Real-Time Automatic Linear Feature Detection in Images

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

by
Guang Zeng
December 2008

Accepted by:
Dr. Stanley Birchfield, Committee Chair
Dr. Adam W. Hoover
Dr. Christina E. Wells
Dr. Ian D. Walker

Abstract

Linear feature detection in digital images is an important low-level operation in computer vision that has many applications. In remote sensing tasks, it can be used to extract roads, railroads, and rivers from satellite or low-resolution aerial images, which can be used for the capture or update of data for geographic information and navigation systems. In addition, it is useful in medical imaging for the extraction of blood vessels from an X-ray angiography or the bones in the skull from a CT or MR image. It also can be applied in horticulture for underground plant root detection in minirhizotron images.

In this dissertation, a fast and automatic algorithm for linear feature extraction from images is presented. Under the assumption that linear feature is a sequence of contiguous pixels where the image intensity is locally maximal in the direction of the gradient, linear features are extracted as non-overlapping connected line segments consisting of these contiguous pixels.

To perform this task, a point process is used to model a network of line segments in images. Specific properties of line segments in an image are described by an intensity energy model. Aligned segments are favored while superposition is penalized. These constraints are enforced by an interaction energy model. Linear features are extracted from the line segments network by minimizing a modified Candy model energy function using a greedy algorithm whose parameters are determined in a data-driven manner. Experimental results from a collection of different types of linear features (underground plant roots, blood vessels, and urban roads) in images demonstrate the effectiveness of the approach.

Dedication

I dedicate this work to my parents and my two sisters Yue and Ming.

Acknowledgments

I would like to acknowledge and express my appreciation for the immeasurable support and guidance contributed by Dr. Stan Birchfield in the preparation of this dissertation. Additionally, I also want to express my gratitude to Dr. Christina E. Wells, Dr. Ian D. Walker and Dr. Adam W. Hoover, not only for their input in the preparation of this dissertation, but also for the many hours of quality instruction they have provided to me in my graduate studies leading up to this point.

Finally, I would like to express my appreciation to my girl friend, Hong Dong, for her unfaltering support and encouragement not only during my graduate studies but throughout our life together.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Linear feature detection in digital images is an important low-level operation in computer vision that has many applications (Figure 1.1). In remote sensing tasks, it can be used to extract roads, railroads, or rivers from satellite or low-resolution aerial images, which can be used for the capture or update of data for geographic information and navigation systems [32] [60] [63]. In addition, it is useful in medical imaging for the extraction of blood vessels from an X-ray angiography [7] [10] [4] [62] or the bones in the skull from a computed tomography (CT) or magnetic resonance (MR) image [35]. It also can be applied in the area of horticulture for underground plant root detection in minirhizotron images [67] [18].

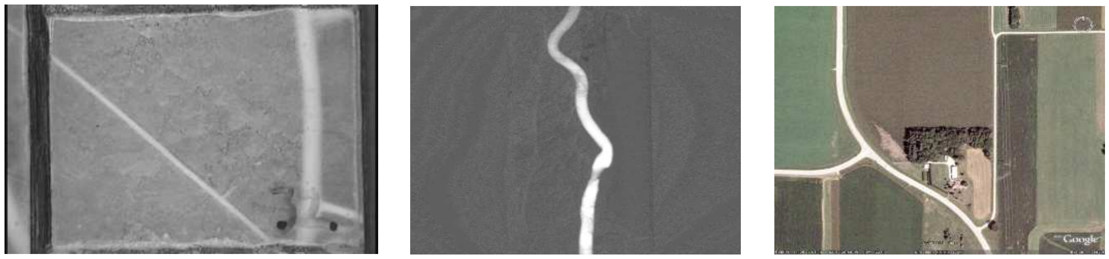


Figure 1.1: Linear features in sample images. From *left* to *right*: Underground roots in minirhizotron image, blood vessel in X-ray digital subtraction angiography (DSA) image, urban roads in satellite image.

1.1 Previous Work

There are a number of techniques in the literature on linear feature detection. Previous work can be classified into four categories. The first approach detects linear features by considering the gray values of the image only and uses purely local criteria such as local gray value differences [19] [70] [9]. Since this will generate many false positive detections, elaborate and computationally grouping schemes such as Autoregressive models [70] [3] and the Hough transform [5] have to be used to select salient lines in the images.

The second type of technique is called template- or model-based. This type of approach assumes that linear features have a locally homogeneous intensity distribution and significant contrast to the background. It requires a series of directional filters to be applied to the image, such as steerable filters [27], 2D matched filters [7] [67] [26] [47], maximum gradient profiles [9], or directional morphological filtering [57].

In Can et al. [4], the authors regard linear features as objects having parallel edges. First, seed points are selected from pixels with local maximum intensity, then local line orientations are determined for selected seed points. Two specially tuned edge detection filters are applied perpendicular to the linear feature, where each filter detects either the left or right edge of the linear feature. The advantage of this type of approach is that the constructed filters can be iterated in scale-space to detect linear features of arbitrary width. However, because the directional filters are not separable, the approaches are computationally expensive.

The third type of approach regards the image as a function $f(x, y)$ and extracts linear features from it by using differential geometric properties. The basic idea is to locate the position of ridges and ravines in the image function. Ridges are found by linking the points on a contour line of the image where the curvature is maximum [31] or found at points where one of the principle curvatures of the image is locally maximal [38]. One drawback of such an approach is their sensitivity to ridges that have a small gradient, which cause the contour lines to become widely separated. In addition, it usually generates multiple responses to a single line with a flat profile.

As an example of this approach, Busch [39] and Wang et al. [65] detect ridges and ravines by locally approximating the image function by its second- or third-order Taylor polynomial. The coefficients of this polynomial are usually determined using the facet model, i.e., by a least squares fit of the polynomial to the image data over a window of a certain size. The direction of the line is determined from the Hessian matrix of the Taylor polynomial. Line points are found by selecting pixels that have a high second directional derivative perpendicular to the line direction.

Steger [58] uses a modification of the differential geometric approach to detect lines and their corresponding edges. By using a Gaussian curve to estimate the derivatives of the image, the algorithm scales lines and their corresponding edges. Carried out in a scale-space analysis, the bias in the extracted line and edge position can be predicted analytically and then removed. One drawback of the algorithm is that it can only be used to detect lines within a certain range of widths.

The fourth type of approach is based on active contour. Active contour techniques consider the linear features as ribbons with parallel borders. The curvature and the gradient are used to define an energy function. The final contour fits the linear feature following a differential equation whose solution corresponding to a local minimum of the energy [44]. Laptev et al. [32] and Neuenschwander et al. [40] introduced a multi-scale based “ziplock” strategy for detecting partially occluded linear features. However, this type of approach need user to give control points for the linear features.

1.2 Our Approach

Several years ago we developed a technique [66] [67] for linear feature detection which enhanced the contrast between linear features and background using 2D matched filtering, and then binarized the linear features using local entropy thresholding. However, the application of this approach is limited by its low performance for linear feature detection in noisy backgrounds and its computation time. These limitations motivated the present work, in which we have developed an algorithm for linear feature detection that is both faster and more accurate than the previous approach.

Some elaborate models based on linear feature network geometry have been able to improve detection. In Tupin et al. [63], a two-step procedure is applied to detect linear features. First, the main segments in the image are extracted. Second, a Markov random field (MRF) is built on a graph composed of the detected segments plus some admissible segments connecting the previous ones. A binary random variable assesses the cost, depending on whether the segments defining the graph represent linear features. An energy function is derived and the linear features are extracted by minimizing this energy function using a simulated annealing algorithm. This is the first approach designed for linear feature detection by building MRF models that go beyond pixel level and define graphs in which nodes represent some higher level primitive. However, this approach must determine the number of nodes and their relations before the model optimization. Another drawback is that this approach is not able to combine segments, create new segments, remove segments or to adjust their relations during the optimization process.

These problems can be handled using *point processes*. In mathematics, a point process is a random element whose values are “point patterns” on a set. Point processes are well studied objects in probability theory and a powerful tool in statistics for modeling and analyzing spatial data. A natural extension of Markov Random Field approaches (MRFs), point processes model images as random configuration of geometric shapes and provide a natural setup for the inclusion of a prior knowledge on the spatial pattern of features. Such models have been used by various authors for extracting roads, trees, or buildings from images [1] [53] [59] [41] [42] [15]. To optimize the energy model, many researchers use a simulated annealing algorithm called Reversible Jump Markov Chain Monte Carlo (RJMCMC) based on Monte Carlo dynamics for finite point processes [59] [60] [62]. Such algorithms are computationally expensive.

In this thesis we also use point processes to model the detection of linear features in images. However, we make several departures from previous approaches. First, we determine the parameters of the algorithm in a data-driven manner by learning the optimal separating hyperplane between pixels belonging to the linear features and pixels belonging to the surrounding background in an off-line training step. Secondly, we introduce an extremely fast procedure to detect seed points in the image that discards more than 99% of the image data while still recovering a satisfactory number of points. Thirdly, we show how to combine these seed points in a RANSAC-like manner to fit and extend contiguous piecewise-linear segments to the seed points. Fourthly, our approach, unlike previous methods that have detected the piecewise linear segments as a network, actually detects the individual segments themselves, thus recovering more discriminatory information about the image. Finally, we introduce a fast procedure that we call *constrained floodfill* to determine the region of pixels belonging to a linear feature, in addition to the segment. This procedure is much faster than the computationally expensive hierarchical A^*-search used by Erz et al. [18] to expand detected edge points. The result is a real-time algorithm that is highly effective at extracting piecewise linear segments.

While the presented algorithm is broadly applicable to multiple types of images, including detecting urban roads in satellite images and blood vessels in Digital Subtraction Angiography (DSA) images, we concentrate our efforts in this thesis on detecting roots in minirhizotron images. Minirhizotron imaging is a state-the-art technique for obtaining high-quality digital images of underground plant roots continuously and non-destructively. Minirhizotrons are transparent plastic tubes buried at an angle in the soil near the plants to be observed [64]. To make root observations with minirhizotrons, a miniaturized color camera on a telescopic hadle is lowered into each tube to capture digital images of the roots that have grown against its outer surface. This process is repeated at regular intervals over a number of years to build an extensive image library of thousands of individual roots as they appear and disappear through time. Fast and accurate underground plant root extraction in minirhizotron images is an important application of the presented algorithm in the area of horticulture [69].

The outline of the thesis is as follows. In Chapter 2, the basic idea of point processes are briefly described, to provide some background for the model used. Then in Chapter 3, a probabilistic model using a first-order Markov assumption with intensity and interaction energy is presented. Two linear discriminant analysis techniques are introduced and compared in Chapter 4, in order to explain the procedure used to obtain the data-driven parameters.

The linear feature detection algorithm itself can be seen as a greedy approach to minimizing the energy of the point process model. The block diagram of the algorithm is shown in Figure 1.2. In Chapter 5, we describe the procedure by which seed points are selected, centerlines of the linear features are fit to the seed points, centerlines are extended (or traced) into areas without seed points, and pixel regions are computed using constrained floodfill. To discard false positives, Chapter 6 describes an approach for discriminating true objects from distracting bright background objects by building a strong classifier from a series of weak feature classifiers using the Adaboost algorithm. In Chapter 7, we show how to apply an accurate length estimator to measure the length of the centerline, which is computed using a shortest path search algorithm. Thorough experimental results from different types of plant roots in minirhizotron images are given in Chapter 8, validating the effectiveness of the technique. In addition, results of the algorithm to other application areas such as blood vessel detection in Digital Subtraction Angiography (DSA) images and urban road detection in satellite images are given in Chapter 9. Finally, conclusions and future work are presented in Chapter 10. For comparison, our previously developed algorithm [66] [67] based on two-dimensional matched filtering and local entropy thresholding is briefly described in Appendix A.

