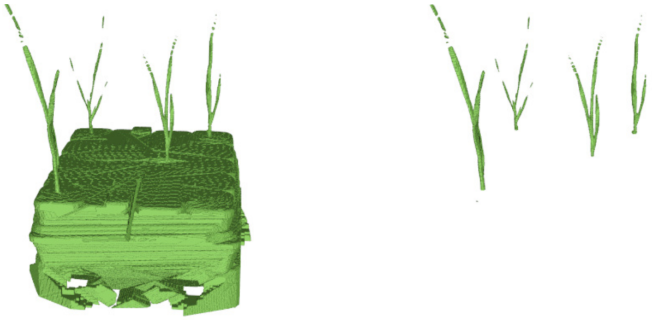


Figure 2.9: Mesh subtraction between plant mesh and tray mesh (source: [7])

Currently, the 3D wheat's plants in each tray are obtained. However, which mesh belong to plant does not define yet, and some leaves may be occluded each other. Thus, to separate plant, a control point is determined.

The control point (see Fig. 2.10) is a centroid point obtained from cutting the mesh in the horizontal plane at different height levels. For a horizontal leaf, the mesh is cut in the vertical plane instead. That means cutting in the horizontal plane for a vertical leaf and cutting in the vertical plane for a horizontal leaf.

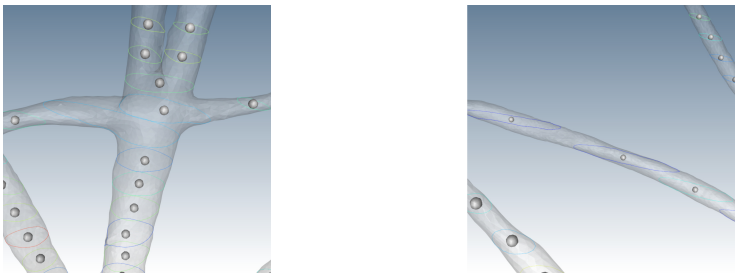


Figure 2.10: Example of control points (source: [7])

Unfortunately, a control point may not locate in the real center of a leaf. Assume that

a leaf shape plane is ellipsoid due to a nature of wheat, so every point does not satisfy the following equation will be eliminated, $D_{max} \div D_{min} > T$, where D_{max} and D_{min} are maximum and minimum distances from the center of the control point on a plane, and T is a constant threshold. Each plant is created by a combination of consecutive points, and it is attached the unique color as a label, as shown in Fig. 2.11.

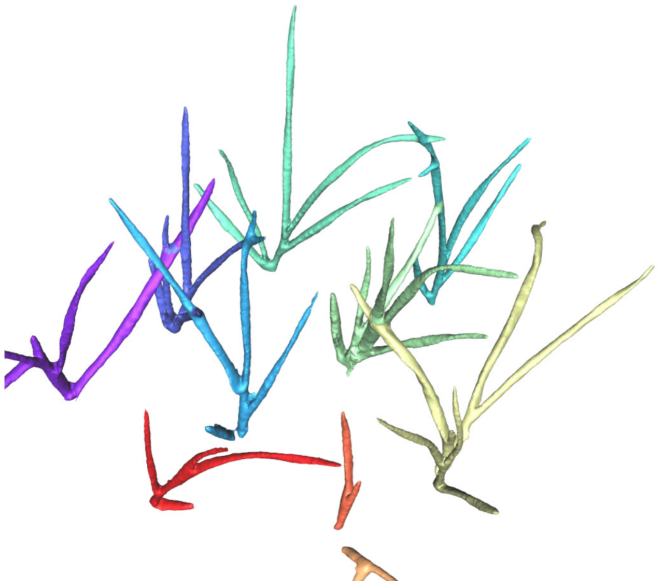


Figure 2.11: Result of Frolov et al.' research (source: [7])

This work can separate 3D mesh of wheat in a tray, but the more computation time is required when a task is solved in 3D.

2.2.2 Leaf segmentation from a complicated background

Next related work is a leaf classification with complicated background proposed by Wang et al. [20]. It is a classification technique that involves leaf extraction from a complicated environment. This research concentrates on the preprocessed part which is leaf extraction.

Watershed transform is a region growing technique to segment a marked object's

area from the others. As mention in Section 2.1.4, it requires a marker to avoid over-segmentation, so this method automatically creates markers for watershed transform.

Otsu's method is picked as the first step of marker construction. It is a threshold selection that selects the threshold value with a minimum variance in the class $\sigma_w^2(t)$.

$$\sigma_w^2(t) = P_1(t)\sigma_1^2(t) + P_2(t)\sigma_2^2(t), \tag{2.4}$$

where P_1 and P_2 are the probabilities of two classes separated by a threshold t, and σ_1^2 and σ_2^2 are variances of their class.

A segmented binary image is obtained by using Otsu's method; then erosion morphological operator is applied to remove some noise and separate occluded objects. However, the current image may contain more than one region, the maximum one is chosen and used as a marker for watershed transform.

Figure 2.12 represents the steps of leaf extraction. The first column (left most) describes input images, second denotes images with Otsu thresholding, next is images with erosion, and the last one determines the result of leaf extraction method.

Figure 2.12: Leaf extraction from leaf image with complication background (source: [20])

The disadvantage of this method is it can extract only one leaf. Moreover, when the maximum region is not a leaf, watershed will segment the wrong object as an output.

2.2.3 Subjective contour extraction

Another research proposed by Teranishi et al. [18] is applied to automatically generate subjective contour from an image. Their work is used to find all outer contours, detect points in the contour which are the intersection point between objects, connect the points together to fill a gap between points, as shown in Fig. 2.13.

Figure 2.13: Input and output images for subjective contour extraction (source: [18])

There are five steps in this algorithm which are in the following:

1. Extract all outer contours in an image

2. Identify the intersection points which are the points where a curvature on the contour suddenly changes (yellow point) or a endpoint of the contour is found (blue point), see Fig. 2.14.

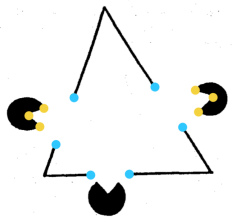


Figure 2.14: Intersection points for subjective contour extraction (source: [18])

3. Add an arrow to each intersection point. The arrow represents a direction of subjective contour. The direction of a yellow point is declared on the left side and the right side of its point, while the direction of a blue point is an orthogonal line of its point. Noted that the contours may be located on the left side of the direction of arrows, as shown in Fig. 2.15.

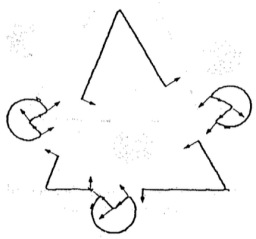


Figure 2.15: Points with its arrow for subjective contour extraction (source: [18])

4. Determine a pair of connected points $f(i, j)$, which is calculated as follows.

$$f(i,j) = k(cos\theta_i + cos\theta_j) - d_{ij}, \tag{2.5}$$

where k denotes positive constant, θ_i and θ_j denote counter-clockwise angle between an arrow and a connected line of two points, and d_{ij} denotes their distance. For each S_i, S_j that has maximum value of $f(i, j)$ is selected as connected line of subjective contour. If the value is less than threshold, it means S_i does not have any pair.

5. Create a line to connect a pair of points in order to produce subjective contour using cubic spline function.

Figure 2.16 describes the result after applied their proposed method to an occluded leaves. Although the method can be divided the occluded leaves accurately, it does not define which leaf is overlay the another. Moreover, this research does not provide the

(a) Input image (b) Output image

Figure 2.16: Result example after applied Teranishi et al.' study (source: [18])

experimental result, so there is no guarantee that this work is really work when other images were applied.

2.2.4 Rosette leaf separation

This related work was proposed by Janssens et al. in 2013 [9]. It provides a method for automatic leaf separation from a plant with a circular arrangement of leaves, rosettes. It also has an algorithm to extract the line of symmetry of the leaf, and computes in parallel. However, our research focuses only the first one which is leaf separation as a literature review.

There are three main processes in leaf separation which are 1) preprocessing, 2) stems' position identification, and 3) leaf extraction. An image is taken with chlorophyll fluorescence in a controlled environment. Then, it is converted to the binary image, segmented by a threshold, and extracted contour, as shown in Fig. 2.17.

(a) (b)

Figure 2.17: Chlorophyll fluorescence image (left) and contour image (right) (source: [9])

Next process is to identify the position of stems. Assumed that pixels of a stem lie close to each other, but they are not close together in the contour list. To identify a stem pixel, each pixel in the contour is compared to the others by computing 1) manhattan distance and 2) the absolute distance in the contour list. The red point in Fig. 2.18 represents the stem pixel. Then, consecutive points are grouped together. Supposed that each stem contains two groups, so the point that closet to a leaf, interior point, and the point that furthest away from the leaf, exterior point of two groups are connected.

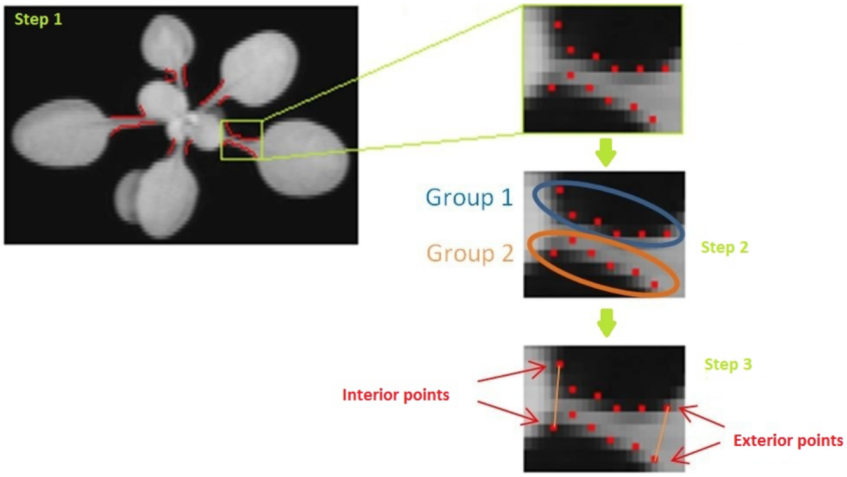


Figure 2.18: Stems' position identification process (source: [9])

Leaves are separated by linking the interior points together and linking the exterior points together, then other areas in the image are extracted and are assigned as leaf image, as shown in Fig. 2.19.

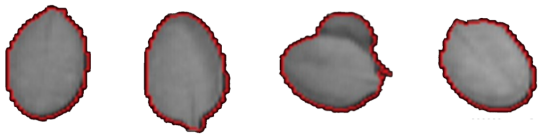


Figure 2.19: Results of leaf extraction (source: [9])

Although this method is clearly separate leaves, there are some limitations of the method. First, only rosettes plant can be applied with the method. Second, if a stem

is hidden, its leaf will be automatically combined with another leaf as a single leaf, so the accuracy will be dropped when the taken plant grows (i.e., too many leaves in the plant). Third, it works only image with bird's-eye view.

Chapter 3

Methodology

In order to produce a set of single-leaf images from an occluded leaf image, intersection points between leaves and a direction field are required. These intersection points are used 1) to define a marker for applying a smoothing filter in a preprocessing step and 2) as a starting position of leaf region estimation process. The direction field of the input image is computed and also used to estimate the leaf regions. After leaf region estimation is done, it produces foreground markers, which are rough leaf regions, as an output. Then a background marker is generated in leaf separation process. Next, watershed transform is applied by using these markers, the foreground and background markers. Finally, the proposed method produces a set of leaf images, one leaf per image, with a plain background as output. The block diagram of the proposed method is shown in Fig. 3.1.

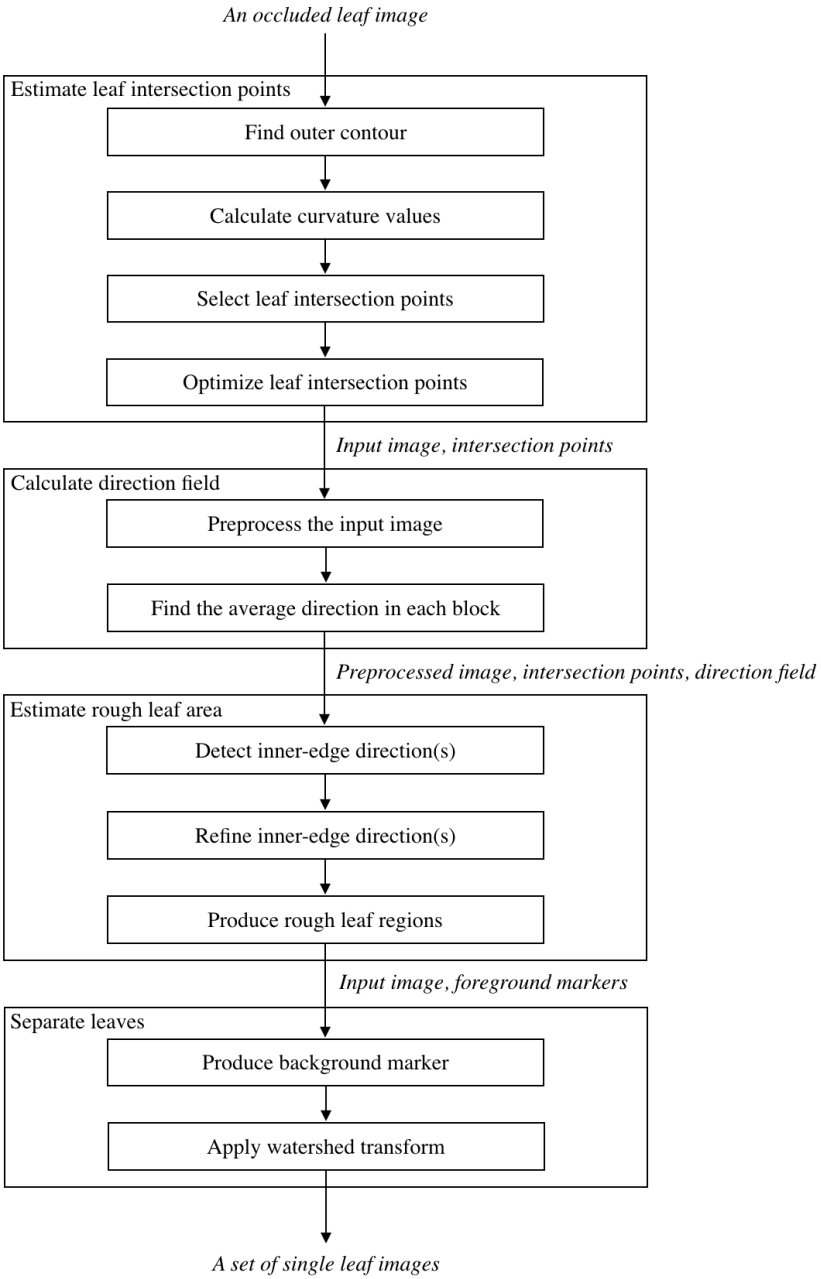


Figure 3.1: Block diagram of this proposed method

3.1 Leaf intersection point estimation

The outer contour of an input image is extracted and analyzed to find intersection points between leaves. The intersection points will be used as a guideline of the next processes (i.e., calculate direction field and estimate rough leaf area). To estimate them, every concave point in the contour is identified. The concave point is a point that curves bend toward inside a contour. It is identified by curvature value, which determines how rapidly a curve changes. It is computed using formula described in [1].

$$\kappa(u) = \lim_{u \to 0} \frac{\phi}{u}, \tag{3.1}$$

where κ is a curvature of the curve C, ϕ is an angle between tangent vector u and tangent vector u over t units, and u is an arc length parameter.

Supposed that the curve is a parametric vector $C(u) = (x(u), y(u))$, so the curvature κ becomes

$$\kappa(u) = \frac{x'(u)y''(u) - x''(u)y'(u)}{(x'(u)^2 + y'(u)^2)^{\frac{3}{2}}} \tag{3.2}$$

When each component is convoluted with a 1D Gaussian kernel $G(u, \sigma)$ of width σ, the resulting curve X and Y are:

$$\begin{aligned} X(u,\sigma) &= x(u) * G(u,\sigma), & Y(u,\sigma) &= y(u) * G(u,\sigma), \\ X_u(u,\sigma) &= x(u) * G'(u,\sigma), & Y_u(u,\sigma) &= y(u) * G'(u,\sigma), \\ X_{uu}(u,\sigma) &= x(u) * G''(u,\sigma), & Y_{uu}(u,\sigma) &= y(u) * G''(u,\sigma) \end{aligned} \tag{3.3}$$

Finally, the curvature of the convolved curve can be formulated as:

$$\kappa(u) = \frac{X_u(u,\sigma)Y_{uu}(u,\sigma) - X_{uu}(u,\sigma)Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2)^{3/2}}, \tag{3.4}$$

To reduce the processing time, points on the contour are down-sampled by $sampling\_num$.

The curvature value of each point is calculated using the above formula. A positive curvature value indicates a concave point, which is a candidate intersection point between two objects. The higher positive value means a curve will bend more toward inside the contour. A set of intersection points is chosen based on a threshold selection. Only a candidate point with the curvature value higher than the threshold value will remain.

A blue point in Fig. 3.2 represents a candidate intersection point in a close-up perspective. From Fig. 3.4b, when two leaves are overlapped, the number of intersection points should not be more than two. Unfortunately, there are too many points (i.e., more than two) in the image, and some points are too close to each other. In order to reduce unnecessary points, the points which are too close to the others will be merged into a single point. They are merged by finding a group of points and selecting the middle one as an intersection point. The threshold distance d is computed as follows:

$$d = \sqrt{(p_{2x} - p_{1x})^2 + (p_{2y} - p_{1y})^2}, \tag{3.5}$$

where p_1 and p_2 are consecutive intersection points.

The points are grouped together if d between them is smaller than $sampling\_num \times 2.5$. Then, the middle one from a group is chosen as a leaf intersection point, as shown in Fig. 3.2. Blue circles in Fig. 3.4c are the middle points of each group.

Figure 3.2: Intersection point combination

There are only two intersection points remaining, but these points are not in the real

position of the intersection points. To optimize their location, an intersection point p is shifted to the position which has the smallest angle between points. Algorithm 1 describes this step in detail.

Algorithm 1 Shift to the smallest angle

for each p in a set of intersection points **do**
  for $round = 4 \times sampling\_num \rightarrow 0$ **do**
    con_s = the outer contour down-sampled by $round$
    if p is not in con_s **then**
      Insert p in con_s
    end if
    ang_l = an angle between $con_s(p-1)$ to its left and right points, $\angle con_s(p-2)con_s(p-1)con_s(p)$
    ang_c = an angle between $con_s(p)$ to its left and right points, $\angle con_s(p-1)con_s(p)con_s(p+1)$ (see Fig. 3.3)
    ang_r = an angle between $con_s(p+1)$ to its left and right points, $\angle con_s(p)con_s(p+1)con_s(p+2)$
    p = a position with $min(ang_l, ang_c, ang_r)$
    $round = round - 1$
  end for
end for

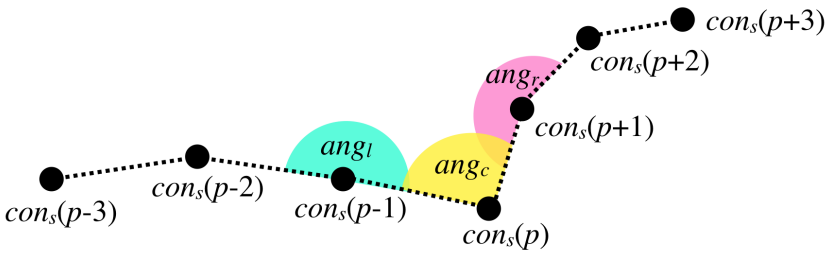
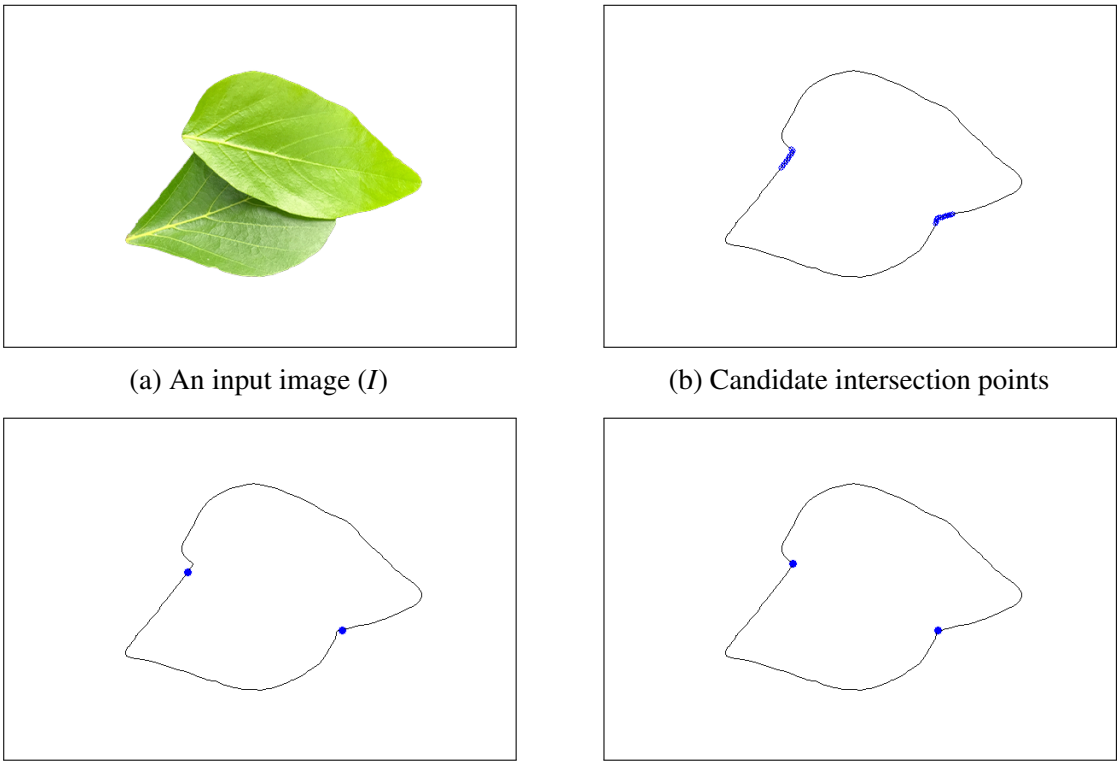


Figure 3.3: Angle calculation

In Algorithm 1, for each intersection point p, the contour is down-sampled by the large value to the smaller value in each iteration (e.g., iteration #1: down-sampled by eight, iteration #2: down-sampled by seven, until zero, which means no sampling). The reason for starting with the larger sampling value is to avoid the local optima of a

concave position. Then p is shifted to the left position in the sampled contour con_s if the angle between $con_s(p-1)$ to $con_s(p)$ and $con_s(p-1)$ to $con_s(p-2)$ is the narrowest angle compare with the others (i.e., the current angle and the right one). If the left, current, or right angle is the narrowest angle, then p is shifted to this point. Next, the next iteration begins until *round* is zero.

The blue circles (see Fig. 3.4d) denote intersection points, which are the result of leaf intersection point estimation.



(a) An input image (I)

(b) Candidate intersection points

(c) Merged intersection points

(d) Shifted intersection points

Figure 3.4: Result of leaf intersection estimation

3.2 Direction field calculation

After obtained the intersection points from the previous process, direction field calculation process begins. There are two main steps used in this process. The first step is

to create a preprocessed image, which is a noise reduced version of an input image, from the input image and the intersection point. The second one is to formulate a direction field, which is used as a guideline to estimate leaf regions, on the preprocessed image. A direction in each block comes from the average direction of edges in that block.

3.2.1 Preprocessing

An image preprocessing process is aimed to improve image data and to make the image more suitable for a specific task. It can be used in various ways, such as image sharpening, image compression, and image blurring.

In this research, the preprocessing is a process applied for blurring an input image. To enhance the edge information, unnecessary information such as other leaf areas is blurred. Bilateral filter, see Section 2.1.2, is selected in order to reduce noise without destroying edges as a blur filter. However, information of leaf boundary may be lost if the whole image is blurred, so every pixel except a background is blurred once in order to completely prevent this loosing information of the leaves' boundary.

Unfortunately, some edge information, such as a midvein and veins, is not totally eliminated because of too strong intensity between the veins and the blade, so these irrelevant information should reduced by blurring only its area. Reminded that the objective of this proposed method is to construct an initial markers for a watershed transform in which each marker represents the rough region of a leaf. Additionally, the regions of a leaf can be separated by using a leaf inner-edge, the line that divides occluded leaves into a set of single leaves. Therefore, if the location of leaf inner-edge is known, the location of other irrelevant information will be known too. Since other areas of the interested area (i.e., leaf inner-edge) are uninterested area.

From these assumption, a blur filter mask M_s is introduced, which is created for blurring pixels that are not a leaf inner-edge area. Supposed that the inner-edge is a straight line between intersection points. All combinations of points are drawn as lines

in M_s. Since a leaf edge is not exactly linear, all lines in M_s are thickened using dilation morphological operator Section 2.1.3. Then, every pixel which is not in the mask is blurred again by using bilateral filter, as shown in Fig. 3.5.

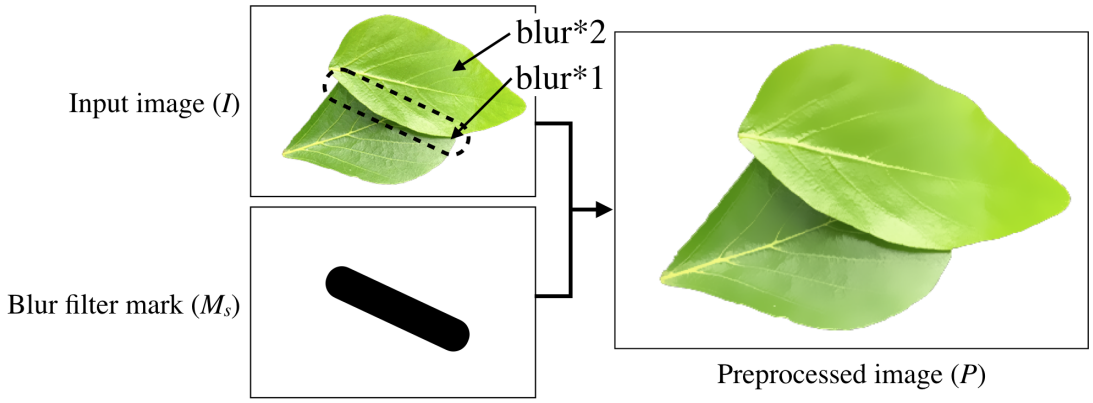


Figure 3.5: Preprocessing steps

The summarized explanation is described in the following.

$$P(x,y) = \begin{cases} I(x,y) & \text{if } I(x,y) \text{ is a background pixel,} \\ B_k(I(x,y)) & \text{if } M_s(x,y) \text{ is a masked pixel,} \\ B_k(B_k(I(x,y))) & \text{otherwise,} \end{cases} \tag{3.6}$$

where, P indicates the preprocessed image, I indicates the input image, M_s indicates the blur filter mask image, B_k indicates the bilateral filter applied to I, and (x,y) indicates a pixel position.

3.2.2 Direction detection

Suppose that all unnecessary information are already removed from a preprocessed image. The next step is to calculate a direction field in a direction detection, which is a method deployed to estimate the edge direction in a local area. There are three main

steps in the direction detection: 1) create a gradient image, 2) calculate an orientation field, and 3) formulate a direction field.

The gradient image describes a direction change of intensity in an image. It usually be used for edge detection. When a gradient image g with Sobel operator in preprocessed image P is approximated, the result will be shown in Fig. 3.6.

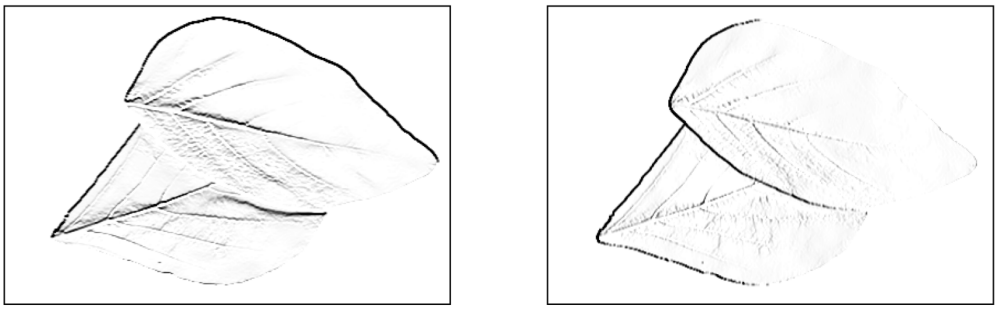


Figure 3.6: Gradient image for x-derivative (g_x) and y-derivative (g_y)

Sobel operator is a derivative filter mask used for edge detection. It consists of two masks which are the mask for x-derivative and the mask for y-derivative. Furthermore, its operator contains 3×3 convolution kernels, as shown in Fig. 3.7. These kernels are the same when one kernel is rotated by 90°

-1	0	1
-2	0	2
-1	0	1

(a) g_x

-1	-2	-1
0	0	0
1	2	1

(b) g_y

Figure 3.7: Sobel convolution kernels

Secondly, the orientation θ is computed in every block with size bs in the gradient

images. It represents the flow of information using an edge image, which is given by:

$$\theta = tan^{-1}\left(\frac{\sum_{i}^{bs}\sum_{j}^{bs} 2g_x(i,j)g_y(i,j)}{\sum_{i}^{bs}\sum_{j}^{bs}\left(g_x^2(i,j) - g_y^2(i,j)\right)}\right), \tag{3.7}$$

where (i, j) is a pixel in a block.