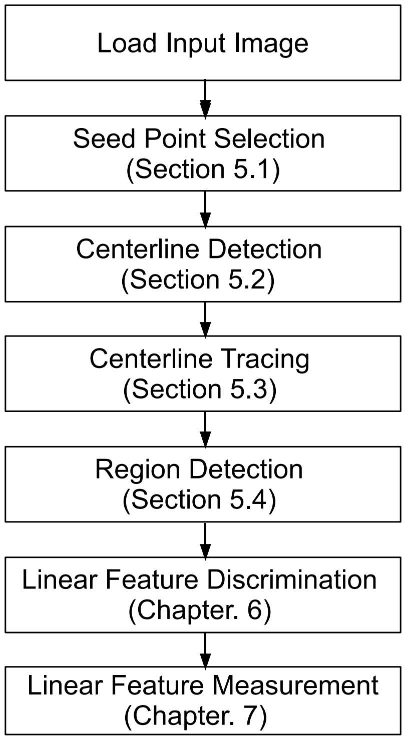

Load Input Image
Seed Point Selection (Section 5.1)
Centerline Detection (Section 5.2)
Centerline Tracing (Section 5.3)
Region Detection (Section 5.4)
Linear Feature Discrimination (Chapter. 6)
Linear Feature Measurement (Chapter. 7)

Figure 1.2: Block diagram of the linear feature detection algorithm.

Chapter 2

Point Processes

As the name indicates, the concept of point processes originated with the study of random point sequences on the time axis. Such processes still play an important role, for example, in models of queuing or telecommunication. A spatial point process is a model for a random pattern of points in an n-dimensional space. Today spatial point processes are used to model phenomena in a wide variety of scientific disciplines, including seismology, ecology, forestry, geography, spatial epidemiology, and material science. Spatial point process models allow the modeling of images as random configurations of geometric shapes and provide a natural setup for the inclusion of a priori knowledge of the spatial pattern of features. Such models were first used in image processing by Baddeley et al. [1] for detecting an unknown number of objects in an image. They provide models that go beyond the pixel level by defining a graph in which the nodes represent some higher level primitive. The vertices induce a neighborhood system on which constraints can be imposed using the definition of the nodes. To handle more complex applications such as road or vascular tree extraction, a marked point process model [60] [62] is explored. Additional parameters (called marks) are associated to each point which define some geometric property of the underlying object. Considering the pairwise interaction between points, a marked Gibbs point process model [60] is used for describing the repulsive interaction between points.

2.1 Point Processes

In mathematics, a point process is a random process that generates "point patterns" on a set K. We consider a configuration of points ω in K as an unordered set of points $\omega = [\omega_1, ..., \omega_n]$, where $\omega_i \in K$ and n is the number of points in the configuration. Denoting Ω as the set of all possible finite configurations and $\mathcal{F}$ as the σ-algebra associated with Ω, we define a point process $\mathbf{X}$ of points in K as a measurable mapping from the probability space (K, $\mathcal{B}$, ν) to the measurable space (Ω, $\mathcal{F}$). Due to the finiteness of the considered configurations along with the boundedness of K, $\mathcal{F}$ is well defined. Accordingly, a point process is a random process whose realizations are random configuration of points.

We call (K, $\mathcal{B}$, ν) the probability triple. The sample space K is a nonempty set whose elements are known as outcomes or states of nature and are often given the symbol ω. The set of all the possible outcomes of an experiment is known as the sample space of the experiment.

The second term, $\mathcal{B}$, is a σ-algebra of subsets of K. Its elements are called events, which are sets of outcomes for which one can compute a probability. The third term ν is a measure on $(K, \mathcal{B})$ such that $\nu(K) = 1$. It is a probability measure on (K, $\mathcal{B}$) which assigns to each set $A \in \mathcal{B}$ a value $\nu \in [0, 1]$ representing the probability that A describes the outcome of the random experiment.

For a set K, a σ-algebra $\mathcal{B}$ is a nonempty collection of subsets of K such that the following properties hold [28]:

1. $K \in \mathcal{B}$

2. If a set $A \in \mathcal{B}$, then its complement $A^c \in \mathcal{B}$

3. If $A_n \in \mathcal{B}$ for $n = 1, 2, \ldots$, then $\cup_{n=1}^{\infty} A_n \in \mathcal{B}$

The pair $\langle K, \mathcal{B} \rangle$ is also a field of sets, sometimes called a *σ-field* or a *measurable space*. An element of $\mathcal{B}$ is called a $\mathcal{B}$-measurable subset of K.

Example. Consider a random experiment of three consecutive coin tosses, where order matters. The set of outcomes is given by $K = \{$ *HHH, HHT, HTH, HTT, THH, THT, TTH, TTT* $\}$. Some possible σ-algebras for K are

- $\mathcal{B} = \{\emptyset, \boldsymbol{K}\ \}$
- $\mathcal{B} = \{\emptyset, \boldsymbol{K}, \{$ *HHH* $\}, \{$ *HHT, HTH, HTT, THH, THT, TTH, TTT* $\}\ \}$
- $\mathcal{B} = \{\emptyset, \boldsymbol{K}, \{$ *HHT* $\}, \{$ *HHH, HTH, HTT, THH, THT, TTH, TTT* $\}\ \}$
- $\mathcal{B} = \{\emptyset, \boldsymbol{K}, \{$ *HHH, HHT* $\}, \{$ *HTH, HTT, THH, THT, TTH, TTT* $\}\ \}$
- $\mathcal{B} = \{\emptyset, \boldsymbol{K}, \{$ *HHH* $\}, \{$ *HHH, HHT* $\}, \{$ *HHT, HTH, HTT, THH, THT, TTH, TTT* $\}, \{$ *HTH, HTT, THH, THT, TTH, TTT* $\}\ \}$
- $\mathcal{B} = \{\emptyset, \boldsymbol{K}, \{$ *HHH, HHT, HTH, HTT* $\}, \{$ *THH, THT, TTH, TTT* $\}\}$

and so on.

Note that in each case the σ-algebra contains K, the complement of any event, and the union of any (finite or countably infinite) sequence of events. The most common σ-algebra is the Borel σ-algebra. The Borel σ-algebra on a topological space $\mathbb{R}^n$ of a set K is a σ-algebra of subsets of K associated with the topology of K. More specifically, it is the smallest σ-algebra containing all open intervals in K. The

elements of the Borel σ-algebra are called Borel sets or Borel-measurable sets. In the above example, the Borel σ-algebra contains all the subsets of K:
$\mathcal{B} = \{\varnothing, \boldsymbol{K}, \{\ HHH\ \}, \{\ HHT\ \}, \{\ HTH\ \}, \ldots, \{\ HHH, HHT\ \}, \{\ HHH, HTH\ \}, \ldots, \ldots, \{\ HHT, HTH, HTT, THH, THT, TTH, TTT\ \}\ \}$.

Some examples in Figure 2.1 illustrate the Borel σ-algebra on the set of real numbers:

(a) A set A_1 containing real intervals on an open half line $A_1 = \{t \mid t \in (a, \infty)\}$ is a Borel set.

(b) A set A_2 containing real intervals on a closed half line $A_2 = \{t \mid t \in [a, \infty)\}$ is a Borel set, which can be expressed as $(-\infty, a)^c$

(c) A set A_3 containing closed intervals $A_3 = \{t \mid t \in [a, b]\}$ is a Borel set, which can be expressed as $((-\infty, a) \cup (b, \infty))^c$

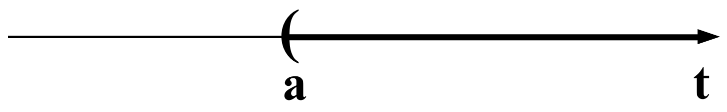
(d) A set A_4 containing half-open and half-closed intervals $A_4 = \{t \mid t \in (a, b]\}$ is a Borel set, which can be expressed as $((a, \infty)^c \cup (b, \infty))^c$

(e) A set A_5 containing only one real number $A_5 = \{t = a\}$ is a Borel set, which can be expressed as $((-\infty, a) \cup (a, \infty))^c$

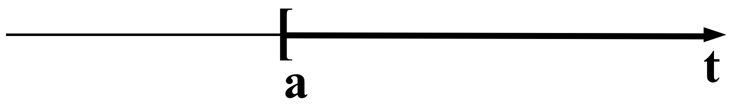
The probability measure ν is a function from $\mathcal{B}$ to the real numbers that assigns to each event a probability between 0 and 1. It has the following properties:

1. $\nu(K) = 1$
2. If A_1, A_2, ... is a sequence of disjoint sets in $\mathcal{B}$, then $\nu(\bigcup_{k=1}^{\infty} A_k) = \sum_{k=1}^{\infty} \nu(A_k)$

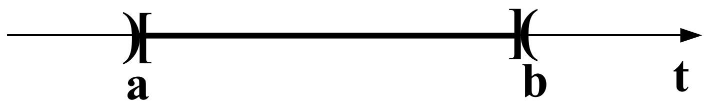
A probability measure on the coin toss example we mentioned earlier is shown below. If the coin has probability p for H and *1-p* for T, and set A_H presents H on



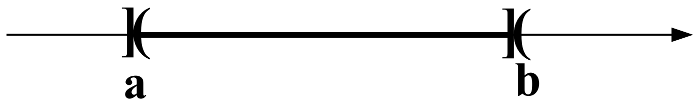
(a) Open half lines are Borel sets.



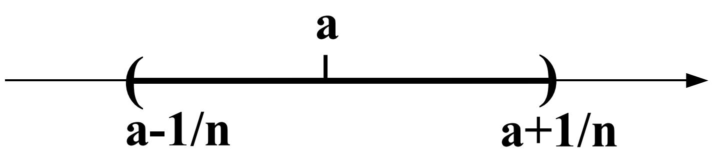
(b) Closed half lines are Borel sets.



(c) Closed intervals are Borel sets.



(d) Half-open and half-closed intervals are Borel sets.



(e) A set containing a real number is a Borel set.

Figure 2.1: Some examples of Borel sets on the real number line.

the first toss, then

$$\begin{aligned}
\nu\{A_H\} &= \nu\{HHH, HHT, HTH, HTT\} \\
&= \nu\{HHH\} + \nu\{HHT\} + \nu\{HTH\} + \nu\{HTT\} \\
&= p^3 + p^2(1-p) + p(1-p)p + p(1-p^2) \\
&= p
\end{aligned}$$

Because ν is a function defined on $\mathcal{B}$ and not on K, the set of events is not required to be the complete power set of the sample space; that is, not every set of outcomes is necessarily an event.

The simplest model for point processes is the *Poisson point process.* Poisson point process describes a collection of random variables indexed by intervals. Let ν be a positive measure on K. A Poisson point process $\mathbf{X}$ with intensity ν has the following properties [13] [42]:

1. For every Borel set $A \in K$, the random variable $N_{\mathbf{X}}(A)$, giving the number of points of $\mathbf{X}$ falling in the set A, follows a discrete Poisson distribution with mean $\nu(A)$,
$$P(N_{\mathbf{X}}(A) = n) = e^{-\nu(A)} \frac{\nu(A)^n}{n!}$$

2. For every finite sequence of non intersecting Borelian sets $D_1, ..., D_p$, the corresponding random variables $N_{\mathbf{X}}(D_1), ..., N_{\mathbf{X}}(D_p)$ are independent.