Finally, a direction field for a block is formulated by using g and θ. It describes the average edge direction of the block. There are two values representing the direction which are 1) angle and 2) magnitude. The angle determines where the edge direction go, whereas the magnitude determines how strong of the edge. The formula of a direction field is explained by:

$$\begin{aligned}
d_+ &= tan^{-1}\left(\frac{g_y(i,j)}{g_x(i,j)}\right),\\
d_- &= (d_+ + 180^\circ) \mod 360^\circ,\\
v_x &= \frac{1}{bs^2}\sum_{i}^{bs}\sum_{j}^{bs}\begin{cases} g_x(i,j) & \text{if } \theta - d_+ < \theta - d_-,\\ -g_x(i,j) & \text{otherwise.}\end{cases}\\
v_y &= \frac{1}{bs^2}\sum_{i}^{bs}\sum_{j}^{bs}\begin{cases} g_x(i,j) & \text{if } \theta - d_+ < \theta - d_-,\\ -g_x(i,j) & \text{otherwise.}\end{cases}\\
\vec{D} &= < v_x, v_y >,\\
\angle\vec{D} &= tan^{-1}\left(\frac{v_y}{v_x}\right),\\
|\vec{D}| &= \sqrt{\left(v_x^2 + v_y^2\right)},
\end{aligned} \tag{3.8}$$

where (i, j) denotes a pixel in a block of g, d_+ and d_- denote a positive and negative direction, v_x and v_y denote vector components in x-axis and y-axis of $\vec{D}$, and $\vec{D}$ denotes the direction field of each block in g.

Figure 3.8 represents the result of this process which is direction field calculation. The blue line in each block indicates the average edge direction of that block. Different directions of a block explain different values of the angle. Moreover, the strong magnitude means its own block contains higher different intensities between pixels. The stronger magnitude, the longer line.

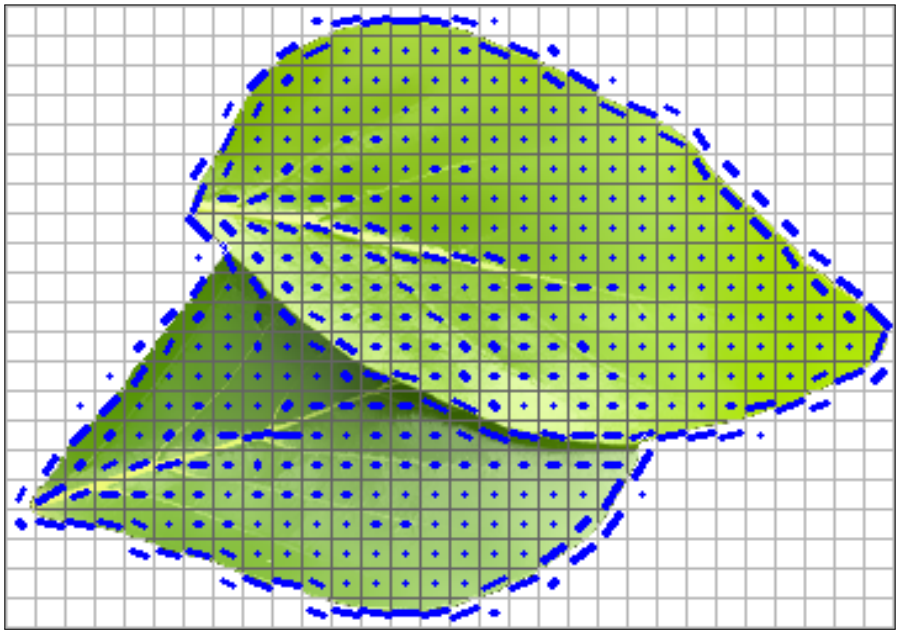


Figure 3.8: Image with direction field

3.3 Rough leaf area estimation

Rough leaf area estimation process is a process to approximate a line that divides overlapping leaves into a set of rough separated leaves. It requires intersection points and a direction field as a guideline to make a decision of the direction of inner-edge. Moreover, it uses the preprocessed image as an input image. There are three main steps in this process, rough leaf area estimation, which are 1) leaf inner-edge estimation, 2) inner-edge refinement, and 3) rough leaf region identification. The first step is used to identify a leaf inner-edge. Sometimes, the inner-edge, which is a set of points, contains

irrelevant information, so inner-edge refinement is applied in order to remove these irrelevant points. After refined inner-edge, the performances of result accuracy and computation time should be increased. Lastly, rough leaf region identification step is used to create a set of foreground markers for a watershed transform.

3.3.1 Leaf inner-edge estimation

As already mentioned, leaf inner-edge estimation is a step to estimate a leaf edge by using an intersection point as a guideline for starting position and then using the direction field as a guideline for the rest.

Algorithm 2 and Fig. 3.9 represent the starting inner-edge position in detail. The preprocessed image P is divided into a block with size bs (same as the previous process). Then the outer contour con is extracted from occluded leaves. To identify a starting point, directions of con on the left (d_l) and on the right (d_r) are indicated (see line #1 or Fig. 3.9a) from the assumption that the short line of a leaf is approximately linear. If it is linear, when d_l and d_r flipped by 180° will be a direction of leaf inner-edge. b_{d_l} and b_{d_r} are the neighbor's blocks that d_l and d_r are lain, respectively. These blocks are a candidate of starting positions. Each intersection point must contain only one inner-edge direction, and the direction should be the block which contains the highest different intensity. Therefore, if the magnitude of b_{d_l} is higher than the magnitude of b_{d_r}, b_{d_l} is selected as a starting block, vice versa.

Noted that a blue point (Fig. 3.9a) and a yellow block (Fig. 3.9e) are an intersection point, a blue line (Fig. 3.9a) are d_l and d_r, a light purple block (Fig. 3.9c) is a block that d_l or d_r are lain, and a red block (Fig. 3.9d and Fig. 3.9e) is starting inner-edge's block.

Algorithm 2 Starting inner-edge selection

Input: *con*: the outer contour and p: an intersection point

1: d_l, d_r = a direction of *con* on the left and the right side of p, respectively (see Fig. 3.9a)
2: Flip d_l and d_r by 180° (see Fig. 3.9b)
3: b_{d_l}, b_{d_r} = neighbor blocks of the block, in which p belongs to, are lain on the direction of d_l and d_r, respectively (see Fig. 3.9c)
4: **if** $|\vec{D}|$ of $b_{d_l} > |\vec{D}|$ of b_{d_r} **then**
5: $currB = b_{d_l}$
6: **else**
7: $currB = b_{d_r}$
8: **end if**

return *currB* (i.e., the red block in Fig. 3.9d and Fig. 3.9e)

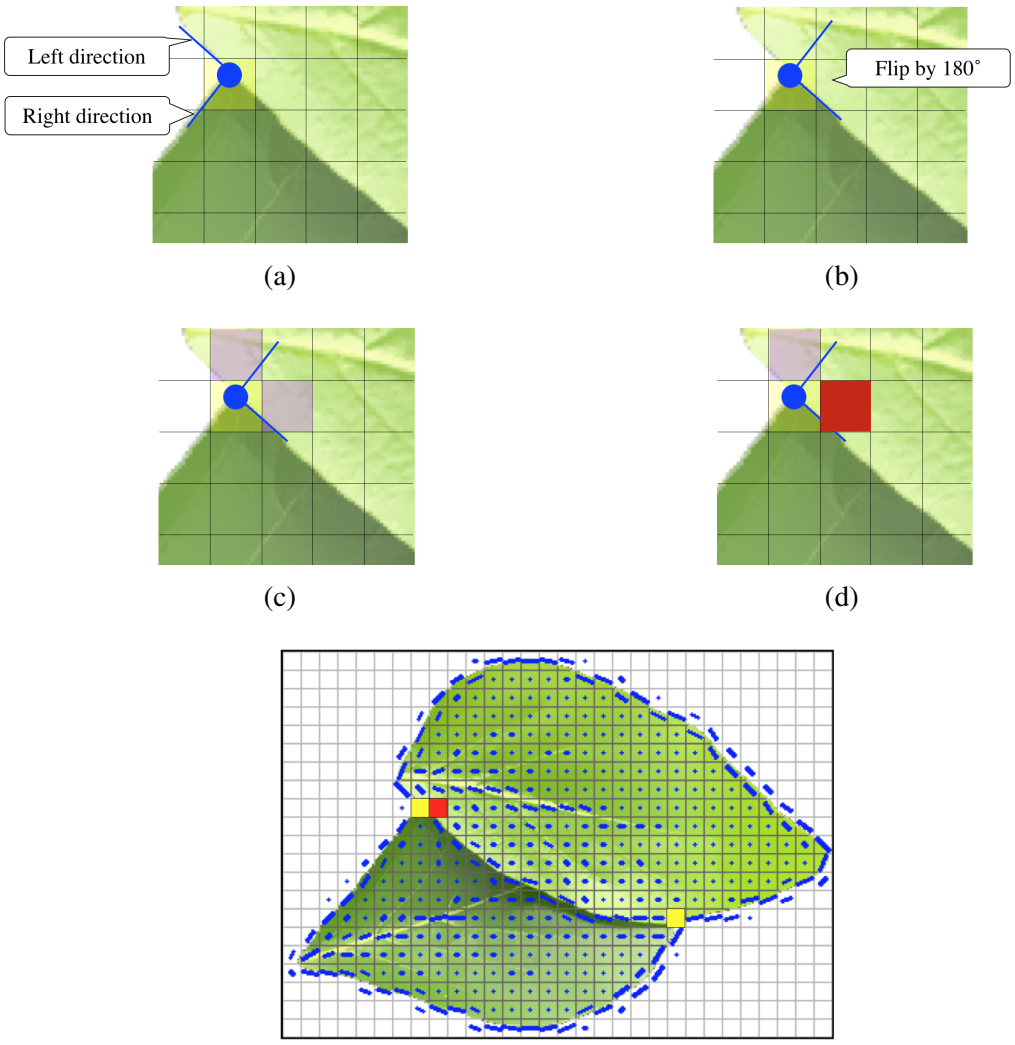


Figure 3.9: Starting inner-edge selection

After starting inner-edge selection was done, inner-edge tracking begins. It finds the path of a separately occluded leaf's edge using information of the direction field as a guideline.

To be specific, Algorithm 3 is formulated to identify leaf inner-edge path from each intersection point. A set of points, $currEdge$, is created to store the inner-edge. An intersection point p is added to $currEdge$ for connecting the outer contour con with the inner-edge $currEdge$, then 1) the center position of the block that p belongs to and 2) the center position of the starting block (i.e., result of Algorithm 2) are added, respectively.

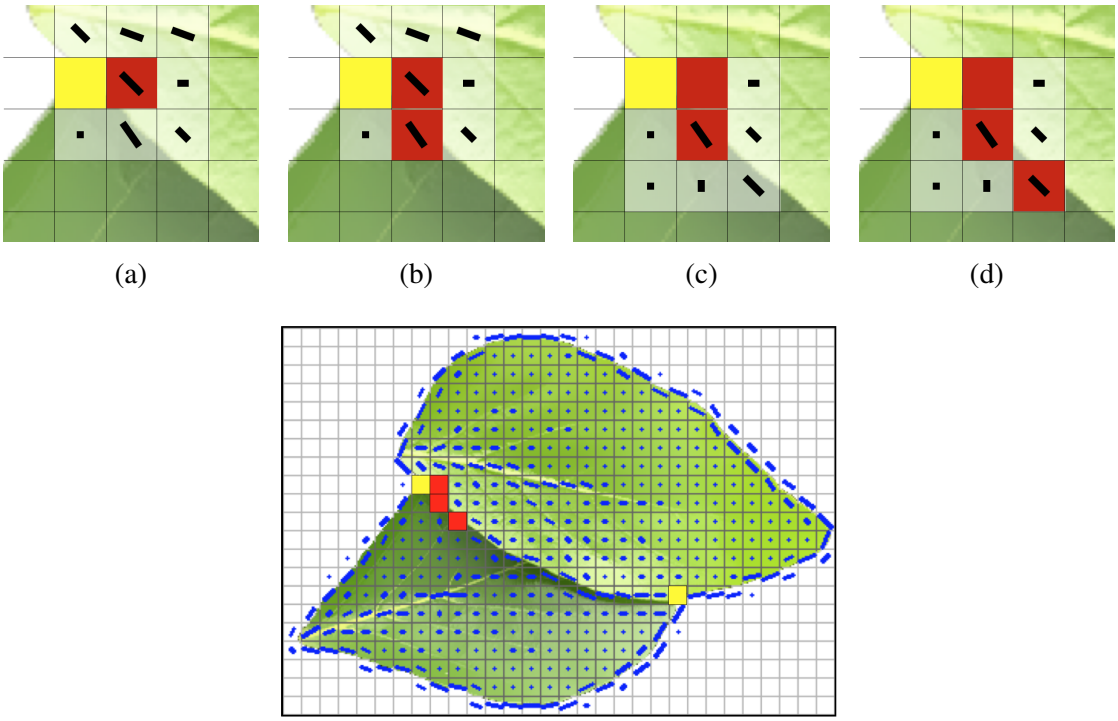
Next, the next block is selected from adjacent neighbor blocks. The block is indicated by choosing the block where maximum direction value belongs to. The direction value from a neighbor $neighW$ is denoted by sum of a constant angle's weight α_w times angle value w_a and a constant magnitude's weight $(1 - \alpha_w)$ times magnitude value w_m (line #17). Noted that sum of the angle's weight and the magnitude's weight is equal to 1, and these weights are predefined. They are used to tell how important they are. If the angle is more important than magnitude, its weight will be higher than another one, vice versa.

w_a is computed by finding the absolute difference between the angle's current block $currB$ and the angle's neighbor block $neighB$, divided by 180° (line #15). Then it is used to subtract from 1 because of the more similar angle, the more value. w_m is computed by normalization of the magnitude of neighbor's block (line #16). The stronger magnitude, the more value. Noted that w_a and w_m are in the range from 0 to 1.

$neighW$ is enhanced by 1.1 if the angles of $currB$ and $neighB$ are the same angle after quantizing by 4 (line #19). The center position of the block, which contains $maxW$, is added to $currEdge$.

Figure 3.10 represents a block selection in detail. The thicken black line in a block means the block's direction. The longer line, the stronger magnitude. The different slopes, the different angles. Additionally, a yellow block means a block in which inter-

section point belongs to, while a red block means an estimated inner-edge. Moreover, the edge direction is chosen based on the values of angle and magnitude. In Fig. 3.10a, the center-bottom block gives the maximum direction value compare to other neighbor's blocks, so it is chosen as the next inner-edge direction and added to inner-edge (i.e., *currEdge*), as shown in Fig. 3.10b. Then the right-bottom is chosen and added again (see Fig. 3.10d), so the current result is shown in Fig. 3.10e.



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

(e)

Figure 3.10: Example of inner-edge extraction

After that, finding the next block is applied again until the *currB* is already in *edge* or any pixel in *currB* is a background pixel. Finally, the current inner-edge *currEdge* is added to a set of edges *edges* (line #33). For example, Fig. 3.11 shows the result of inner-edge extraction, which is explained in Algorithm 3. To be concluded, leaf inner-edge estimation is calculated by:

1. For each intersection point

Algorithm 3 Inner-edge extraction

Input: α_w: angle's weight (range from 0 to 1)

con = the outer contour
p = an intersection point
prevB = the block in *P* that *p* belongs to
$currEdge = \{p\}$
Add the center position of *prevB* to $currEdge$
currB = the result of Algorithm 2
Add the center position of *currB* to $currEdge$
while true **do**
    $maxW = -\infty$
    for each 3×3 neighbor blocks of *currB* **do**
        neighB = the current neighbor block
        if (It is the first iteration of the while loop, and *neighB* contains a background pixel) or (*neighB* is the same block as *prevB* or adjacent to *prevB*) **then**
            Continue to the next neighbor
        end if
        $w_a = 1 - ((|\angle\vec{D} \text{ of } currB - \angle\vec{D} \text{ of } neighB|) \div 180^\circ)$
        $w_m = |\vec{D}|$ of *neighB*
        $neighW = \alpha_w w_a + (1 - \alpha_w) w_m$
        if $\angle\vec{D}$ of *currB* with quantized by 4 is equals $\angle\vec{D}$ of *neighB* with quantized by 4 **then**
            $neighW = neighW \times 1.1$
        end if
        if $maxW < neighW$ **then**
            $maxW = neighW$
            $maxB = neighB$
        end if
    end for

    $prevB = currB$
    $currB = maxB$
    Add the center position of *currB* to $currEdge$
    if (*currB* is in *edges*) or (*currB* contains a background pixel) **then**
        Break out of the while true loop
    end if
end while

Add $currEdge$ to *edges*

(a) Using Algorithm 2 to find inner-edge starting position

(b) Using Algorithm 3 to find other inner-edge positions

2. Send the result *edges* to the next step, which is inner-edge refinement

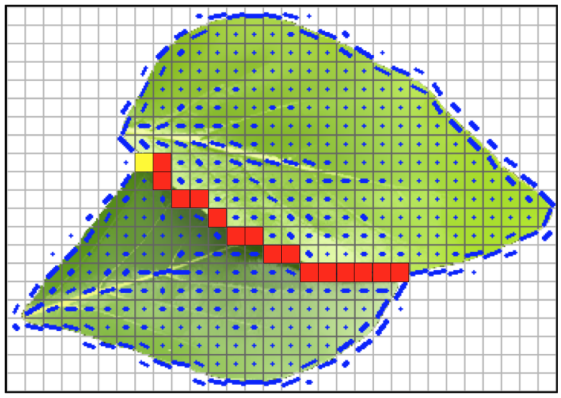


Figure 3.11: Result of inner-edge extraction

3.3.2 Inner-edge refinement

The inner-edge refinement is applied for cleaning every inner-edge by removing unwanted points in the edge. It consists of three steps: solving cyclic edge, straightening tortuous edge, and removal of irrelevant edge. Regarded that every step has to be applied in *edges* sequentially.

Firstly, the solving cyclic edge (see Fig. 3.12) is included to loose all circles in a edge. It starts by finding an intersection point in the edge. If two lines are crossing each other, then every points between them will be erased, and unlinked points will be connected.

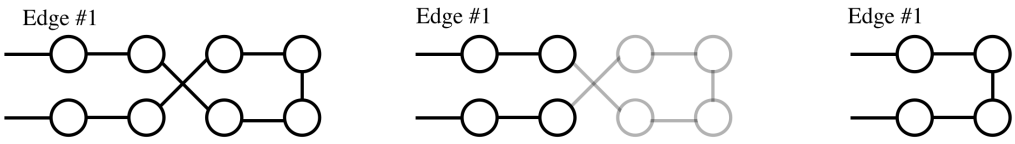


Figure 3.12: Example of solving cyclic edge

Secondly, the straightening tortuous edge (see Fig. 3.13) is used to find the shortest path in the edge by eliminating every tortuous line. This step can be used to reduce the processing time of the next step, and improve the result accuracy of this proposed method.

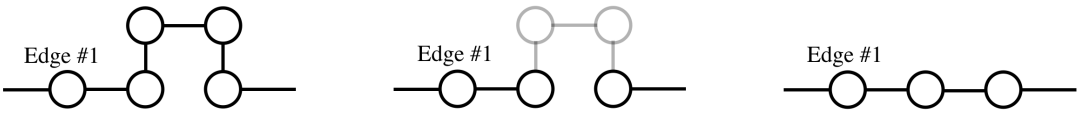


Figure 3.13: Example of straightening tortuous edge

Finally, the removal of irrelevant edge (see Fig. 3.14) is used to combine edges and eliminate unwanted points from the inner-edge. It connects the edges that can be link from one intersection point (yellow circle) to another point, then any irrelevant path in the edge is removed.

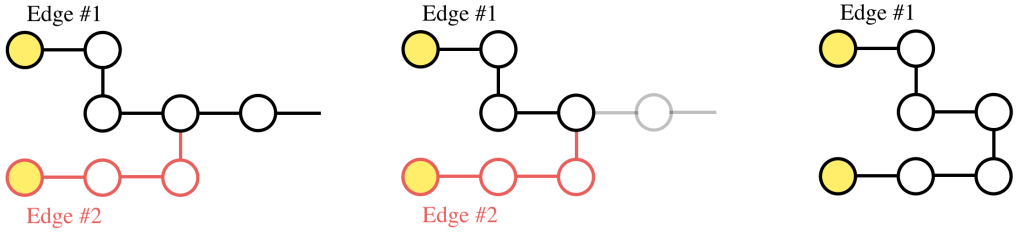


Figure 3.14: Example of removal of irrelevant edge

After the refinement, the remaining edges *edges* and the outer contour *con* are drawn in a binary image E, as shown in Fig. 3.15.

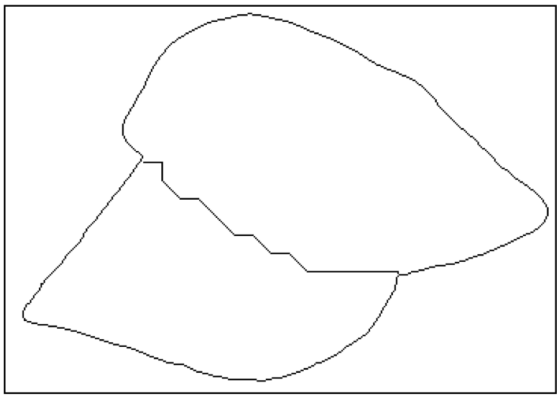


Figure 3.15: All leaf edges (E)

3.3.3 Rough leaf region identification

This step is performed to create a set of foreground markers for a watershed transform.

$$M_f(x,y) = \begin{cases} P(x,y) & \text{if } P(x,y) \text{ is a background pixel,} \\ E^{-1}(x,y) & \text{otherwise.} \end{cases} \tag{3.9}$$

where M_f is a foreground marker set and E^{-1} is the invert color of E. The result of the invert leaf edges image is presented in Fig. 3.16.

Erosion morphological operator Section 2.1.3 is applied to M_f in order to erode the leaf regions for receiving the foreground marker set (see rough leaf region image M_f in Fig. 3.16). All eroded areas denote a non-marked-pixel, while other areas are marked as a leaf region, as shown in the first and the second regions in M_f. Then the marker set will be applied with watershed transform to judge which leaf a non-marked-pixel belongs to. Before performing watershed, too small regions are considered as noise and removed from the image.

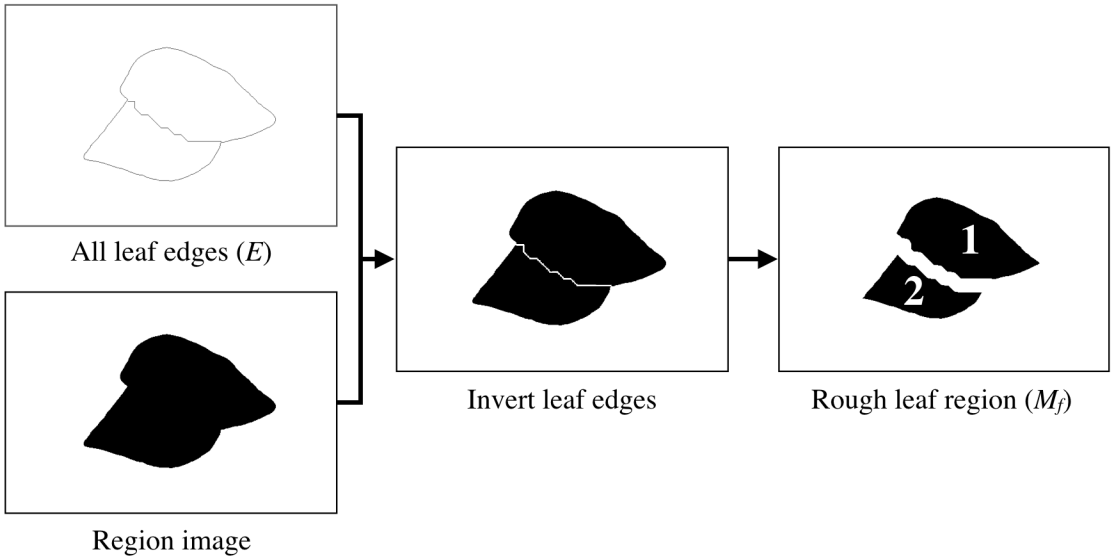


Figure 3.16: Rough leaf region identification steps

3.4 Leaf separation

Although the result of the previous process is enough for leaf separation (achieve the objective of this research), the result accuracy can be further improved when watershed transform is applied.

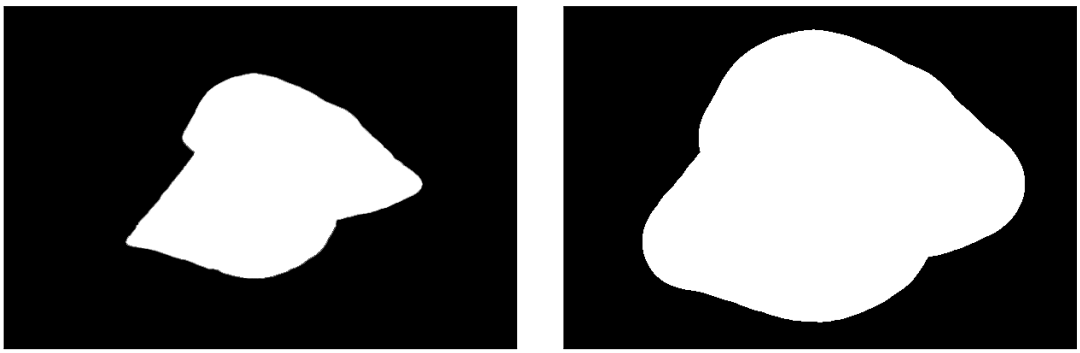
Watershed transform (see more detail in Section 2.1.4) is used to make a decision which a non-marked-pixel belongs to a region. It also requires a set of predefined markers to recognize the total number of regions and their location. There are two kinds of a marker which are a foreground marker and a background marker. Additionally, the number of result images is equal to the total number of regions. Moreover, the more accurate marker, the more accurate result. On the other hand, the less accurate marker may produce the less accurate result.

At this moment, M_f is used as foreground markers for watershed transform. How-

ever, a background marker is not generated, so it will be created in this step as follows:

$$M_b(x,y) = D(B(I(x,y))), \tag{3.10}$$

where M_b denotes a background marker, D denotes a dilation morphological operator, and B denotes a binary operator that converts foreground pixels of I to one color and converts background pixels of I to another color (see Fig. 3.17).



(a) Binary image $B(I(x,y))$ (b) Background marker (M_b)

Figure 3.17: Background marker computation

Fig. 3.18 represents the result after applying watershed transform with the markers M_f and M_b to the input image I. The result is a set of single-leaf images with a plain background, which can be used as an input of many applications, such as leaf identification, nutrient deficiency detection, and leaf symptom detection.

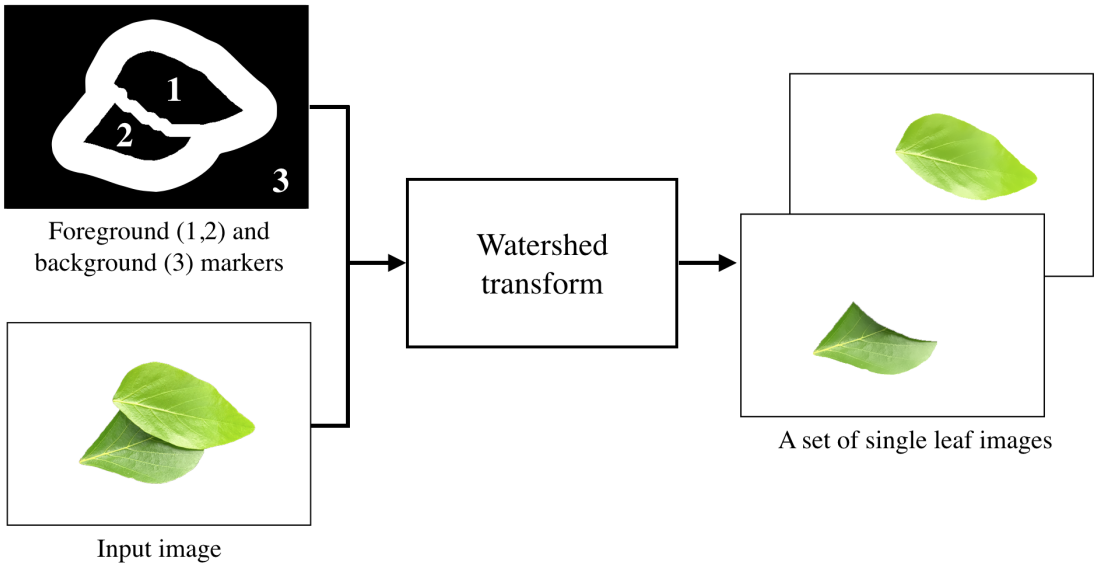


Figure 3.18: Result of this proposed method

Chapter 4

Experiments and discussion

Experiments, its result and discussion are determined in this chapter. This chapter consists of four experiments which are 1) finding the parameters for the proposed method, 2) evaluation of intersection point estimation process, 3) evaluation of the proposed method, and 4) verification of the proposed method.

4.1 Finding the parameters for the proposed method

4.1.1 Objective

This experiment was done to select the best parameters for the proposed method. It was very important because when the parameters' value was changed, the performance of this research became different. In addition, while one of the parameters' value was not fit to a problem, it is hard for the rest to produce a good result. Noted that changeable parameter meant a parameter which can be adjusted to make it proper for a specific problem.

4.1.2 Experiment setup

The image dataset was applied in this experiment consisted of 128 occluded leaf images with different species in the same image. They were collected around the campus (KMITL, Thailand) and captured in different angles in a controlled environment (i.e., in a photo light tent) with iPhone 7. There were 35, 35, 35, and 23 images which contain 2, 3, 4, and more than four overlapped leaves with a plain background, respectively. Each image was cropped and then resized to 600×400 pixels, as shown in Fig. 4.1.