2.2 Spatial Point Processes

In the simplest case, a spatial point process is a finite random subset of a given bounded region $S \subset \mathbb{R}^2$, and a realization of such a process is a spatial point pattern

$\omega = \{\omega_1, ..., \omega_n\}$ of $n \geq 0$ points contained in S [37], where $\omega_i \in \mathbb{R}^2$. Let A be some metric space and assume this space to be Polish, i.e., complete and separable. For each bounded Borel set $B \subset A$, let $\phi(B)$ be the number of events in B. Thus we can identify a point configuration with a counting measure ϕ on Borel sets on A. Let N be the set of all such measures and $(K,\, \mathcal{A},\, \nu)$ some probability space. On N we define $\mathcal{N}$ as the smallest σ-algebra generated by sets of the form $\phi \in N: \quad \phi(B) = n$, $\forall\, n \in 0, 1, 2, ...$ A spatial point process $X \subset A$ can then be regarded as a mapping from $(K,\, \mathcal{A},\, \nu)$ into $(N,\, \mathcal{N})$.

We say that the point process is defined on S, and we write $x = \varnothing$ for the empty point pattern. The number of points, $n(X)$, is a random variable, and an equivalent approach is to specify the distribution of the variables $N(B) = n(X_B)$ for subsets $B \subseteq S$ where $X_B = X \cap B$.

If it is not known on which region the point process is defined, or if the process extends over a very large region, or if certain invariance assumptions such as stationarity are imposed, then it may be appropriate to consider an infinite point process on $\mathbb{R}^2$. We define a spatial point process $\mathbf{X}$ on $\mathbb{R}^2$ as a locally finite random subset of $\mathbb{R}^2$, i.e., $N(B)$ is a finite random variable whenever $B \subset \mathbb{R}^2$ is a bounded region. We say that $\mathbf{X}$ is stationary respective isotropic if its distribution is invariant under translations in $\mathbb{R}^2$ and respective rotation about the origin in $\mathbb{R}^2$.

One basic statistics for describing spatial point patterns is *intensity*. The intensity of the process is defined as the density, or average number of events per unit area,

$$\mu(x) = \lim_{|dx| \to 0} \left\{ \frac{E[N(dx)]}{|dx|} \right\} \tag{2.1}$$

where $\mu\,(x)$ is the intensity at a given point x, $E[X]$ is the expectation of a random variable X, and $N(dx)$ is the number of events within an infinitesimal region that contains the point x.

2.3 Marked Point Processes

The configurations of points described so far only include simple points of $\mathbb{R}^2$. To describe random configurations of geometrical objects, marked point processes are used. A marked point process is a point process for which a mark $m_i \in M$ is associated to each point ω_i. In this case we consider a point process on $K \times M$ as the random sequence $\psi = \{(\omega_n, m_n)\}$ where $m_n \in M$ is the label corresponding to each ω_n. M is the space of labels and $\mathcal{M}$ is the associated Borel σ-algebra. When the label space M is equipped with a probability measure ν_M, we say that we have a *marked point process* if the distribution of the location only is a point process on K. It is a measurable mapping from some probability space into $(\Omega, \mathcal{F})$. $\mathcal{F}$ is the σ-algebra generated by the mappings that count the number of marked points in Borel sets $A \subseteq K \times M$. A marked point process is usually called an object point process if the marks represent the geometrical parameters of an object.

Figure 2.2 is an example of studying forest patterns by means of marked point processes. It shows the distribution of some trees (*circles*) with different diameters separated from each other with random distances in a rectangular plot. To describe a tree p_i in our process, we need three parameters: its position (x_i, y_i) and its diameter r_i (see Figure 2.3).

To summarize, a tree of our marked process is distributed in the space $M = [0, X_{max}] \times [0, Y_{max}] \times [0, R_{max}]$. $(x_i, y_i) \in [0, X_{max}] \times [0, Y_{max}]$ and r_i stands in the interval $[0, R_{max}]$. A realization of our marked point process is an element of M^n, $n \in \mathbb{N}$.

The simplest marked point process is the Poisson marked point process with

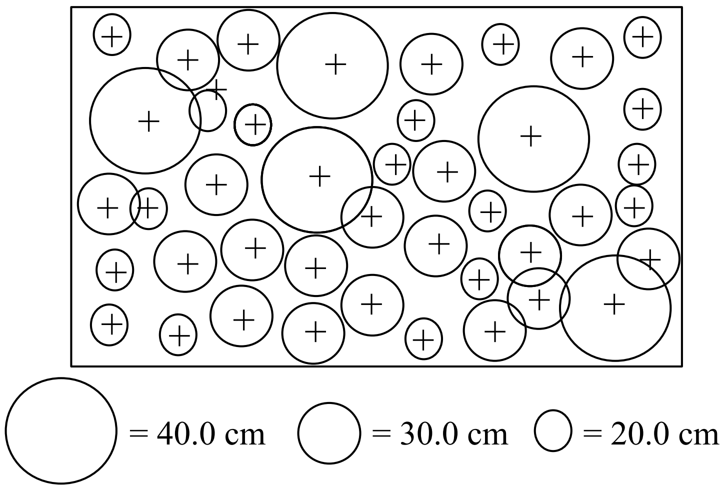


Figure 2.2: A toy example of studying trees distributed pattern in a rectangular plot by means of marked point processes. Locations of trees (+) with diameters proportional to the diameter of the circle in the figure.

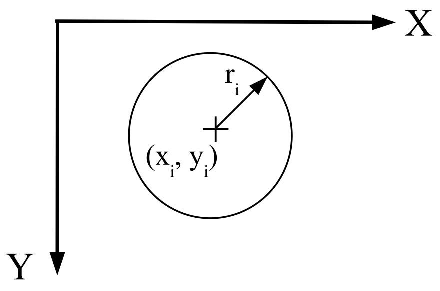


Figure 2.3: The object i in the marked point process: the point (x_i, y_i) and its associated mark r_i.

the probability measure

$$\begin{aligned}\mu(F) &= \sum_{n=0}^{\infty} \frac{e^{-\nu(K)}}{n!} \int_{K\times M} ... \int_{K\times M} \mathbf{1}_F\{(\omega_1, m_1)..., (\omega_n, m_n)\} \\ &\quad \times d\nu(\omega_1)...d\nu(\omega_n)d\nu_M(m_1)...d\nu_M(m_n) \qquad (2.2)\end{aligned}$$

for all $F \in \mathcal{F}$, where $\mathbf{1_F}\{(\boldsymbol{\omega_1}, \mathbf{m_1})..., (\boldsymbol{\omega_n}, \mathbf{m_n})\}$ is the indicator function counts the number of marked points in Borel sets $F \in \mathcal{F}$.

According to a Poisson law of intensity $\nu(K)$, this process distributes points uniformly in K. The point marks are chosen independently according to ν_M.

2.4 Gibbs Point Processes

The Poisson marked point process does not take into account interaction between the marked points (Figure 2.4.a). To allow for this, a Gibbs point process model is constructed by capturing the pairwise interactions [29] [49]. There, the joint probability density $f(\omega_1, ..., \omega_N)$ of the positions $\omega_1, ..., \omega_N$ of a fixed number of N objects in the configuration ω is,

$$f(\omega_1, ..., \omega_N) = c\beta^N exp(-U(\omega_1, ..., \omega_N)) \qquad (2.3)$$

where c is the normalizing constant, β is the intensity of the point process which controls the average number of points in a realization, and N is the number of points in the configuration ω. $U(\omega_1, ..., \omega_N)$ is the total potential energy of the object configuration.

The relation with an interaction point process follows naturally by writing the interaction potential energy as:

$$U(\omega_1, ..., \omega_N) = \sum_{\omega_i \in \omega} \beta(\omega_i) + \sum_{\omega_i, \omega_j \in \omega} \gamma(\omega_i, \omega_j) + \dots \tag{2.4}$$

where $\beta(\omega_i)$ is the 1-clique potential function, $\gamma(\omega_i, \omega_j)$ is the 2-clique interaction function.

In our research, we ignore higher-order clique interaction functions. Therefore, U can be written as a sum of pairwise potential energy functions $\phi(\mid \omega_i - \omega_j \mid)$, which describe the interaction between objects. $\phi(\mid \omega_i - \omega_j \mid) = 0$ if and only if the two points ω_i and ω_j are not mutually interacting.

$$U(\omega_1, ..., \omega_N) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \phi(\mid \omega_i - \omega_j \mid) \tag{2.5}$$

Many different functions ϕ have been studied in the literature as models for spatial point processes, including “hard core” models limiting particles separations to distances greater than a prescribed limit r (i.e., a system of hard spheres or discs of diameter r) and “soft core” models resulting from weak repulsive forces.

One of the assets of Gibbs point processes in applications is that interpretation is possible in terms of interaction. Indeed, inspecting the pair-potential function, positive values of $\phi(r)$ indicate rejection on a scale defined by r and values of magnitude zero mean vanishing interaction; negative values of $\phi(r)$ show attraction at this r.

A simple example of a Gibbs process is the Strauss process [61], for which

$$\phi(r) = \begin{cases} \gamma, & \text{if } \ r < R \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

Accordingly, points with mutual distance less than a fixed radius R are neighbors (and interacting). If $S(x)$ is the number of neighboring pairs, then Equation 2.3 can be rewritten in the form,

$$f(\omega_1, ..., \omega_N) \propto \beta^N \gamma S(x) \tag{2.7}$$

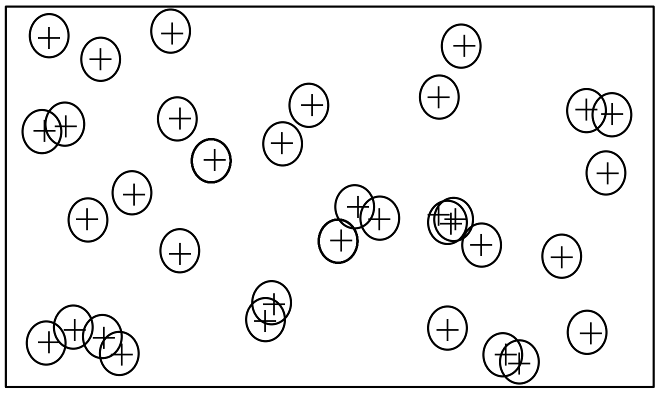
This model holds only for nonnegative γ corresponding to rejection interaction. If $\gamma = \infty$, a hard core model results. To obtain a normalizable model, we must have $0 \leq \gamma < 1$. This is then a repulsive process penalizing overlapping objects. A simulation of a homogeneous Poisson process and a Strauss process with the same average number of objects [14] is shown in Figure 2.4.

It is common for the Gibbs model to include only pairwise interaction processes. One particular and important subclass of Gibbs point process is Markov point process. In a Markov process, the probability of a particular configuration is a function of both the individual points and the interactions between the points. By the Hammersley-Clifford theorem, this can be written as the product of clique interaction functions:

$$\begin{aligned} p(\omega) &= \alpha \prod_{\omega_i \in \omega} e^{-\beta(\omega_i)} \prod_{\omega_i, \omega_j \in \omega} e^{-\gamma(\omega_i, \omega_j)} \qquad (2.8) \\ &= \alpha \exp(-U(\omega)), \qquad (2.9) \end{aligned}$$

where α is a constant that ensures $\sum_{\omega \in \Omega} p(\omega) = 1$, and $U(\omega) = \sum_{\omega_i \in \omega} \beta(\omega_i) + \sum_{\omega_i, \omega_j \in \omega} \gamma(\omega_i, \omega_j)$. Without loss of generality, the energy term U enables us to consider the process to be a Gibbs process [50] .

For our problem, the data to be analyzed consists of linear features such as plant roots spread in a finite window K. We want to extract the centerline of these linear features. It is natural to consider the centerline $\mathbf{s}$ we want to detect as a set


(a)

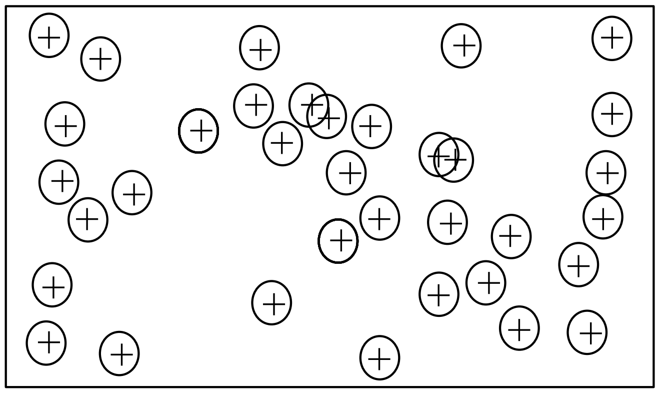

(b)

Figure 2.4: The simulation results of a homogeneous Poisson process (*top*) and a Strauss process (*bottom*) with the same average number of objects.

of random segments being the realization of a marked Gibbs point process. The probability density of such a marked Gibbs point process is given by

$$f(s) \propto \beta^n exp[-(U_D(s) + U_I(s))] \tag{2.10}$$

with the term $U_D(s)$ and $U_I(s)$ being the data energy and the interaction energy, respectively. Note that each point in the point process represents a segment in the network of segments rather than an individual pixel.

In the following chapter, we will present the two components of the energy function. We will apply the model to describe two-dimensional centerline networks of linear features. Considerations about minimizing the energy of such models using a greedy algorithm will be given in Chapter 5.

Chapter 3

Energy Model

Our definition of linear feature is a sequence of contiguous pixels where the image intensity is locally maximal in the direction of the gradient. In the method proposed here, we suppose that the linear feature can be expressed as a sequence of non-overlapping connected segments consisting of these contiguous pixels. Locally, the curvature between neighboring segments is assumed to be small. However, we notice that just a few short linear features can be represented by isolated segments. A natural choice for simulating the interaction energy between random segments becomes the marked Gibbs point process model [60].

We first present an overview of our energy model and the marks we added to each point (segment) describing the configurations of geometrical objects. Then, we propose an energy formulation based on both an intensity (data) term which measures the coherence between the linear features configuration and the image, and a modified Candy interaction term which takes into account some interactions existing between neighboring linear features. In the chapter, the energy model is presented using the example of plant root detection in minirhizotron images.

3.1 Marked Gibbs Process Model

Our goal in this work is to extract the roots in an image, which we model as non-overlapping connected segments. Under these considerations, a natural choice for simulating the interaction energy between random segments becomes the marked Gibbs point process model [59] [60] (Figure 3.1). Here a segment is given by $s_i = (\omega_i, m_i)$, where $\omega_i = (x_i, y_i)$ are the coordinates of its center, and $m_i = (\ell_i, \theta_i)$ contains the length and orientation of the segment. The line segment set $\mathbf{s} = \{s_1, \ldots, s_n\}$ that we wish to extract is considered as the realization of a point process on $K \times M$ where $\omega_i \in K$ and $m_i \in M$ for all $i = 1, ..., n$ and $K = [0, X_{max}] \times [0, Y_{max}]$, $M = [L_{min}, L_{max}] \times [0, \pi)$. Within the framework of a Gibbs point process, the probability density of our model is $p(\mathbf{s}) \propto \exp(-U(\mathbf{s}))$, where

$$U(\mathbf{s}) = U_D(\mathbf{s}) + U_I(\mathbf{s}), \tag{3.1}$$

and where the two terms represent the *intensity* (or data) model and the *interaction* model, respectively.[1] The network estimate is obtained by minimizing the energy functional $U(\mathbf{s})$:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \{U_D(\mathbf{s}) + U_I(\mathbf{s})\}.$$

3.2 Intensity Model

The graylevel profile of the cross section of a young root approximates a Gaussian curve [67], with the peak of the Gaussian in the center of the root (Figure 3.2). Based on this observation, the intensity model seeks to fit each segment to nearby

[1]Note that the term *intensity* in this context does not directly refer to image intensity but rather to the intensity of the point process.

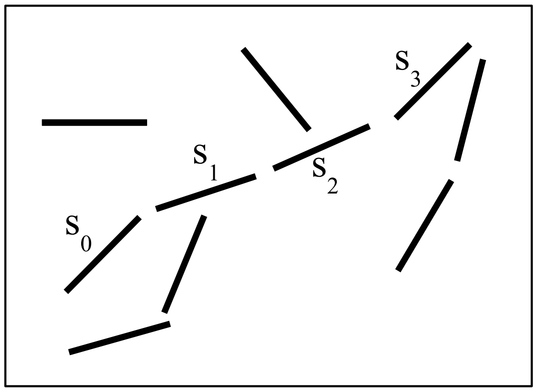


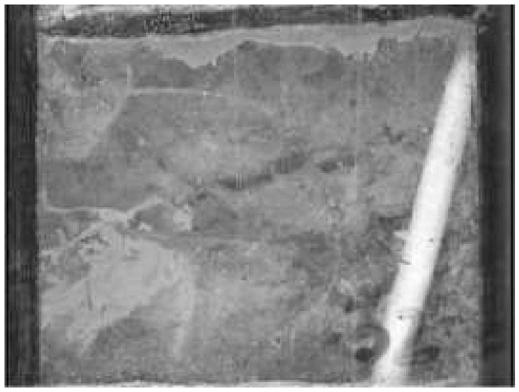
Figure 3.1: The proposed energy model extracts connected segments (S_0, S_1, S_2, S_3) of a linear feature by minimizing the interaction energy between interacting segments pairs in the segments network.

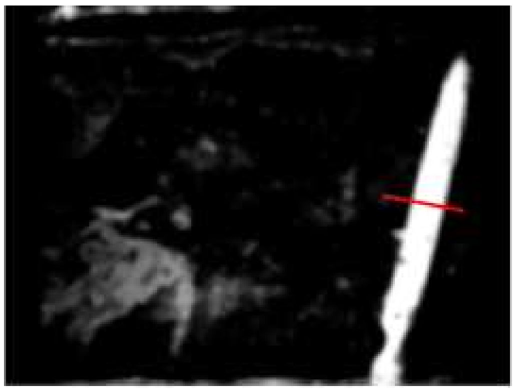
local maxima of the image. Let $\mathcal{X}$ be the set of pixels in the image which are local maxima, and let $\mathcal{X}_i \subseteq \mathcal{X}$ be the set of local maxima that lie within a rectangle of fixed width centered and aligned with a line segment s_i. We say that $\mathcal{X}_i$ contains the maxima that *lend support* to s_i. Let us define the *spread* of $\mathcal{X}_i$ as the shortest path length connecting the points divided by the number of points $\mid \mathcal{X}_i \mid$. The spread of a line segment is equivalent to the average distance between neighboring points in the set, with neighbors defined by first ordering the points according to their projection onto the segment. Define the *residue* of $\mathcal{X}_i$ as the mean squared error of the points with respect to their distance from s_i.

With these definitions, we formulate the intensity energy of a segment as a linear combination of various pieces of evidence:

$$U_D(s_i) = \sum_{j=1}^{N_D} w_j g_j(s_i), \tag{3.2}$$

where $g_1(s_i)$, $g_2(s_i)$, and $g_3(s_i)$ are respectively the spread, residue, and support of s_i, as defined above. A fourth property is based on the observation that the width of a root tends to be fairly constant. We define $g_4(s_i)$ as the *width stability*, or the





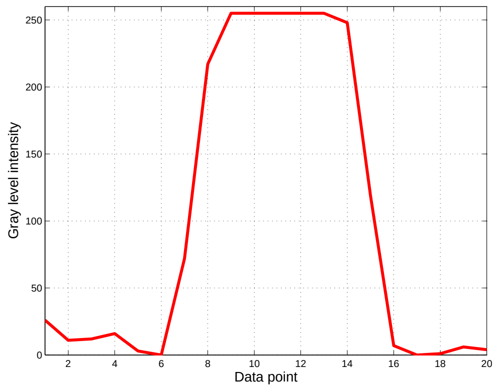


Figure 3.2: The graylevel profile of the cross section of a young root approximates a Gaussian curve, with the peak of the Gaussian in the center of the root. From *top* to *bottom*: A minirhizotron image containing one root, the preprocessed version, a plot of the intensity profiles of the root cross sections in the preprocessed image along the line shown.

percentage of pixels whose width (perpendicular distance to the nearest intensity edge) is within some tolerance of the estimated root width.

As mentioned earlier, the graylevel intensity profile of the cross section of a young root can be approximated as a Gaussian curve. We use the matching of the graylevel intensity profile and a Gaussian curve along the root's centerline to study the intensity distribution pattern of a root. Instead of matching a single intensity profile of a root's cross section each time, we match a number of cross sections of identical profiles simultaneously by building a two-dimensional matched filter according to the root's orientation and width. After applying the matched filter to all the points in $\mathcal{X}_i$, we estimate the fifth property *smoothness* (g_5) of a centerline segment as the standard deviation of measured matched filter response (MFR) values. Details of matched filter and its application are introduced in Appendix A.1. Thus we set $N_D = 5$.

The total intensity energy is obtained by assuming linear independence between the segments: $U_D(\mathbf{s}) = \sum_{i=1}^{n} U_D(s_i)$, and the weights w_j are learned from the data, as explained later.

3.3 Interaction Model

Stoica et al. [59] [60] constructed a Candy model to represent the interaction between segments [59] [60]. Let $p_i = \omega_i - \frac{\ell}{2}(\cos\theta_i, \sin\theta_i)$ and $q_i = \omega_i + \frac{\ell}{2}(\cos\theta_i, \sin\theta_i)$ be the two endpoints of s_i. They define the *attraction region* $\mathcal{A}_i$ of s_i as the set of points such that $\omega \in \mathcal{A}_i$ if and only if $|| \omega_i - p_i ||< \tau_i$ or $|| \omega_i - q_i ||< \tau_i$. Two segments s_i and s_j are considered to have an attraction interaction with one another if $p_i \in \mathcal{A}_j$ or $q_i \in \mathcal{A}_j$ or $p_j \in \mathcal{A}_i$ or $q_j \in \mathcal{A}_i$.

To penalize segments that overlap, they defined a rejection $\mathcal{R}_i$ region around each segment s_i which is a circle centered at ω_i with a radius r_i $(= \ell_i/2)$. Two segments s_i and s_j are considered to have a rejection interaction with one another if $(x_i, y_i) \in \mathcal{R}_j$ or $(x_j, y_j) \in \mathcal{R}_i$. Segment pairs with rejection interaction cannot be connected.

Because our algorithm extracts segments from an image at different scales, there are lots of overlapping between segment pairs from the same linear feature. The definition of the rejection interaction will keep them as separate segments in the network. To remedy this problem, our interaction model adopt the attraction interaction only (See Figure 3.3). Considering some linear features are shorter or may be partially occluded by background noise, we choose $\tau_i = \ell_i/3$ instead of $\ell_i/4$ [59] [60].