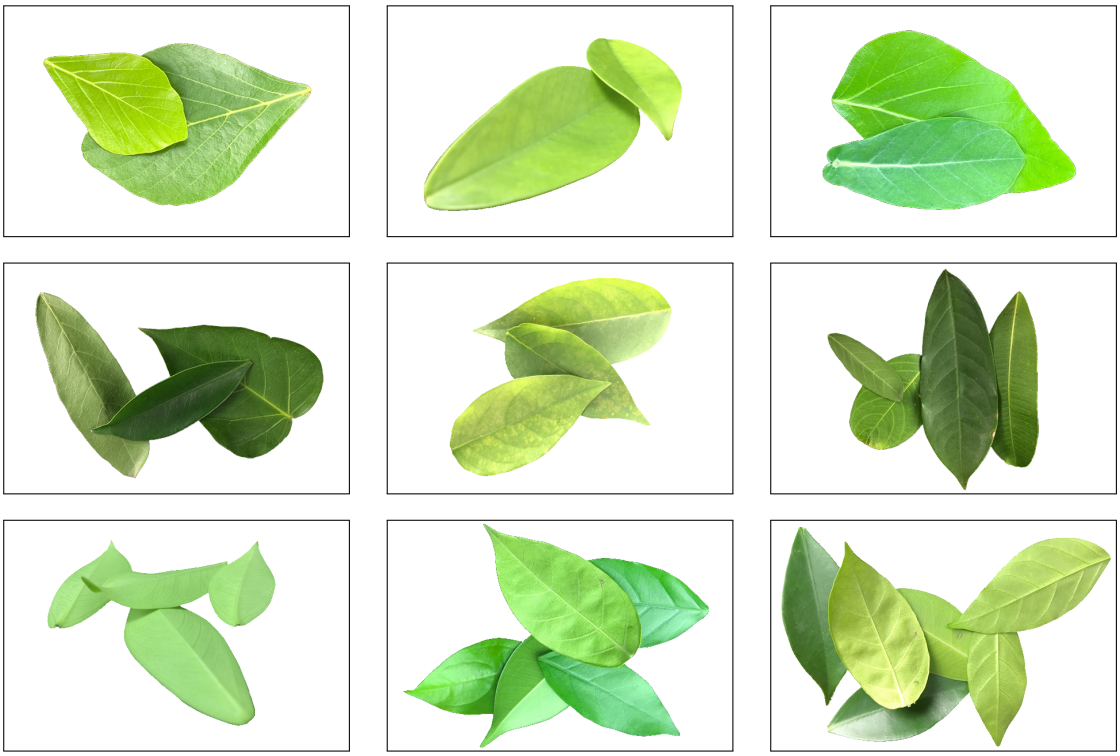


Figure 4.1: Examples of occluded leaf image with controlled environment

After acquired input images, this experiment was conducted. Any image with more than four occluded leaves was excluded. Then, the proposed method was executed many times with different parameters' value used in the proposed method. There were three changeable parameters affected the method which were sampling number, curvature threshold value, and angle's weight. The first two ones were used in the intersection

point estimation process, and the last one was used in the leaf inner-edge estimation process.

The first parameter was a value used to sample the outer contour. To be specific, when sampling number equaled to n, the index of every point modulo n be equal to zero was remained (the others would be eliminated), and the size of the contour was divided by n. Moreover, the sampling number affected the computation time in this research. The higher value, the less computation time. However, too high value might produce the inaccurate result because some information of the contour was lost.

The second one was the curvature threshold value, which was used to select a candidate intersection point. Only the point in which the curvature value higher than the threshold was chosen. This value was also very important because a performance of the proposed method is sensitive to the number of intersection points.

Lastly, angle's weight was a changeable parameter that was used to consider the importance of angle and magnitude. Reminded that sum of the angle's and magnitude's weights equaled to one, and the weights' range was from zero to one. The higher value of the angle's weight meant the more important angle and the less important magnitude. In addition, when the angle was important than the magnitude, leaf inner-edge extraction process would consider a neighbor's block with a similar angle to the current block more than the block with stronger magnitude.

The evaluated result consisted of 1) the percentage of input images with correct intersection points, 2) the average of error distances, and 3) the percentage of input images with correct result images. The first one was counted when the number of predicted intersection points equaled to the number of actual intersection points in an input image. The second one, error distance, meant the average of the euclidean distance between the predicted intersection point and the actual intersection point. It represented how far from actual point to predicted point. The less error, the greater result. Another one, the percentage of input images with correct intersection points, was increased when the

number of predicted output images equaled to the actual images. Reminded that the result of the proposed method was a set of single-leaf images.

4.1.3 Result and discussion

Table 4.1 represents the result of different parameter values. The sampling number, the curvature threshold value, and the weight of angle began at 1, 0.00, and 0.0, respectively. The step of sampling number and curvature threshold were increased by 1 and 0.01 until the percentage of a correct number of result image was dropped. Furthermore, the angle's weight was in the range from 0.0 to 1.0 with step size 0.1. When the sampling number equaled to 1, the contour was not sampled. When the curvature threshold equaled to 0.00, every point, which was concave, was selected. When the angle's weight was 0.0, the leaf inner-edge estimation process concerned only the angle in a block, not the magnitude.

Table 4.1: Different parameter values and their result

Parameter values			Their result		
Sampling number	Curvature threshold	Angle's weight	Correct intersection points (%)	Error distance	Correct result images (%)
1	0.00	0.0	0.00	76.52	29.52
		0.1			32.38
		0.2			29.52
		0.3			28.57
		0.4			29.52
		0.5			29.52
		0.6			35.24
		0.7			26.67

continued ...

Different parameter values and their result (continue)

Parameter values			Their result		
Sampling number	Curvature threshold	Angle's weight	Correct intersection points (%)	Error distance	Correct result images (%)
		0.8			25.71
		0.9			26.67
		1.0			26.67
	0.01	0.0	44.76	20.83	62.86
		0.1			59.05
		0.2			60.00
		0.3			61.90
		0.4			60.95
		0.5			60.95
		0.6			54.29
		0.7			56.19
		0.8			57.14
		0.9			57.14
		1.0			52.38
	0.02	0.0	65.71	6.72	63.81
		0.1			63.81
		0.2			64.76
		0.3			64.76
		0.4			62.86
		0.5			62.86
		0.6			60.00
		0.7			64.76

continued . . .

Different parameter values and their result (continue)

Parameter values			Their result		
Sampling number	Curvature threshold	Angle's weight	Correct intersection points (%)	Error distance	Correct result images (%)
		0.8			65.71
		0.9			61.90
		1.0			62.86
	0.03	0.0	54.29	4.53	70.48
		0.1			72.38
		0.2			75.24
		0.3			76.19
		0.4			74.29
		0.5			73.33
		0.6			69.52
		0.7			74.29
		0.8			74.29
		0.9			72.38
		1.0			67.62
	0.04	0.0	34.29	2.95	68.57
		0.1			70.48
		0.2			71.43
		0.3			72.38
		0.4			70.48
		0.5			68.57
		0.6			65.71
		0.7			67.62

continued . . .

Different parameter values and their result (continue)

Parameter values			Their result		
Sampling number	Curvature threshold	Angle's weight	Correct intersection points (%)	Error distance	Correct result images (%)
		0.8			66.67
		0.9			63.81
		1.0			63.81
2	0.00	0.0	21.90	47.02	52.38
		0.1			54.29
		0.2			51.43
		0.3			47.62
		0.4			52.38
		0.5			55.24
		0.6			50.48
		0.7			51.43
		0.8			54.29
		0.9			50.48
		1.0			46.67
	0.01	0.0	44.76	4.21	71.43
		0.1			73.33
		0.2			**76.19**
		0.3			75.24
		0.4			72.38
		0.5			73.33
		0.6			68.57
		0.7			69.52

continued . . .

Different parameter values and their result (continue)

Parameter values			Their result		
Sampling number	Curvature threshold	Angle's weight	Correct intersection points (%)	Error distance	Correct result images (%)
		0.8			67.62
		0.9			64.76
		1.0			67.62
	0.02	0.0	25.71	2.57	65.71
		0.1			68.57
		0.2			66.67
		0.3			66.67
		0.4			63.81
		0.5			62.86
		0.6			59.05
		0.7			58.10
		0.8			65.71
		0.9			60.00
		1.0			60.00
3	0.00	0.0	52.38	22.12	60.95
		0.1			61.90
		0.2			60.95
		0.3			59.05
		0.4			56.19
		0.5			55.24
		0.6			48.57
		0.7			51.43

continued . . .

Different parameter values and their result (continue)

Parameter values			Their result		
Sampling number	Curvature threshold	Angle's weight	Correct intersection points (%)	Error distance	Correct result images (%)
		0.8			55.24
		0.9			56.19
		1.0			56.19
	0.01	0.0	19.05	3.17	58.10
		0.1			60.95
		0.2			62.86
		0.3			60.00
		0.4			57.14
		0.5			55.24
		0.6			53.33
		0.7			51.43
		0.8			51.43
		0.9			50.48
		1.0			59.05

Table 4.1 could be performed into a graph, which was shown in Fig. 4.2. The x-axis denoted the weight of angle in the range of 0.0 to 1.0 with step size 0.1, while the y-axis denoted the accuracy of the result images in the range of 0% to 80% with step size 10. Recall that the system resulted in this research received an occluded leaf image and then produced a set of single-leaf images as output. Thus, the result accuracy would be increased when the number of result images fits with their ground truth. A slimmer line was the result accuracy fixed the two parameters, sampling number and curvature

threshold, with different angle's weights. Moreover, the thicker line was the average accuracy at different angle's weights.

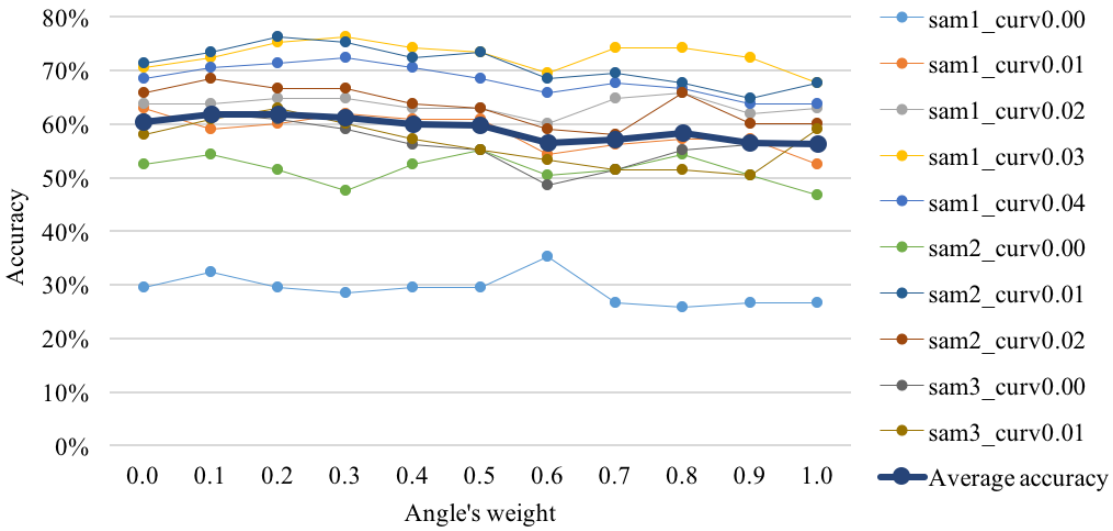


Figure 4.2: Angle's weight information

In this figure, an average accuracy with the angle's weight less than 0.5 produced the greater result than more than 0.5. That meant the leaf inner-edge estimation process should consider the angle more than the magnitude. Furthermore, the highest one was 0.2 for the angle's weight. Therefore, 0.2 should be selected as the weight of angle parameter value.

After the parameter of the angle's weight was fixed, the relationship between the other two parameters was shown in Table 4.3. Each row represented the sampling number, while each column represented the curvature threshold value with 0.2 angle's weight. There were three different results in each cell. The first result was the average of the intersection point accuracy. The accuracy was counted when the number of intersection points in an input image fixed with their ground truth. The second one was the average of error distances. The distance was calculated from the euclidean distance between an intersection point and its ground truth. Lastly, the average of result accuracies

of the proposed method.

Table 4.2: Different results when the angle's weight equaled to 0.2. Each cell represented the accuracy of intersection point (%), error distance, and the accuracy of the proposed method (%), respectively.

Sampling number \ Curvature threshold	0.00	0.01	0.02	0.03	0.04
1	0.00% 76.52 29.52%	44.76% 23.83 60.00%	65.71% 6.72 64.76%	54.29% 4.53 75.24%	34.29% 2.95 71.43%
2	21.90% 47.02 51.43%	**44.76%** **4.21** **76.19%**	25.71% 2.57 66.67%		
3	52.38% 22.21 60.95%	19.05% 3.17 62.86%			

The parameter selected was 2 for the sampling number, 0.01 for the curvature threshold, and 0.2 for the weight of angle (the bold font). Reminded that the first two parameters were used in the intersection point estimation process, and another was used in the leaf inner-edge estimation process. Since, when these parameters were applied, the proposed method produced a result with the highest accuracy (76.19%) compared with the others. Moreover, error distance was only 4.21, which meant the detected intersection points were very close to their ground truth.

Even though, with these parameters, the accuracy of correct result images was highest, and error distance was acceptable, the correct intersection points was quite mod-

erate. Since the point with curvature larger than threshold was used as a candidate intersection point. Reminded that the curvature determined how rapidly curve changed, also called concave. Sometimes, all concave points might not occur when leaves were overlapped. For example, two occluded leaves commonly had two concave points, but the image in Fig. 4.3 had only one point (the blue circle). For the right point, the boundary between leaves was too smooth to have the concave (no bend toward). This was one of the reasons why the accuracy of correct intersection point was not high.

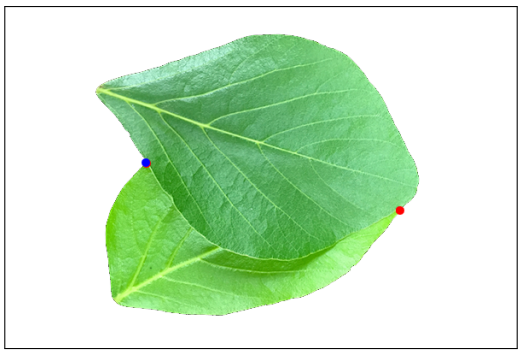


Figure 4.3: Image with a single concave point

Fortunately, lesser total intersection points might provide the correct result because a mere single point could be used to extract the correct separated-leaf-edge. For example, 65.71% for correct intersection point (higher than selected) with the non-sampling number and 0.02 threshold value, the accuracy of correct result images (64.76%) was lower than the selected one (76.19%).

The ordered priority used to select parameters was error distance to correct result image to correct intersection point. However, choosing the smallest error, 2.57, with only 25.71% of the accuracy of correct intersection points was not a good idea because the accuracy of the points was too low. The 2.95 and 3.17 error distances are also the same reason.

As the result, 2 for the sampling number, 0.01 for the curvature threshold value, and 0.2 for the weight of angle were the best choice for selected parameters.

4.2 Evaluation of intersection point estimation

4.2.1 Objective

This experiment was done to measure the correctness of the intersection point detection process. The correctness represented a comparison between the total number of predicted points and actual points (their ground truth), and error distance between them.

4.2.2 Experiment setup

This experiment was conducted with the same dataset as the previous experiment, Section 4.1. It also applied the selected parameters obtained from the previous experiment, 2 for sampling number and 0.01 for intersection point threshold value.

After executed intersection point estimation process, the correctness of this process was presented and described in detail in Section 4.2.3.

4.2.3 Result and discussion

Table 4.3 presented the number of estimated intersection point with selected parameters. The first column was the estimation error (the left column) equaled to the total number of predicted intersection points minus the total number of actual points in an input image. Non-estimation-error meant the total number of intersection points fit with the total number of ground truth points. A positive error meant the total number of intersection points was greater than their ground truth, and vice versa. The second column was the total number of input images when the number of detected intersection points fit with their ground truth, and the last one was the second column in percent.