Two segments s_i and s_j are said to be connected if only one end point of a segment is in the attraction region of the other segment With respect to this definition, segments can be connected at both of its end points, either of its end points, or they can be unconnected. The interaction function penalizes segments that are connected but not well aligned:

$$U_I(s_i, s_j) = \begin{cases} \sum_{j=1}^{N_I} w'_j h_j(s_i, s_j) & \text{if } s_i \text{ and } s_j \text{ are connected} \\ u_c & \text{otherwise} \end{cases} \tag{3.3}$$

where the *proximity* $h_1(s_i, s_j) = \min\{\| p_i - p_j \|, \| p_i - q_j \|, \| q_i - p_j \|, \| q_i - q_j \|\}$ measures the minimum Euclidean distance between the end points on two segments, the *alignment* $h_2(s_i, s_j) = | (\pi - \theta_{ij})/\pi |$, where θ_{ij} is the angle between the two segments and measures their curvature, and u_c is a constant.

As with the intensity model, the weights w' are learned from the data, and independence among the segment pairs is assumed: $U_I(\mathbf{s}) = \sum_{s_i \diamond s_j, i<j} U_I(s_i, s_j)$, where $s_i \diamond s_j$ means that segments s_i and s_j are interacting segments.

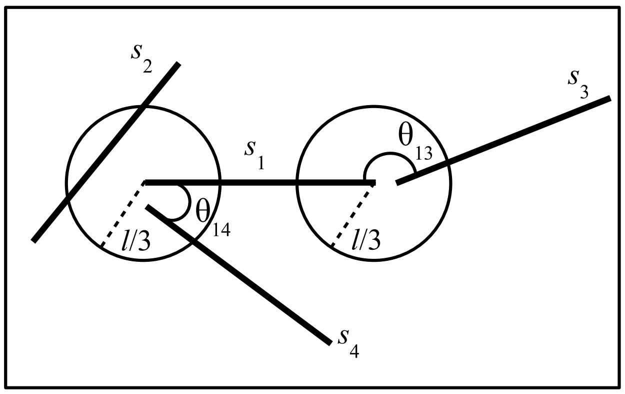


Figure 3.3: A segment s_1 and its attraction region. Segments s_3 and s_4 interact with s_1, while s_2 does not.

Chapter 4

Linear Discriminant Analysis

As we will see in the next chapter, several places in our algorithm involve making a binary decision in a vector space. For example, to detect seed points we apply a linear classifier to the pixels to determine whether they are more similar to the positive (pixels on a root) or negative (pixels on the background) examples that were manually labeled in an off-line training step. Thus, in our research we need to distinguish between two linearly separable classes of patterns. In this chapter we describe two common approaches to this problem and compare them, as background material for our algorithm.

Let $\mathbf{x}^{(n)}$, $n = 1, \ldots, N$ be a set of N patterns in a m-dimensional space, where $m = 2$. Each pattern has an associated binary value $\phi(\mathbf{x}^{(n)})$ that indicates to which class the pattern belongs. Letting f be the active function (e.g., step or sign function), we want to find a vector $\bar{\mathbf{w}}$ such that

$$V_n = f(\bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(n)}) \qquad n = 1, \ldots, N, \tag{4.1}$$

approximates $\phi(\mathbf{x}^{(n)})$ in the sense that the quantity $\sum_{n=1,\ldots,N} [V_n \neq \phi(\mathbf{x}^{(n)})]$ is minimized. The vector $\bar{\mathbf{x}}^{(n)} = [\,\mathbf{x}^{(n)} \quad 1\,]^T$ consists of the input pattern $x^{(n)}$ of dimension d plus an extra component equal to one, and $\bar{\mathbf{w}} = [\,\mathbf{w} \quad b\,]^T$ consists of the weight vector $\mathbf{w}$ augmented by the threshold b [46]. The vector $\bar{\mathbf{w}}$ is represented by a point in the $(N+1)$-dimensional weight space, and it defines a hyperplane that divides the weight space into two subspaces. For a given $\bar{\mathbf{w}}$, each pattern $\mathbf{x}^{(n)}$ is classified as c_1 if the quantity $V_n = \phi(\mathbf{x}^{(n)})$, or c_2 if $V_n \neq \phi(\mathbf{x}^{(n)})$.

Two major approaches are considered here. One approach determines the optimal separating hyperplane by maximizing the between-class variance relative to the within-class variance. The other approach determines the separating hyperplane by minimizing a measurement of overlap in the training data by using a logistic regression model.

4.1 Fisher's Linear Discriminant

The Fisher approach [20] is based on a projection of m-dimensional data onto a line. The intention is that these projections onto the decision line will be well separated by their class. Thus, the line is oriented to maximize this class separation (see Figure 4.1).

Assume we have a set of N m-dimensional samples $x^{(1)}, x^{(2)}, ..., x^{(N)}$, of which N_1 belong to class c_1, and N_2 belong to class c_2. Note that $N_1 + N_2 = N$. We seek to obtain a scalar y by projecting the samples x onto a line

$$y = \mathbf{w}^T\mathbf{x} \tag{4.2}$$

Of all the possible lines we would like to select the one that maximizes the separability of the projections. In order to find a good projection vector, we need to define a

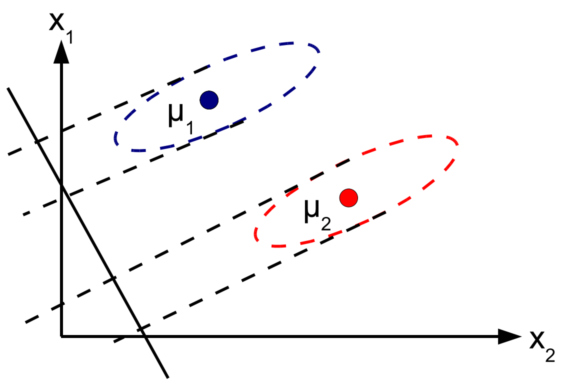


Figure 4.1: The $m = 2$ example of the Fisher's discriminant on a set of samples ($c = 2$). The mean vector of the two classes are marked as *red* and *blue* respectively.

measure of separation between the projections. The mean vector of each class is given by

$$\mu_i = \frac{1}{N_i} \sum_{\mathbf{x} \in c_i} \mathbf{x} \tag{4.3}$$

$$\tilde{\mu_i} = \frac{1}{N_i} \sum_{y \in c_i} y = \frac{1}{N_i} \sum_{\mathbf{x} \in c_i} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mu_i, \tag{4.4}$$

where $i = 1$ for class c_1, and $i = 2$ for class c_2. The solution proposed by Fisher is to maximize a function that represents the difference between the means, normalized by a measure of the within-class scatter. For each class we define the scatter, an equivalent of the variance, as

$$\tilde{s_i}^2 = \sum_{y \in c_i} (y - \tilde{\mu_i})^2. \tag{4.5}$$

The Fisher linear discriminant maximizes the criterion function

$$J(\mathbf{w}) = \frac{(\tilde{\mu_1} - \tilde{\mu_2})^2}{\tilde{s_1}^2 + \tilde{s_2}^2}. \tag{4.6}$$

Therefore, we will be looking for a projection where examples from the same class are projected very close to each other, meanwhile the examples from different classes are projected as far apart as possible.

In order to find the optimum projection $\mathbf{w}^*$, we need to express $J(\mathbf{w})$ as an explicit function of $\mathbf{w}$. We define a measure of the scatter in multivariate feature space $\mathbf{x}$, which are scatter matrices

$$S_i = \sum_{x \in c_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T \tag{4.7}$$

$$S_1 + S_2 = S_{\mathrm{W}} \tag{4.8}$$

where S_{W} is called the within class scatter matrix.

The scatter of the projection y can then be expressed as a function of the scatter matrix in feature space $\mathbf{x}$

$$\tilde{s_i}^2 = \sum_{y \in c_i} (y - \tilde{\mu_i})^2 = \sum_{\mathbf{X} \in c_i} (\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mu_i)^2 = \sum_{\mathbf{X} \in c_i} \mathbf{w}^T (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T \mathbf{w} = \mathbf{w}^T S_i \mathbf{w} \tag{4.9}$$

and

$$\tilde{s_1}^2 + \tilde{s_2}^2 = \mathbf{w}^T S_{\mathrm{W}} \mathbf{w}. \tag{4.10}$$

Similarly, the difference between the projected means can be expressed in terms of the means in the original feature space

$$(\tilde{\mu_1} - \tilde{\mu_2})^2 = (\mathbf{w}^T \mu_1 - \mathbf{w}^T \mu_2)^2 = \mathbf{w}^T (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \mathbf{w} = \mathbf{w}^T S_B \mathbf{w}, \tag{4.11}$$

where the matrix $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ is called the between-class scatter matrix.

We can finally express the Fisher criterion in terms of S_{W} and S_B as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}} \tag{4.12}$$

Taken together, we wish to maximize the criterion function. After differentiating $J(\mathrm{w})$ and equating to zero,

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{S_B \mathbf{w}(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}) - S_{\mathrm{W}} \mathbf{w}(\mathbf{w}^T S_B \mathbf{w})}{(\mathbf{w}^T S_{\mathrm{W}} \mathbf{w})^2} = 0, \tag{4.13}$$

from which we acquire $S_{\mathrm{W}}^{-1} S_B \mathbf{w} - J\mathbf{w} = 0$.

Solving the generalized eigenvalue problem ($S_{\mathrm{W}}^{-1} S_B \mathbf{w} = J\mathbf{w}$) yields Fisher's Linear Discriminant:

$$\mathbf{w}^* = \arg\max \left\{ \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_{\mathrm{W}} \mathbf{w}} \right\} = S_W^{-1}(\mu_1 - \mu_2), \tag{4.14}$$

which arises from the fact that $S_B \mathbf{w}$ is proportional to $\mu_1 - \mu_2$, and the scale of the vector does not matter since it is only used to define the normal to the separating hyperplane.

A simple LDA example (Figure 4.1) is shown below by computing the Linear Discriminant projection for the following two-dimensional dataset.

1. $X_1 = (x_1,\, x_2) = \{(4,1),\ \ (2,4),\ \ (2,3),\ \ (3,6),\ \ (4,4)\}$
2. $X_2 = (x_1,\, x_2) = \{(9,10),\ \ (6,8),\ \ (9,5),\ \ (8,7),\ \ (10,8)\}$

Solution:

The class statistics are:

$$S_1 = \begin{bmatrix} 0.80 & -0.40 \\ -0.40 & 2.60 \end{bmatrix};\ S_2 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix};$$

$$\mu_1 = \begin{bmatrix} 3.00 & 3.60 \end{bmatrix};\ \mu_2 = \begin{bmatrix} 8.40 & 7.60 \end{bmatrix};$$

The within-class and between-class scatter matrices are

$$S_B = \begin{bmatrix} 29.16 & 21.60 \\ 21.60 & 16.00 \end{bmatrix};\ S_W = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix};$$

The LDA projection is then obtained as the solution of the generalized eigenvalue problem

$$S_W^{-1} S_B \mathbf{v} = \lambda\,\mathbf{v} \Rightarrow \mid S_W^{-1} S_B - \lambda\, I \mid = 0 \Rightarrow \begin{bmatrix} 11.89 - \lambda & 8.81 \\ 5.08 & 3.76 - \lambda \end{bmatrix} = 0 \Rightarrow \lambda = 15.65;$$

$$\begin{bmatrix} 11.89 & 8.81 \\ 5.08 & 3.76 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 15.65 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0.91 \\ 0.39 \end{bmatrix}$$

Or directly by

$$\mathbf{w}^* = S_{\mathrm{W}}^{-1}(\mu_1 - \mu_2) = \begin{bmatrix} -0.91 - 0.39 \end{bmatrix}^T.$$

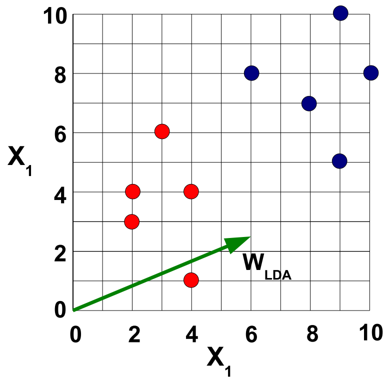


Figure 4.2: The result of applying Fisher's approach on a set of samples ($p = 2$, $c = 2$). The LDA projection vector $W_{LDA} = [-0.91\ -0.39]^T$.

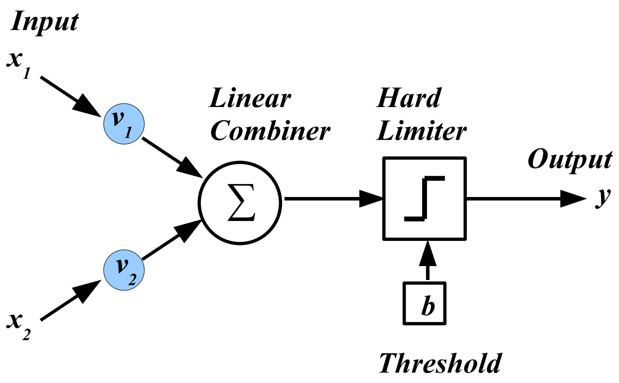


Figure 4.3: The McCulloch and Pitts neuron model. The model consists of a linear combiner followed by a hard limiter. The weighted sum of the inputs is applied to the hard limiter.

4.2 Perceptron Learning

The perceptron is a kind of single-layer artificial network with only one neuron that is typically used for classification. The operation of Rosenblatt's perceptron [52] is based on the McCulloch and Pitts neuron model [36], which consists of a linear combiner followed by a hard limiter. The weighted sum of the m-dimensional inputs is applied to the hard limiter (Step or sign function) (Figure 4.3).

The neuron computes the weighted sum of the input signals and compares the result with a threshold value, b. If the net input is less than the threshold, the neuron

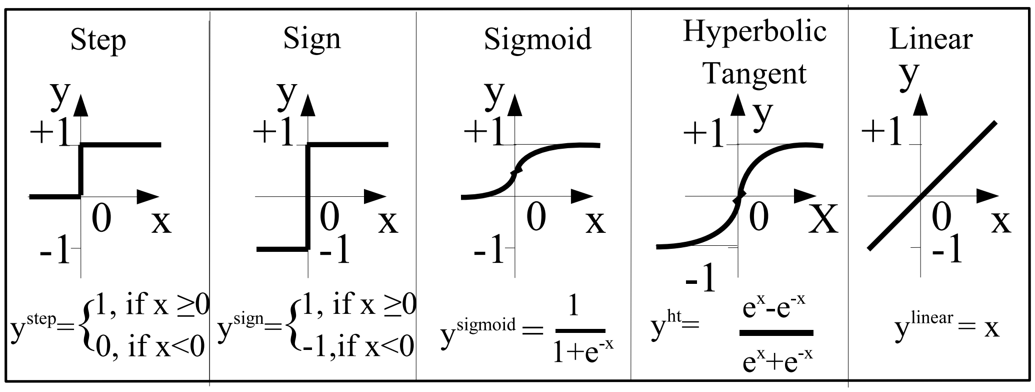

Step
Sign
Sigmoid
Hyperbolic Tangent
Linear
y
+1
0
x
-1
y^step = {1, if x ≥0; 0, if x<0
y^sign = {1, if x ≥0; -1,if x<0
y^sigmoid = 1/(1+e^-x)
y^ht = (e^x-e^-x)/(e^x+e^-x)
y^linear = x

Figure 4.4: Possible activation functions. From *left* to *right* are: Step function, sign function, sigmoid function, hyperbolic tangent function and linear function.

output is -1. But if the net input is greater than or equal to the threshold, the neuron becomes activated and its output attains a value +1. The neuron uses the following transfer or activation function:

$$X = \sum_{i=1}^{m} x_i v_i \tag{4.15}$$

$$y = \operatorname{sgn}(X - b) = \begin{cases} 1, & \text{if } X \geq b \\ -1, & \text{otherwise} \end{cases} \tag{4.16}$$

This type of activation function is called a *sign function*. Some other types of activation functions are shown in Figure 4.4.

Assume we have a set of N m-dimensional samples $x^{(1)}, x^{(2)}, ..., x^{(n)}$, of which N_1 belong to class c_1, and N_2 belong to class c_2, the perceptron learning algorithm works as follows:

(1) Initialize the weight vector $\mathbf{v}$ and bias b to small random numbers.

(2) Activate the perceptron by applying inputs $x_1^{(n)}$, $x_2^{(n)}$,..., $x_i^{(n)}$ and desired output

$y(n)$. Calculate the actual output at iteration $n = 1$,

$$y(n) = \text{sgn}\left[\sum_{i=1}^{m} x_i^{(n)} v_i(n) - b\right]$$

where m is the dimension of the perceptron inputs and sgn is the sign function.

(3) Update the weights according to:

$$v_i(n+1) = v_i(n) + \eta(d(n) - y(n))x_i(n) \tag{4.17}$$

where d is the desired output, n is the iteration number, and η is the learning rate, where $0 \leq \eta \leq 1.0$.

(3) Repeat Step 2 and Step 3 until the stopping criteria is met, for example the iteration error is less than a user-specified error threshold, or a predetermined number of iterations have been completed.

According to the convergence theorem, if the input d-dimensional input data $\mathbf{x}$ is linearly separable, then optimal weights $\mathbf{v}^*$ can be found in a finite number of steps using the perceptron learning algorithm [51]. The hyperplane is defined by the function,

$$\sum_{i=1}^{m} x_i v_i - b = 0 \tag{4.18}$$

We now show a simple example of applying the perceptron learning algorithm to the following three-dimensional dataset:

$$\left\{x_1^{(n)}, x_m^{(n)};\ \phi(\mathbf{x}^{(n)})\right\} = \{(0,0;\ 0), (0,1;\ 1), (1,0;\ 1), (1,1;\ 1)\}, \tag{4.19}$$

where $m = 2$ and $n = 4$. The weight vector $\mathbf{v}$ is initialized as $[v_1 \quad v_2] = [0.1 \quad 0.3]$.

The threshold is $b = 0.5$, the learning rate $\eta = 0.2$. The optimal weights are found after four steps of learning.

Step.1

$$\begin{aligned}
y(1) &= sgn([0\ \ 0] \cdot [0.1\ \ 0.3] - 0.5) = sgn(-0.2) = 0 \\
\mathbf{v}(2) &= \mathbf{v}(1) + \eta \cdot (d(1) - y(1)) \cdot x(1) \\
&= [0.1\ \ 0.3] + 0.2 \cdot 0 \cdot [0\ \ 0] \\
&= [0.1\ \ 0.3] \\
y(2) &= sgn([0\ \ 1] \cdot [0.1\ \ 0.3] \quad - 0.5) = sgn(-0.2) = 0 \\
\mathbf{v}(3) &= \mathbf{v}(2) + \eta \cdot (d(2) - y(2)) \cdot x(2) \\
&= [0.1\ \ 0.3] + 0.2 \cdot 1 \cdot [0\ \ 1] \\
&= [0.1\ \ 0.5] \\
y(3) &= sgn([1\ \ 0] \cdot [0.1\ \ 0.5] \quad - 0.5) = sgn(-0.4) = 0 \\
\mathbf{v}(4) &= \mathbf{v}(3) + \eta \cdot (d(3) - y(3)) \cdot x(3) \\
&= [0.1\ \ 0.5] + 0.2 \cdot 1 \cdot [1\ \ 0] \\
&= [0.3\ \ 0.5] \\
y(4) &= sgn([1\ \ 1] \cdot [0.3\ \ 0.5] \quad - 0.5)] = sgn(0.3) = 1 \\
\mathbf{v}(5) &= \mathbf{v}(4) + \eta \cdot (d(4) - y(4)) \cdot x(4) \\
&= [0.3\ \ 0.5] + 0.2 \cdot 0 \cdot [1\ \ 1] \\
&= [0.3\ \ 0.5] \\
\textstyle\sum_{i=1}^{4} error_1 &= 2.
\end{aligned}$$

Step.2

$$\begin{aligned}
y(5) &= sgn([0 \;\; 0] \cdot [0.3 \;\; 0.5] - 0.5) = sgn(-0.5) = 0 \\
\mathbf{v}(6) &= \mathbf{v}(5) + \eta \cdot (d(5) - y(5)) \cdot x(1) \\
&= [0.3 \;\; 0.5] + 0.2 \cdot 0 \cdot [0 \;\; 0] \\
&= [0.3 \;\; 0.5] \\
y(6) &= sgn([0 \;\; 1] \cdot [0.3 \;\; 0.5] - 0.5) = sgn(0) = 0 \\
\mathbf{v}(7) &= \mathbf{v}(6) + \eta \cdot (d(6) - y(6)) \cdot x(2) \\
&= [0.3 \;\; 0.5] + 0.2 \cdot 1 \cdot [0 \;\; 1] \\
&= [0.3 \;\; 0.7] \\
y(7) &= sgn([1 \;\; 0] \cdot [0.3 \;\; 0.7] - 0.5) = sgn(-0.2) = 0 \\
\mathbf{v}(8) &= \mathbf{v}(7) + \eta \cdot (d(7) - y(7)) \cdot x(3) \\
&= [0.3 \;\; 0.7] + 0.2 \cdot 1 \cdot [1 \;\; 0] \\
&= [0.5 \;\; 0.7] \\
y(8) &= sgn([1 \;\; 1] \cdot [0.5 \;\; 0.7] - 0.5) = sgn(0.7) = 1 \\
\mathbf{v}(9) &= \mathbf{v}(8) + \eta \cdot (d(8) - y(8)) \cdot x(4) \\
&= [0.5 \;\; 0.7] + 0.2 \cdot 0 \cdot [1 \;\; 1] \\
&= [0.5 \;\; 0.7] \\
\textstyle\sum_{i=1}^{4} error_2 &= 2.
\end{aligned}$$

Step.3

$$\begin{aligned}
y(9) &= sgn([0\ \ 0]\cdot[0.5\ \ 0.7]-0.5) = sgn(-0.5) = 0\\
\mathbf{v}(10) &= \mathbf{v}(9)+\eta\cdot(d(9)-y(9))\cdot x(1)\\
&= [0.5\ \ 0.7]+0.2\cdot 0\cdot[0\ \ 0]\\
&= [0.5\ \ 0.7]\\
y(10) &= sgn([0\ \ 1]\cdot[0.5\ \ 0.7]-0.5) = sgn(0.2) = 1\\
\mathbf{v}(11) &= \mathbf{v}(10)+\eta\cdot(d(10)-y(10))\cdot x(2)\\
&= [0.5\ \ 0.7]+0.2\cdot 0\cdot[0\ \ 1]\\
&= [0.5\ \ 0.7]\\
y(11) &= sgn([1\ \ 0]\cdot[0.5\ \ 0.7]-0.5) = sgn(0) = 0\\
\mathbf{v}(12) &= \mathbf{v}(11)+\eta\cdot(d(11)-y(11))\cdot x(3)\\
&= [0.5\ \ 0.7]+0.2\cdot 1\cdot[1\ \ 0]\\
&= [0.7\ \ 0.7]\\
y(12) &= sgn([1\ \ 1]\cdot[0.7\ \ 0.7]-0.5) = sgn(0.9) = 1\\
\mathbf{v}(13) &= \mathbf{v}(12)+\eta\cdot(d(12)-y(12))\cdot x(4)\\
&= [0.7\ \ 0.7]+0.2\cdot 0\cdot[1\ \ 1]\\
&= [0.7\ \ 0.7]\\
\textstyle\sum_{i=1}^{4} error_3 &= 1.
\end{aligned}$$