Table 4.3: The accuracy of detected intersection points

Estimation error	Number of images	Percentages (%)
$+1$	3	2.86
0	**47**	**44.76**
-1	46	43.81
-2	8	7.62
-3	1	0.95
Total	105	100.00

More than half images in dataset had negative estimation error and a few images had positive estimation because every intersection point had to be a starting position used to separate leaves. If the number of detected points was more than its ground truth, the number of resulting images might be more than expected, which decreased the accuracy of the proposed method. On the other hand, less number of intersection points might extract divided edge successfully and accurately. Therefore, the negative estimation error was better than the positive one for the number of intersection points.

From every intersection point in images, the error distance between the point and its corresponding ground truth was explained in Table 4.4. The minimum error was zero, which meant the position of the point and its ground truth was exactly the same position (see the left circle in Fig. 4.4). The distance between the points was very close together as the average value, as shown the right circles in the same image. Noted that a blue circle and a red circle denote an actual and predicted intersection points, respectively. 136 error distance for the maximum value was considered as an outlier because the standard deviation was only 13.29. This average error distance was acceptable with merely 4.21.

Table 4.4: Error distance between intersection points and their ground truth

Min	Max	Avg	SD
0.00	136.00	4.21	13.29

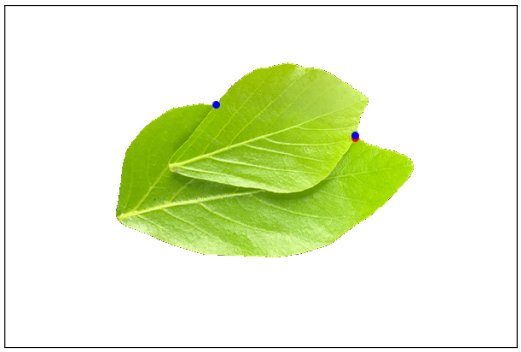


Figure 4.4: Image with intersection points (blue) and ground truth points (red)

4.3 Evaluation of the proposed method

4.3.1 Objective

The experiment was conducted to measure the accuracy of quantity and quality of this proposed method by comparing between result images and their corresponding ground truth. For the quantity measurement, the experiment would be focused on the total number of result images and the total number of their ground truth. For the quality measurement, it would be focused on segmentation accuracy in each result image.

4.3.2 Experiment setup

This experiment was applied the same dataset as the previous ones. The dataset used in this experiment was divided into two types: dataset with and without more than four occluded leaves. Reminded that the proposed method received an occluded leaf image

as an input then produced a set of single-leaf images as an output. In order to measure by quantity, form each input image, the total number of result images was compared with its corresponding ground truth.

For quality measurement, only the images with prediction error equaled to 0 were applied. There were four values used in the measurement which were the average accuracy, true positive rate (TPR) or recall, true negative rate (TNR), and precision of the proposed method. They were computed as follows.

$$\begin{aligned} Accuracy &= \frac{tp+tn}{tp+tn+fp+fn}, \\ TPR &= \frac{tp}{tp+fn}, \\ TNR &= \frac{tn}{tn+fp}, \\ Precision &= \frac{tp}{tp+fp}, \end{aligned} \tag{4.1}$$

where tp denoted the total number of target-leaf pixels that were correctly identified as target-leaf, tn denoted the total number of non-target-leaf pixels that were correctly identified as non-target-leaf, fp denoted the total number of non-target-leaf pixels that were incorrectly identified as target-leaf and, fn denoted the total number of target-leaf pixels that were incorrectly identified as non-target-leaf, as shown in Fig. 4.5.

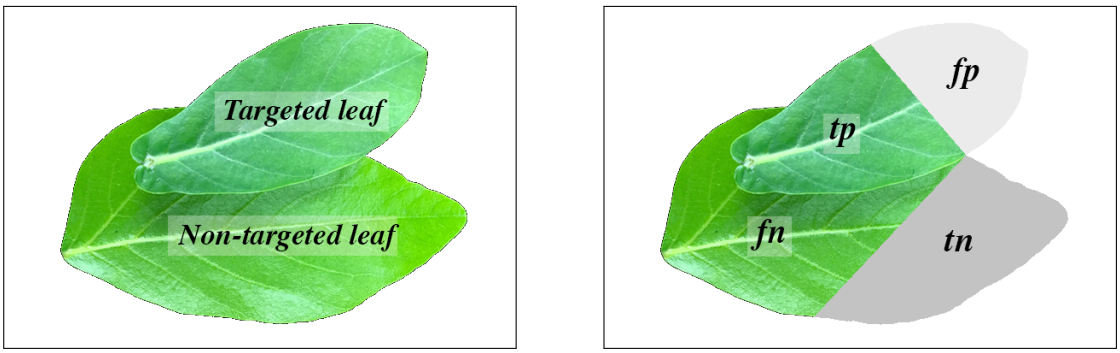


Figure 4.5: Quality measurement explanation

4.3.3 Result and discussion

Table 4.5 and Table 4.6 presented the result measured by quantity. The separation error was similar to the estimation error in Table 4.3, but measured the total number of result images instead. To be specific, the error denoted the total number of predicted result images minus the total number of actual result images (ground truth). For example, in Fig. 4.6, the input image contains five leaves, and its result is a set of five images with a single leaf, so the separation error will be zero (also called a non-separation-error).

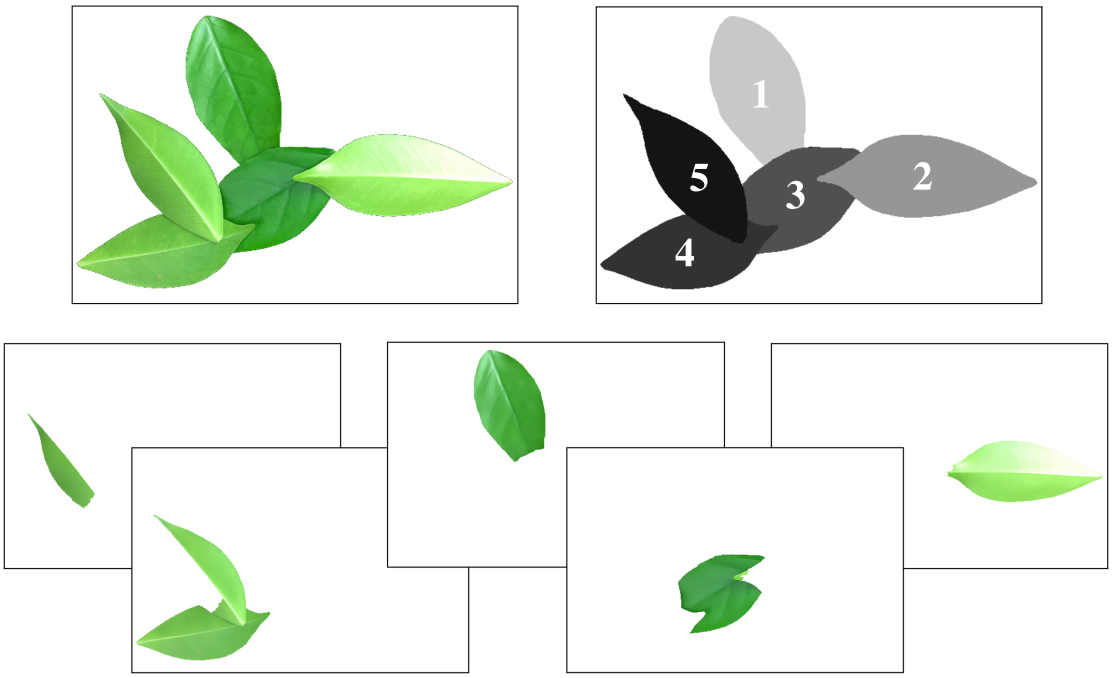


Figure 4.6: Example of input image, its ground truth, and its result

The proposed method correctly produced two, three, four, and more than four occluded leaves with 85.71% (30 out of 35), 82.86% (29 out of 35), 60.00% (21 out of 35), and 47.83% (11 out of 23), respectively. The accuracy of non-separation-error was declined sequentially when the number of occluded leaves was risen. In addition, the negative error was greater than the positive one. That meant the total result images was less than expected, since the total number of points in the intersection point estimation

process was less than the ground truth. To solve this problem, adding the method to estimate an intersection point without using information of curvature should give the better result.

Table 4.5: The accuracy of result images in detail

Separation error	Number of occluded leaves in input images (in total and in percent)							
	2		3		4		> 4	
$+1$	3	8.57	5	14.29	4	11.43	7	30.43
0	**30**	**85.71**	**29**	**82.86**	**21**	**60.00**	**11**	**47.83**
-1	2	5.71	1	2.86	9	25.71	5	21.74
-2	0	0.00	0	0.00	1	2.86	0	0.00
#image	35		35		35		23	

The summarized version of Table 4.5 was shown in Table 4.6, which consisted of two versions: the total result images and without more than four occluded leaf input images.

Table 4.6: The accuracy of result images in summary

Separation error	Number of result images			
	Excluding > 4		Total	
$+1$	12	11.43	12	9.38
0	**80**	**76.19**	**87**	**67.97**
-1	12	11.43	23	17.97
-2	1	0.95	6	4.69
#image	105		128	

For the first version, the accuracy of the proposed method with non-separation-error

was 80 input images from 105, which was 76.19%. Most of the absolute errors were only 1, which was acceptable error (i.e., 22.86% came from 11.43% for plus-one error and 11.43% for minus-one error). Only one input image produced the negative error more than one.

When more than four occluded leaf images were included, the correct result was decreased by around 8% (dropped from 76.19% to 67.97%). That meant the proposed method could handle a bit of image with more than four occluded leaves, and the one-absolute separation-error was 27.34% out of 32.03% (all errors), which was also acceptable error.

The quality measurement was conducted and described in Table 4.7. Each number represented the correctness of the total pixels in single-leaf images. The images included only non-separation-error. That meant 87 input images out of 128 images were used to measure for aspect quality. To be specific, a set of 268 result images were applied in this measurement.

Table 4.7: Overall accuracy of the number of result images with non-separation-error

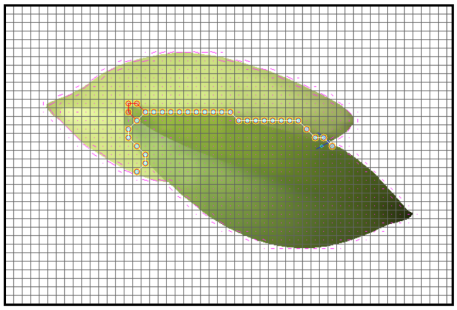
	Number of occluded leaves				Average	Average
	2	3	4	> 4	without > 4	of the total
Accuracy (%)	84.20	95.49	91.10	90.30	90.96	90.87
TPR or recall (%)	79.86	91.16	80.92	72.79	84.50	82.81
TNR (%)	82.44	97.00	94.71	94.69	92.39	92.70
Precision (%)	82.01	95.30	88.12	78.85	89.23	87.80

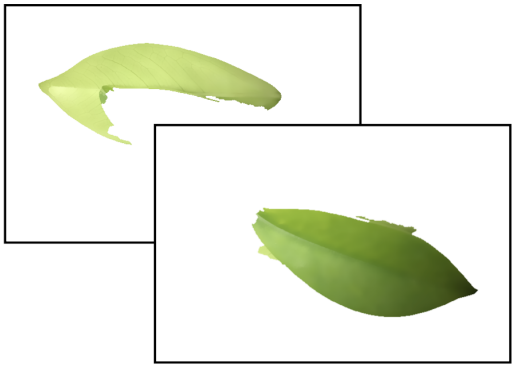
Although the total number of result images excluding and including more than four were different, but the accuracy, recall, true negative rate (tnr), and precision of the total result images were very similar. Thus, measured by quality with non-separation-error, the results were not different. Moreover, with four values, accuracy, recall, tnr, and

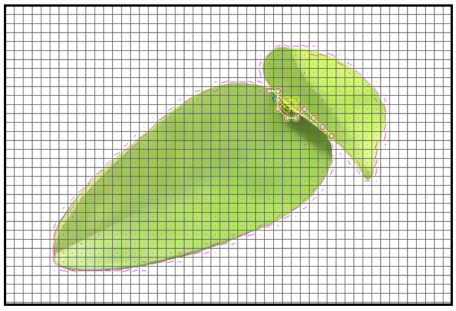
precision, no one was less than 80%, and the accuracy was up to 90%, which was a preferable result.

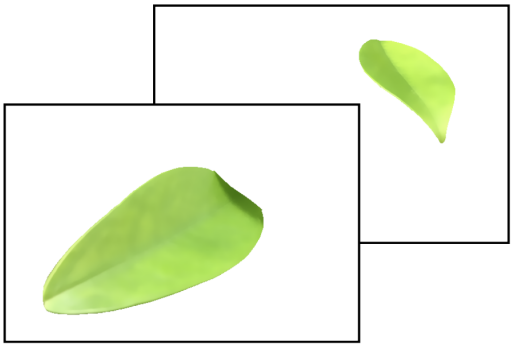
Computation time of the proposed method was around 1.193 seconds per image, tested on a 2.5 GHz Intel Core i7 machine, and implemented using C++ language. For 128 images, it spent around 2 minutes 30 seconds to produce a set of 415 images as outputs.

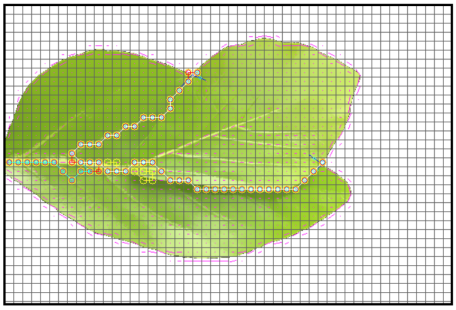
Figure 4.7 represented a leaf inner-edge direction (leaf column) and a result of the proposed method (right column), which was a set of single-leaf images. A red and blue lines in the left column was an edge removed in the edge refinement process. The first and the second examples were captured in different angles, but the result images could be produced accurately. This meant that the proposed method supported different angle of an occluded leaf image. In the third, the fourth, and the fifth examples, some inner-edge directions moved toward to the mid-vein because of its strong edge. However, the third step in edge refinement, removal of irrelevant edge, could successfully eliminate this edge and produced the more accurate result compare with no edge refinement.