Step.4

$$\begin{aligned}
y(13) &= sgn([0 \;\; 0] \cdot [0.7 \;\; 0.7] - 0.5) = sgn(-0.5) = 0 \\
\mathbf{v}(14) &= \mathbf{v}(13) + \eta \cdot (d(13) - y(13)) \cdot x(1) \\
&= [0.7 \;\; 0.7] + 0.2 \cdot 0 \cdot [0 \;\; 0] \\
&= [0.7 \;\; 0.7] \\
y(14) &= sgn([0 \;\; 1] \cdot [0.7 \;\; 0.7] - 0.5) = sgn(0.2) = 1 \\
\mathbf{v}(15) &= \mathbf{v}(14) + \eta \cdot (d(14) - y(14)) \cdot x(2) \\
&= [0.7 \;\; 0.7] + 0.2 \cdot 0 \cdot [0 \;\; 1] \\
&= [0.7 \;\; 0.7] \\
y(15) &= sgn([1 \;\; 0] \cdot [0.7 \;\; 0.7] - 0.5) = sgn(0.2) = 1 \\
\mathbf{v}(16) &= \mathbf{v}(15) + \eta \cdot (d(15) - y(15)) \cdot x(3) \\
&= [0.7 \;\; 0.7] + 0.2 \cdot 0 \cdot [1 \;\; 0] \\
&= [0.7 \;\; 0.7] \\
y(16) &= sgn([1 \;\; 1] \cdot [0.7 \;\; 0.7] - 0.5) = sgn(0.9) = 1 \\
\mathbf{v}(17) &= \mathbf{v}(16) + \eta \cdot (d(16) - y(16)) \cdot x(4) \\
&= [0.7 \;\; 0.7] + 0.2 \cdot 0 \cdot [1 \;\; 1] \\
&= [0.7 \;\; 0.7] \\
\textstyle\sum_{i=1}^{4} error &= 0.
\end{aligned}$$

During training, it is often useful to measure the performance of the network as it attempts to find the optimal weight set. Another common error measure or cost function used is sum-squared error. It is computed over all of the input vector/output vector pairs in the training set and is given by the equation below:

$$E = \frac{1}{2} \sum_{i=1}^{p} || \; y^{(i)} - d^{(i)} \; ||^2 \tag{4.20}$$

Classifier	Test 1 (%)	Test 2 (%)
Fisher's method	100.0	96.0
Perceptron learning	100.0	99.0

Table 4.1: Performance of linear discrimination using Fisher's method and Perceptron learning. The two classifiers are trained using the full data set in *Test 1* and 60% data set in *Test 2*.

where p is the number of input and output vector pairs in the training set.

4.3 Comparison of Fisher's Linear Discriminant and Perceptron Learning

For linear discriminant analysis, methods based on discriminative training such as perceptron learning often yields higher accuracy than methods based on modeling the conditional density functions such as Fisher's linear discriminant. Meanwhile, it is often easier with the perceptron learning method than the Fisher's linear discriminant method to handle missing data. The performance of these two methods are compared using the following example. As we can see in Figure 4.5 and Table 4.1, using the 2-dimensional data set containing 500 points of two classes, the perceptron learning performs better than Fisher's method. When we remove 40% of the data at random, the perceptron learning trained classifier is more accurate than the Fisher's method trained classifier.

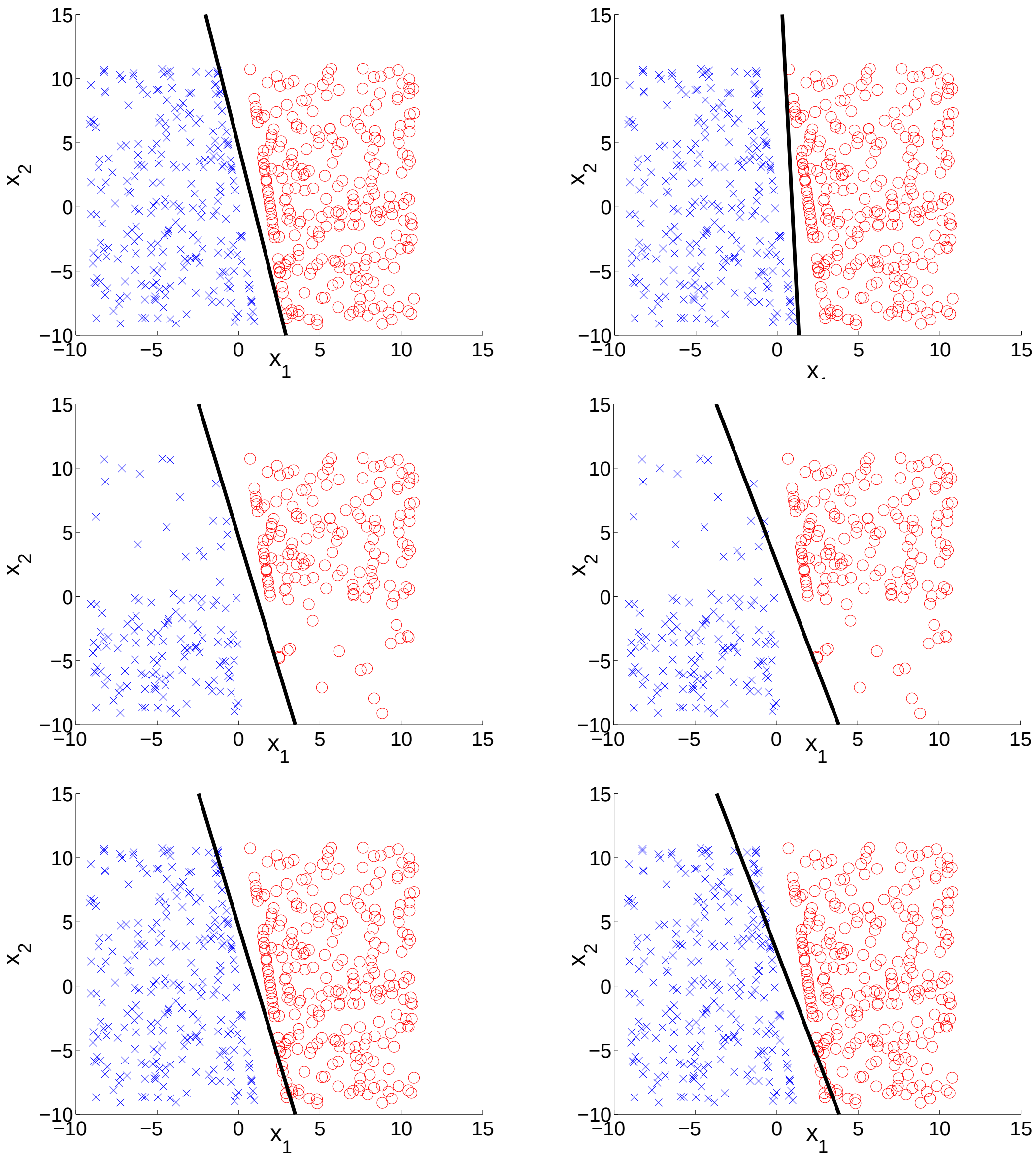


Figure 4.5: Performance comparison of perceptron learning (*left*) and Fisher's linear discriminant (*right*) on a set of samples ($p = 2$, $c = 2$). Two classes of data are marked as *circle*(*red*) and *cross*(*blue*) respectively. From *top* to *bottom* are: Performance of perceptron learning and Fisher's method using the full data set, performance of the two methods after removing 40% of the data, apply the two classifiers above to the full data set.

Chapter 5

Detection Algorithm

In Chapter 3, we proposed a probabilistic model for linear feature network using Gibbs point process. One method to simulate the finite point process is Reversible Jump Markov Chain Monte Carlo (RJMCMC) [59] [62]. However, this simulation process is very time consuming. Therefore, we developed a greedy algorithm for linear feature network extraction by minimizing the model energy function [69]. The greedy algorithm involves six steps: Selecting seed points, grouping seed points into connected line segments, validating linear features, combine oversegmented centerline, connect curved centerline and trace centerline. The object process diagram (OPD) of the algorithm is shown in Figure 5.1. Four linear discriminators are built using the perceptron algorithm. The weights used in these discriminators are learned from the training set. As mentioned earlier, we use minirhizotron root images throughout this chapter to illustrate the algorithm.

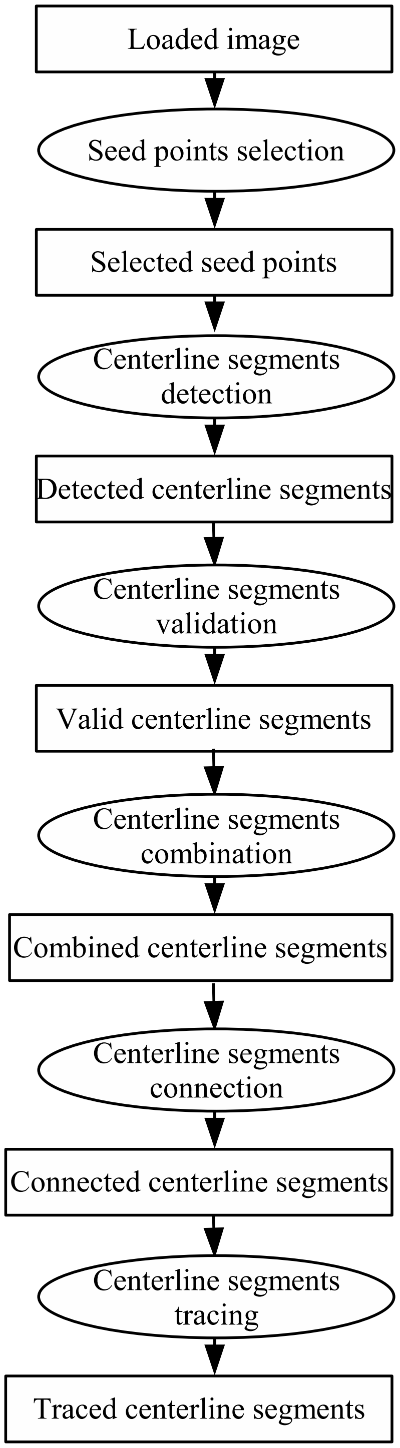

Loaded image
Seed points selection
Selected seed points
Centerline segments detection
Detected centerline segments
Centerline segments validation
Valid centerline segments
Centerline segments combination
Combined centerline segments
Centerline segments connection
Connected centerline segments
Centerline segments tracing
Traced centerline segments

Figure 5.1: The object process diagram of the greedy algorithm.

5.1 Seed Point Selection

The first step of the algorithm is to search for local maxima in the smoothed image intensity function, which is obtained by convolving with a lowpass Gaussian filter ($\sigma = 2.0$) to reduce the effects of noise. Computation is greatly reduced by searching in 1-D, only along a fraction $1/N_0$ of the rows and columns of the original image, where N_0 is the spacing between horizontal or vertical grid lines that are processed. The resulting local maxima from this step in the computation are termed *candidate seed points*.

For each candidate seed point, we compute two properties: the height e, which is the normalized gray level intensity value, and the breadth b, which is the image distance between the two neighboring local minima on either side of the maximum. This distance is computed horizontally (vertically) for candidate seed points along the rows (columns). To distinguish true seed points from local maxima due to background noise, we apply a linear discriminator $\mathbf{u}_a = [v_1 \quad v_2 \quad v_T]^T$ to the (b, e) pair. The discriminator $\mathbf{u}_a$ is learned by applying the perceptron algorithm [52] [54]. The detail of perceptron learning is described in Chapter 4.