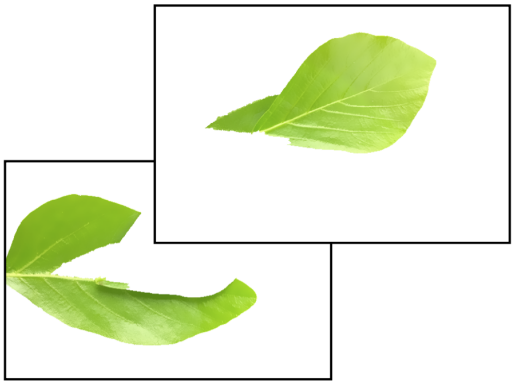


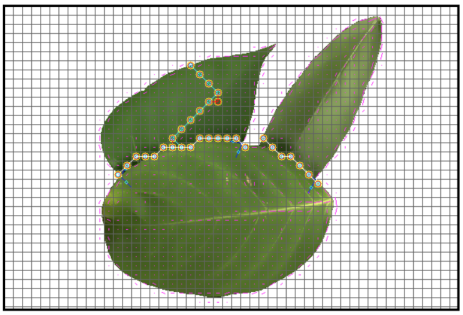


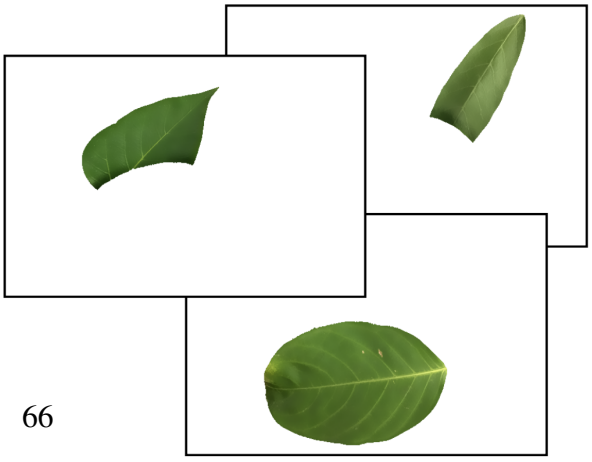


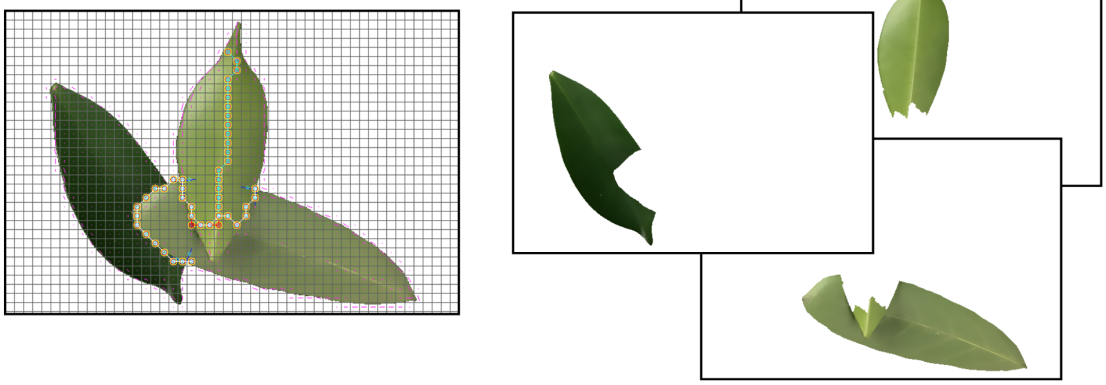


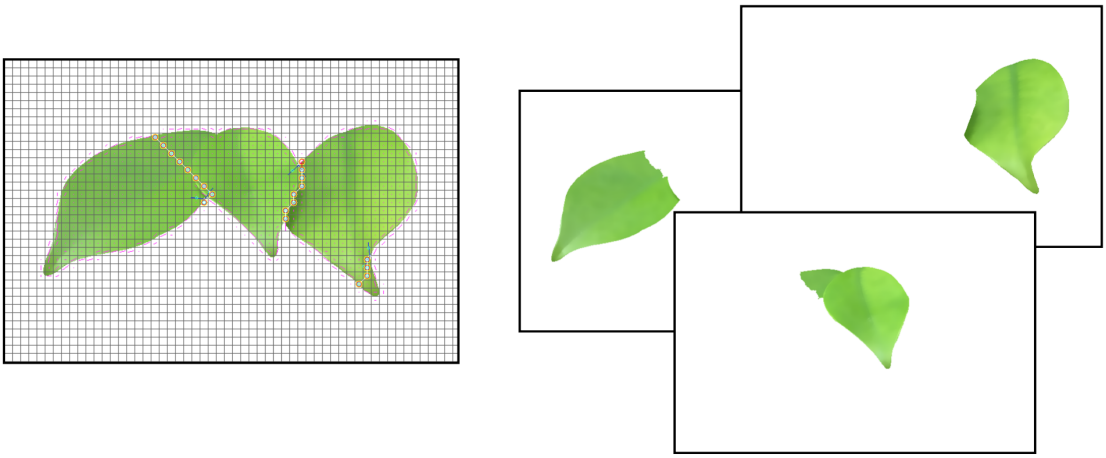


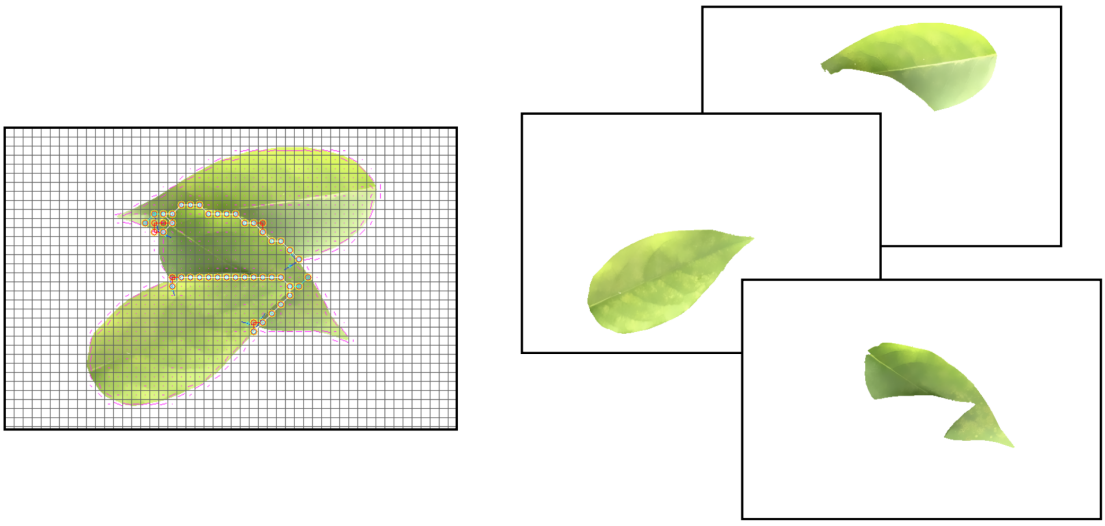












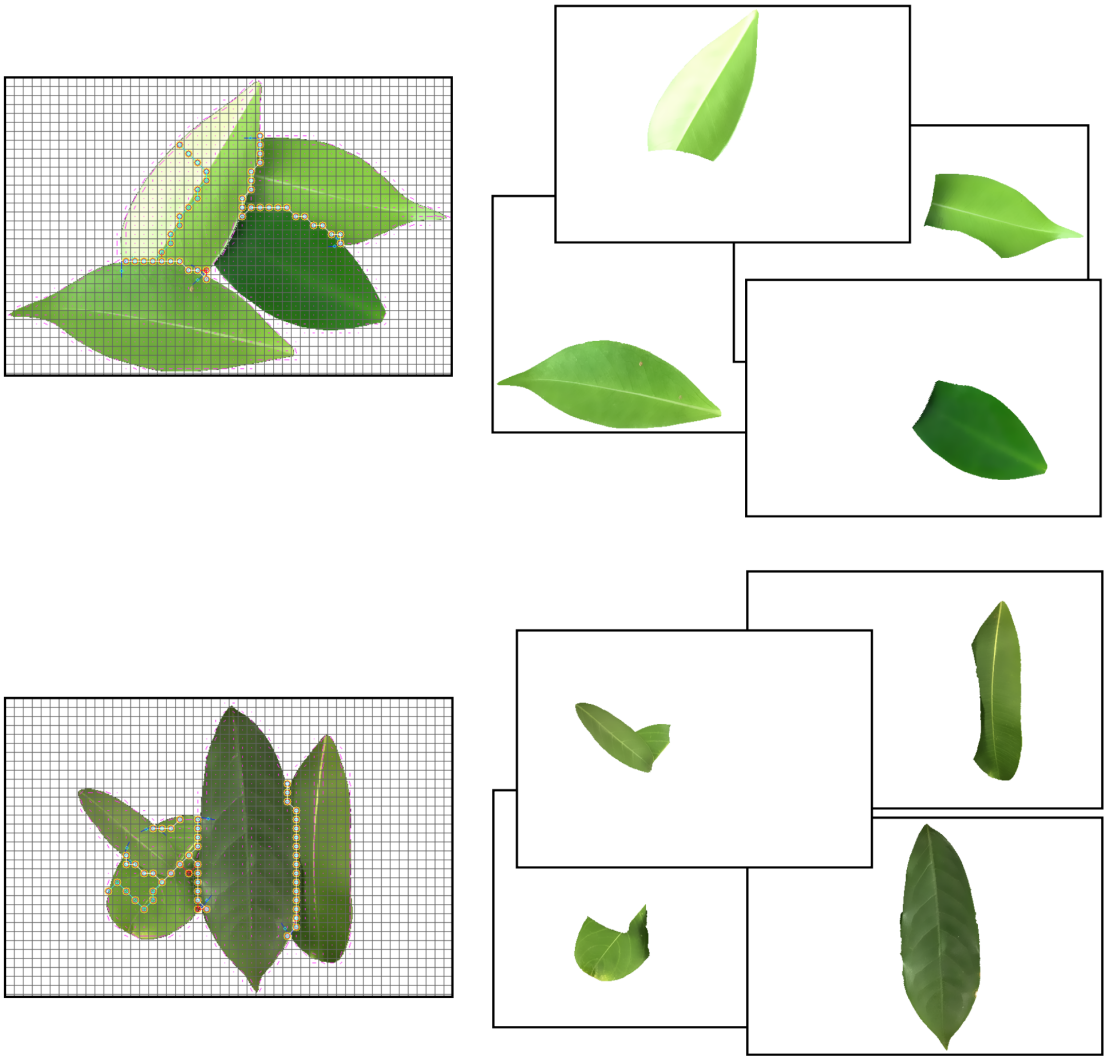


Figure 4.7: Step and result examples

4.4 Verification of the proposed method

4.4.1 Objective

The experiment was conducted to verify the performance of this proposed method using a new dataset. It also measured the accuracy regarding quantity and quality.

4.4.2 Experiment setup

This experiment used the same parameters and the same measurement as the previous experiment, except the dataset.

The previous dataset was captured in a controlled environment (controlled light), but a new dataset was collected in a natural environment. The new dataset consisted of 30 occluded leaf images. Each image contained only a single species. It was taken on a natural background with iPhone 7 at Bangkok, Thailand. Then, each image was manually eliminated background, cropped, and resized to 600×400 pixels. The first row in Fig. 4.8 represented a taken photo, and the second one represented its corresponding removed background image.

The difference between this dataset and the previous one was an environment. The first one was taken in a light photo box, while this one was taken in a real environment. An image would have the effect of the light. Moreover, when an occluded leaf image was captured in a natural environment, the leaves were still attached to their stem. There was more depth between each leaf, so leaf's shadow would be more clear compared with the previous dataset.

Figure 4.8: Examples of occluded leaf image with uncontrolled environment

4.4.3 Result and discussion

For the quantity measurement, the accuracy of the result images after applied a new dataset was described in Table 4.8. 25 out of 30 (92.59%) occluded leaf images could produce the result (i.e., a set of single leaf images) without separation error. In addition, a non-separation error meant the number of result images fit with its ground truth. The error appeared only in +1 and -1 separation error. Thus, this accuracy was a preferable result, and this proposed method supported the new dataset (images captured in a natural environment).

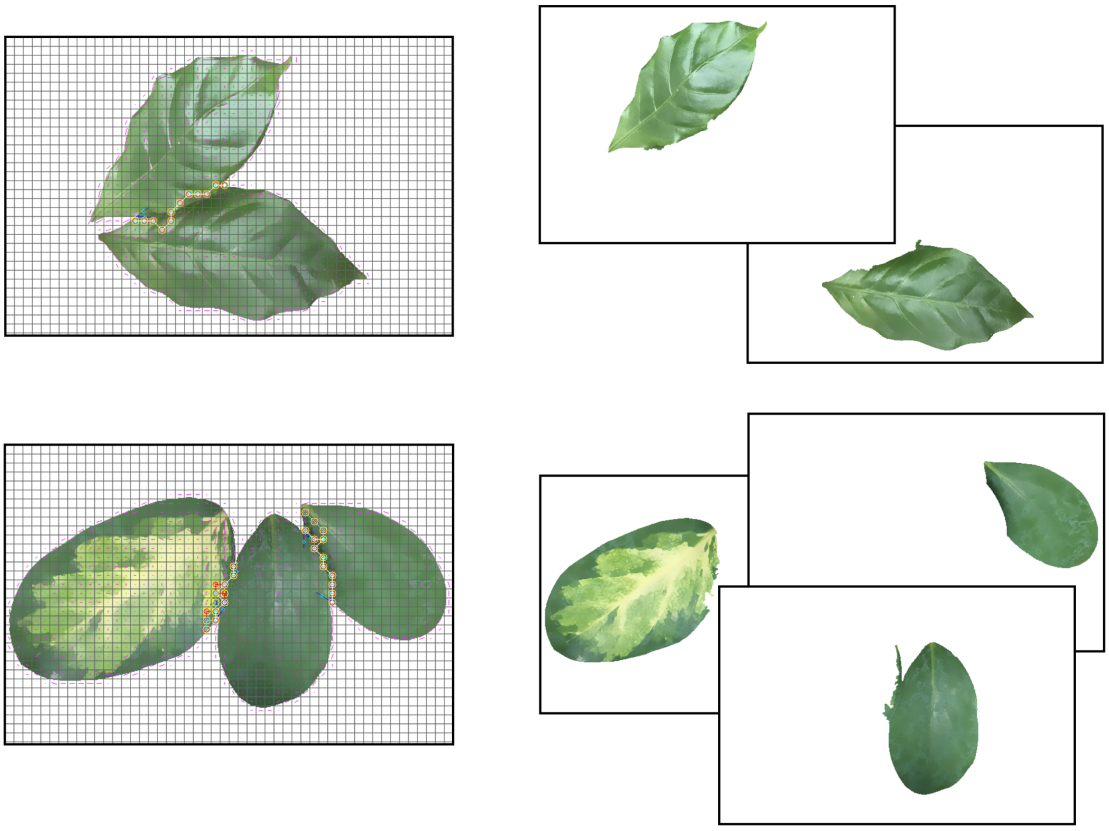
Table 4.8: The accuracy of result images in a natural environment

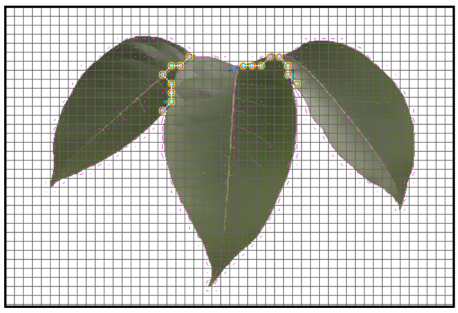
Separation error	Number of result images	
$+1$	3	11.11
0	**25**	**92.59**
-1	2	7.41
#image	30	

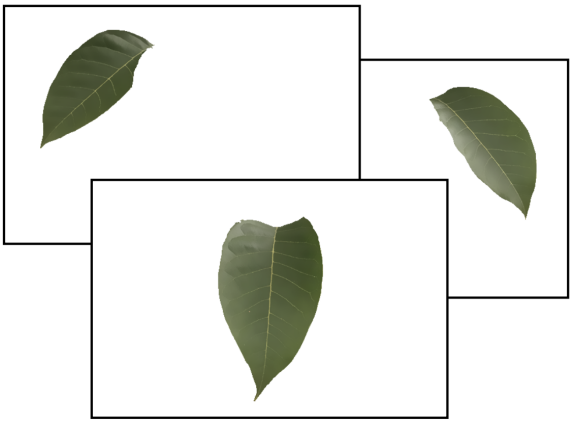
Moreover, Table 4.9 represented the quality measurement. The segmentation accuracy was applied to only result images with the non-separation error. 92.42% could be accurately segmented for the accuracy, 88.39% for the recall, 94.29% for tnr, and 91.43% for the precision. These result with a natural environment dataset was a little higher than the result applied a controlled environment, and result examples were shown in Fig. 4.9.

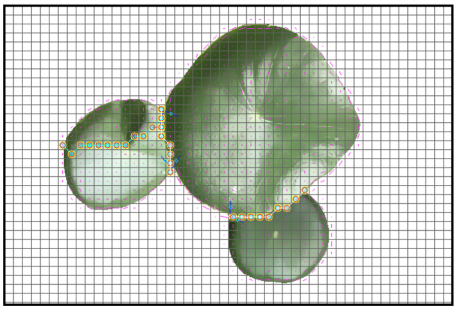
Table 4.9: Overall accuracy of the number of result images with non-separation-error in a natural environment

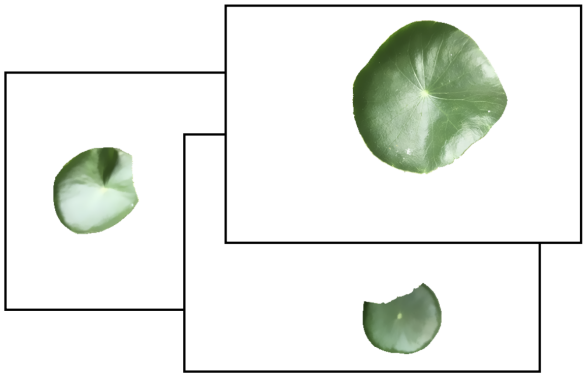
	Average
Accuracy (%)	92.42
TPR or recall (%)	88.39
TNR (%)	94.29
Precision (%)	91.43

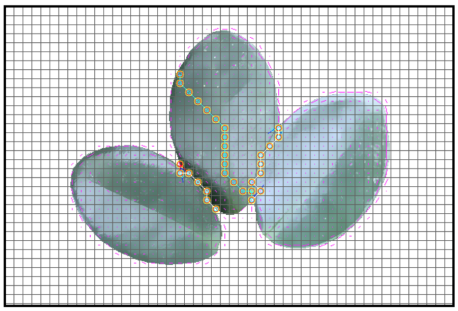


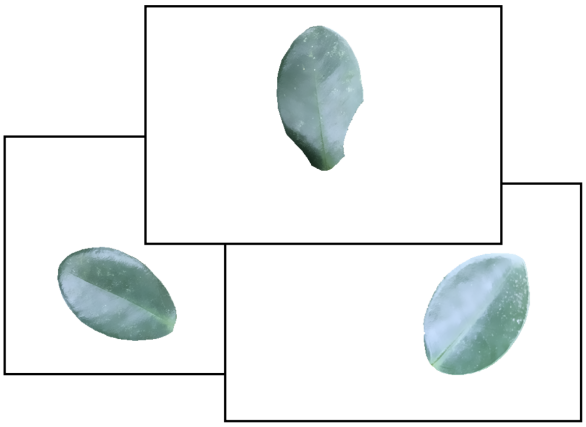












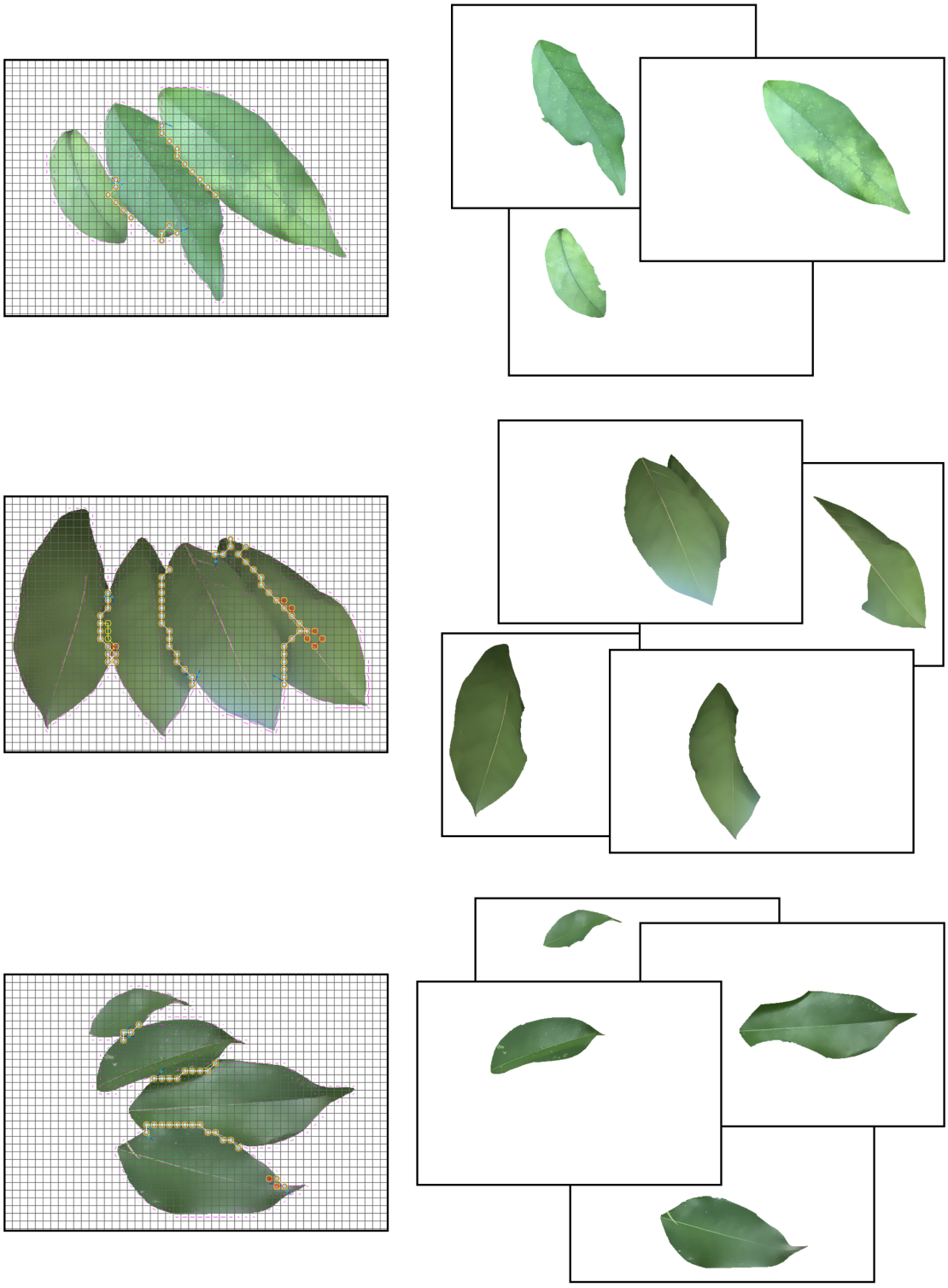


Figure 4.9: Examples of step and result example

Chapter 5

Conclusion

This proposed method is an automatic method used to separate occluded leaf image into a set of single-leaf images. To identify the region of a single leaf, intersection points and direction field are required. An intersection point is defined by threshold selection of a curvature value. Only a position with curvature value higher than a threshold is chosen. If a point is the intersection point, which means this point denotes a concave point between leaves. Moreover, the intersection points are used as the starting position of leaf estimation process. Another one, the direction field describes the average direction of edges in a local area, which composes of the angle and the magnitude. The first value denotes the direction, and another one determines how strong the direction is. Furthermore, direction field is also used to guide the estimation process. After intersection points and direction field were calculated, leaf inner-edge estimation process begins. This process uses intersection points as a starting position and direction to find leaf inner-edge, which is the line that can separate occluded leaves. Information from direction field is applied to the other leaf inner-edge directions until the background is located. Regarded that each intersection point will produce only one leaf inner-edge unless the point is the end of another edge. Next, every leaf inner-edge is refined in edge refinement, which is created for removing any irrelevant point from the edge. At

this moment, all leaf boundaries are obtained. It comes from the combination between outer contour and leaf inner-edge. After that, the region of the occluded leaves is applied to all leaf boundaries using invert operation to receive a set of rough leaf regions as a set of foreground markers for a watershed transform. Then, a background marker is generated from the other areas of the occluded leaf region. The foreground marker set and the background marker are used as predefined markers of watershed transform in leaf separation process. Finally, this process produces a single-leaf image set as an output of the proposed method.

There are two primary experiment datasets, which are the dataset with a controlled environment and the dataset with a natural environment.

For the first dataset, image dataset was collected and taken around the campus (KMITL, Thailand), with a total of 128 images. They consisted of two, three, four, and more than four occluded leaves with 35, 35, 35, and 23 images sequentially. After that, they were divided into two sets: without more than four (105 in total) and with more than four occluded leaf images (128 in total). For the first one, the proposed method correctly produced the number of result images for 80 out of 105 testing images, which is 76.19%. For the second one, it accurately generated 67.97% (87 out of 128). For the images with correctly produced result images (non-separation error), around 90% of them had been segmented successfully.

Another one was captured in a natural environment around Bangkok, Thailand. This dataset contained 30 occluded leaf images, which was manually eliminated background. Then, it was applied to the proposed method in order to produce a set of single-leaf images. The result accuracy was around 90% (25 out of 30) in which the number of result images fits with its ground truth. The segmentation accuracy in each image was up to 90%, which was a favored result.

The advantage of this proposed method is that it does not require an image for training, and the computation time is concise. Moreover, the proposed method uses only

information of the edge such as curvature, contour, and direction field. Thus, it should be light robustness shown in the experiment using the dataset with a natural environment.

A result image of this proposed method can be used as an input image for other applications, which require a single-leaf image. It can reduce manual labor task for a human.

Although this research provides the method for occluded leaf separation, the author believes that this proposed method can be used to separate other occluded objects that have a simple shape. Since, this technique uses only information of edges, such as curvature and direction field. Moreover, the result accuracy from the dataset captured in a natural environment can be used to guarantee that the light does not affect the performance of proposed method. Thus, it might be possible to separate other overlapping objects.

Bibliography

[1] S. Abbasi, F. Mokhtarian, and J. Kittler, "Curvature scale space image in shape similarity retrieval," Proceedings of Multimedia Systems, vol.7, pp.467–476, 1999.

[2] S. Beucher and C. Lantuejoul, "Use of watersheds in contour detection," International Workshop on Image Processing, pp.17–21, 1979.

[3] J.S. Cope, D. Corney, J.Y. Clark, P. Remagnino, and P. Wilkin, "Plant species identification using digital morphometrics: A review," Expert System with Applications, vol.39, no.8, pp.7562–7573, 2012.

[4] J. Cuerda, Atlas Básico de Botánica, Parramón, 2002.

[5] B. Ellis, D.C. Daly, L.J. Hickey, K.R. Johnson, and J.D. Mitchell et al., Manual of Leaf Architecture, Cornell University Press, 2009.

[6] D. Font, T. Pallej, M. Tresanchez, D. Runcan, and J. Moreno et al., "A Proposal for Automatic Fruit Harvesting by Combining a Low Cost Stereovision Camera and a Robotic Arm," Sensors, vol.14, no.7, pp.11557–11579, 2014.

[7] K. Frolov, J. Fripp, C.V. Nguyen, R. Furbank, and G. Bull et al., "Automated Plant and Leaf Separation: Application in 3D Meshes of Wheat Plants," Proceedings of 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp.1–7, 2016.

[8] R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison-Wesley, 1992.

[9] O. Janssens, J.D. Vylder, J. Aelterman, S. Verstockt, and W. Philips et al., "Leaf segmentation and parallel phenotyping for the analysis of gene networks in plants," Proceedings of 2013 European Signal Processing Conference (EUSIPCO), pp.1–5, 2013.

[10] J. Liu, L. Chen, J. Tian, and D. Zhu, "Learning-based leaf occlusion detection in surveillance video," Proceedings of 2016 IEEE Conference on Industrial Electronics and Applications (ICIEA), pp.1000–1004, 2016.

[11] O. Marques, Practical Image and Video Processing Using MATLAB, Wiley, 2011.

[12] A. McAndrew, Introduction to Digital Image Processing with MATLAB, Course Technology, 2004.

[13] F. Meyer, "Color image segmentation," Proceedings of 1992 International Conference on Image Processing and Its Applications, pp.303–306, 1992.

[14] M.L. Nguyen, V. Ciesielski, and A. Song, "Rice leaf detection with genetic programming," Proceedings of 2013 IEEE Congress on Evolutionary Computation, pp.1146–1153, 2013.

[15] S. Paris, "Fixing the Gaussian blur: the bilateral filter," A Gentle Introduction to Bilateral Filtering and its Applications Presentation, 2008.

[16] M. Rojanamontien, P. Sihanatkathakul, N. Piemkaroonwong, S. Kamales, and U. Watchareeruetai, "Leaf identification using apical and basal features," Proceedings of 2016 International Conference on Knowledge and Smart Technology (KST), pp.234–238, 2016.

[17] D. Story, M. Kacira, C. Kubota, A. Akoglu, and L. An, "Lettuce calcium deficiency detection with machine vision computed plant features in controlled environments," Computers and Electronics in Agriculture, vol.74, no.2, pp.238–243, 2010.

[18] M. Teranishi, N. Ohnishi, and N. Sugie, "Subjective contours are useful for extracting contours with very weak contrasts," Proceedings of 1993 International Joint Conference on Neural Networks, pp.139–142, 1993.

[19] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Proceedings of 1998 International Conference on Computer Vision, pp.839–846, 1998.

[20] X.F. Wang, D.S. Huang, J.X. Du, H. Xu, and L. Heutte, "Classification of plant leaf images with complicated background," Applied Mathematics and Computation, vol.205, no.2, pp.916-926, 2008.

[21] U. Watchareeruetai, Y. Takeuchi, T. Matsumoto, H. Kudo, and N. Ohnishi, "Computer vision based methods for detecting weeds in lawns," Machine Vision and Applications, vol.17, no.5, pp.287–296, 2006.

[22] C. Xia, J.M. Lee, Y. Li, Y.H. Song, and B.K. Chung et al., "Plant leaf detection using modified active shape models," Biosystems Engineering, vol.116, no.1, pp.23–35, 2013.

[23] G. Xu, F. Zhang, S.G. Shah, Y. Ye, and H. Mao, "Use of leaf color images to identify nitrogen and potassium deficient tomatoes," Pattern Recognition Letters, vol.32, no.11, pp.1584–1590, 2011.

[24] L. Zhang, P. Weckler, N. Wang, D. Xiao, and X. Chai, "Individual leaf identification from horticultural crop images based on the leaf skeleton," Computers and Electronics in Agriculture, vol.127, pp.184–196, 2016.

[25] Z.Q. Zhao, Y. Hong, P. Zheng, and X. Wu, "Plant identification using triangular representation based on salient points and margin points," Proceedings of 2015 IEEE International Conference on Image Processing (ICIP), pp.1145–1149, 2015.