Points for which $[\, b \quad e \quad 1\,]\, \mathbf{u}_a \geq 0$ are retained as *seed points*, while the other points are discarded. Figure 5.3a shows the discriminator $\mathbf{u}_a$ learned by applying the perceptron algorithm to data collected from a training set of 50 minirhizotron images. The positive examples include all the pixels along a centerline, while the negative examples include the remaining pixels. The plot shows the results from all three image sizes, scaled to the original size. The true positive rate, or sensitivity, of the discriminator $\mathbf{u}_a$ on the training set is 92%. The OPD of this seed point selection step is shown in Figure 5.2.

Processing only a subset of the rows and columns achieves a significant increase

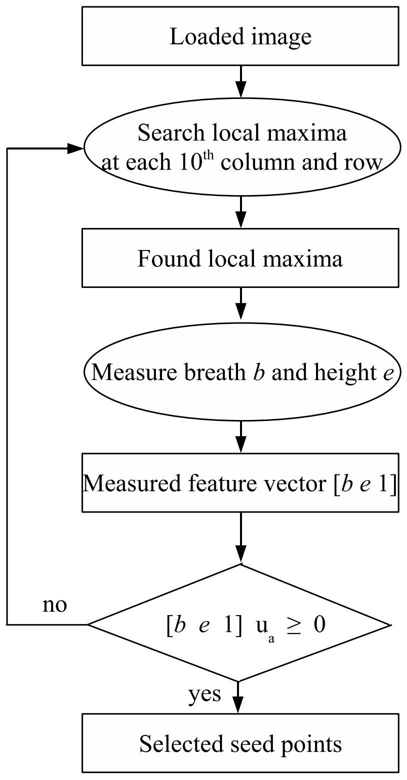

Loaded image
Search local maxima at each 10th column and row
Found local maxima
Measure breath b and height e
Measured feature vector [b e 1]
$[b\ e\ 1]\ u_a \geq 0$
no
yes
Selected seed points

Figure 5.2: The object process diagram of the seed point selection step.

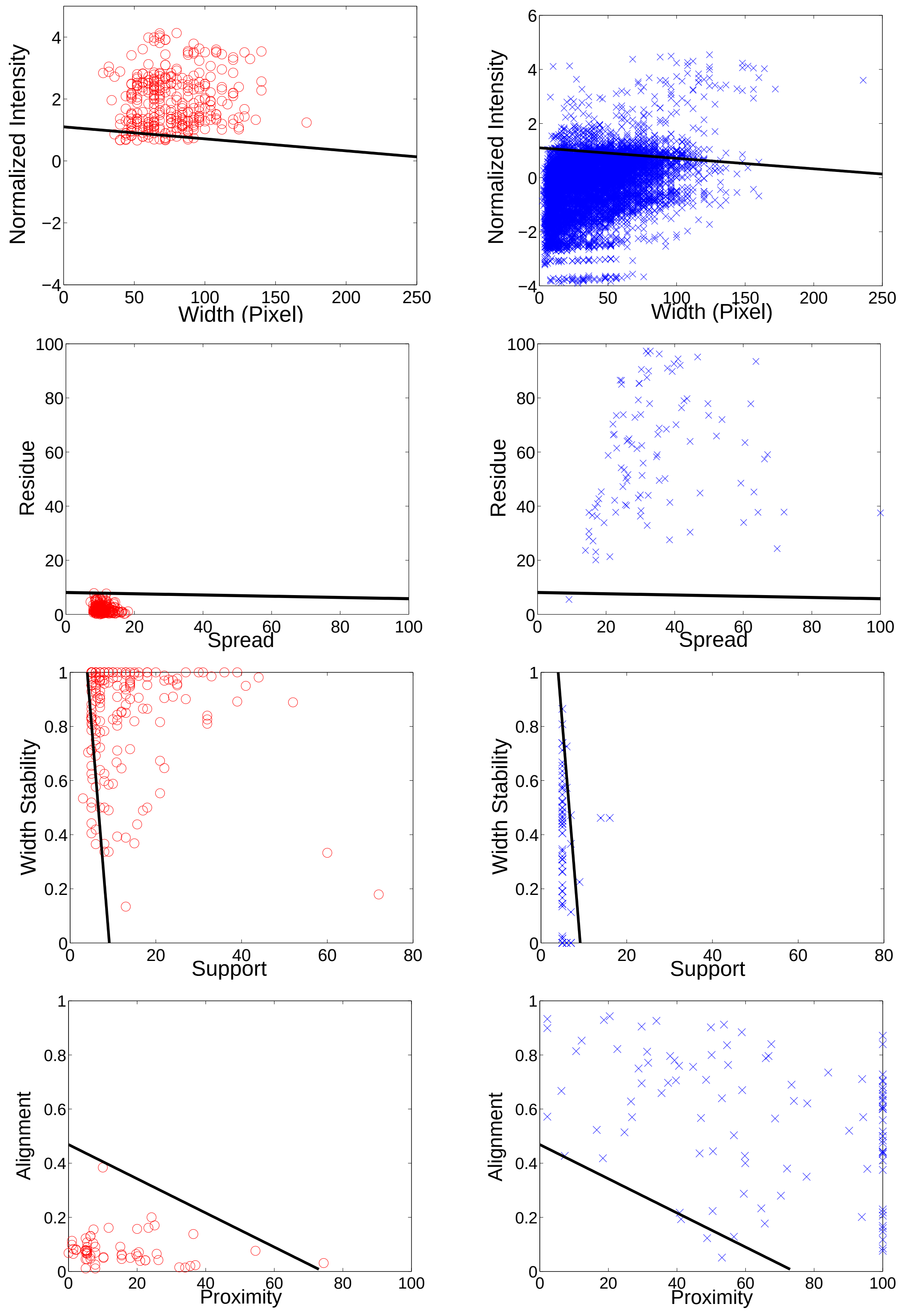


Figure 5.3: Each row shows the distribution of a pair of parameters for actual roots (red o, left) and false root-like objects (blue x, right).

in speed at the expense of missed detections. Figure 5.4 shows an inverse relationship between the *grid spacing* N_0 and the number of detected local maxima, seed points, centerline segments, and centerlines (formed by connecting the segments), as well as the computation time, on the training database. We define the performance P as the ratio of the number of detected centerlines and the computation time. As can be observed from the figure, the peak at $N_0 = 10$ exhibits a reasonable tradeoff between the computation cost and detection performance.

By processing only every tenth row and every tenth column, 90% of the data is ignored outright. After seed points have been detected, then more than 99.6% of the data is ignored in subsequent processing, thus greatly improving the running time of the algorithm. We apply the seed point detection at three separate image scales, downsampling by a factor of two in each direction for each successive scale. For the downsampled image at level k, the grid spacing used is $N_k = N_0 2^{-k}$, $k = 0, 1, 2$. Seed points that overlap another seed point at a lower resolution are discarded, where overlap is defined using the width of the local maximum. The results of this seed point selection step by rows and by columns is displayed in Figure 5.5.

5.2 Centerline Detection

Once seed points have been detected in an image, they are used to estimate the location of the linear feature centerlines. Figure 5.6 shows the primary steps involved: line segments are first fitted to seed points, then similar line segments are *combined*, finally compatible line segments are *connected*.

A region-growing procedure is adopted to group the seed points. A point is selected at random, along with its closest neighbor, and a line segment is fit to the pair. The segment is then extended and adjusted by iteratively incorporating nearby

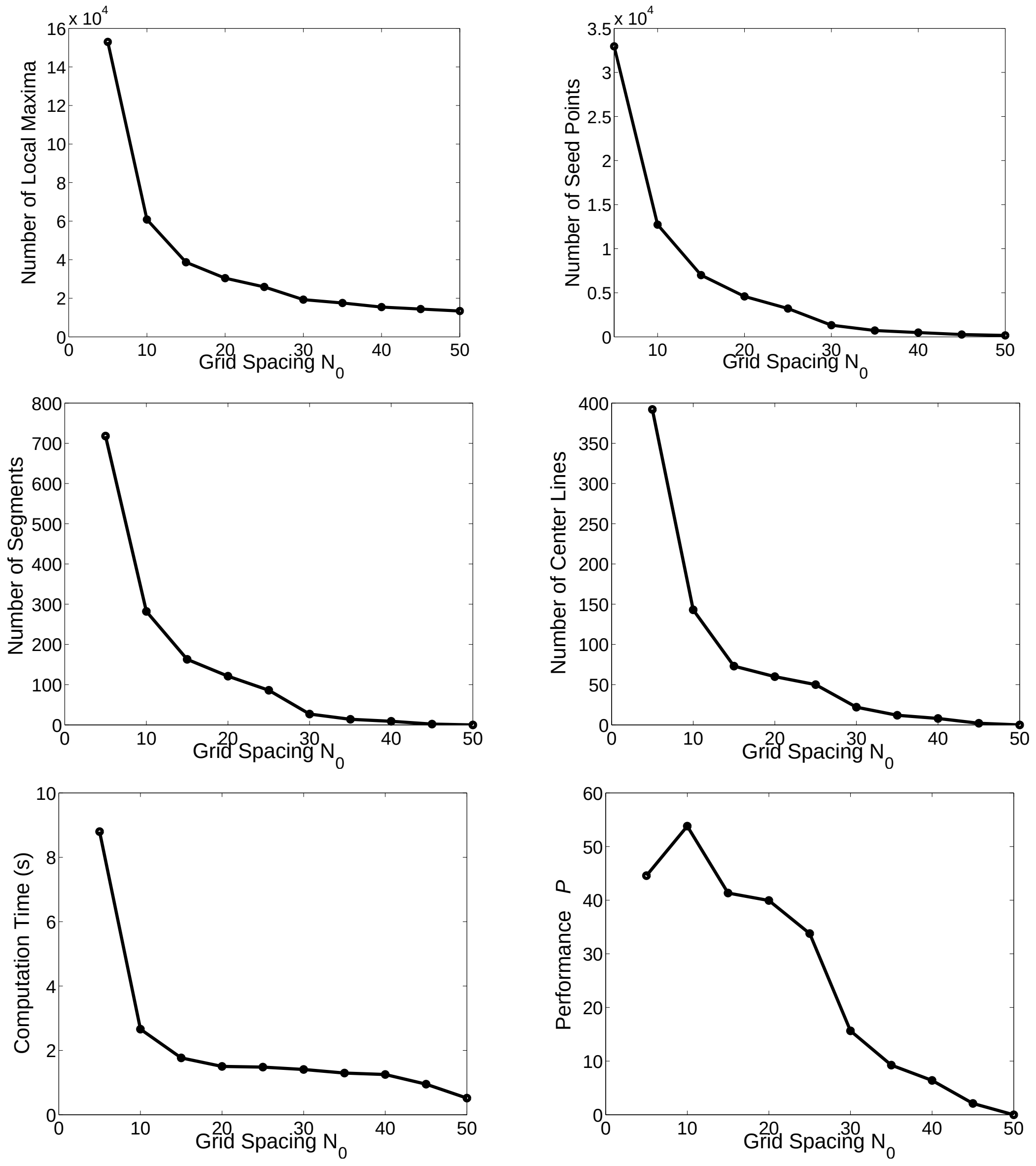


Figure 5.4: The optimal grid spacing N_o is empirically determined as a tradeoff between the number of detections and computation. The performance P, which is the ratio of the two plots to its left, peaks at $N_0 = 10$.

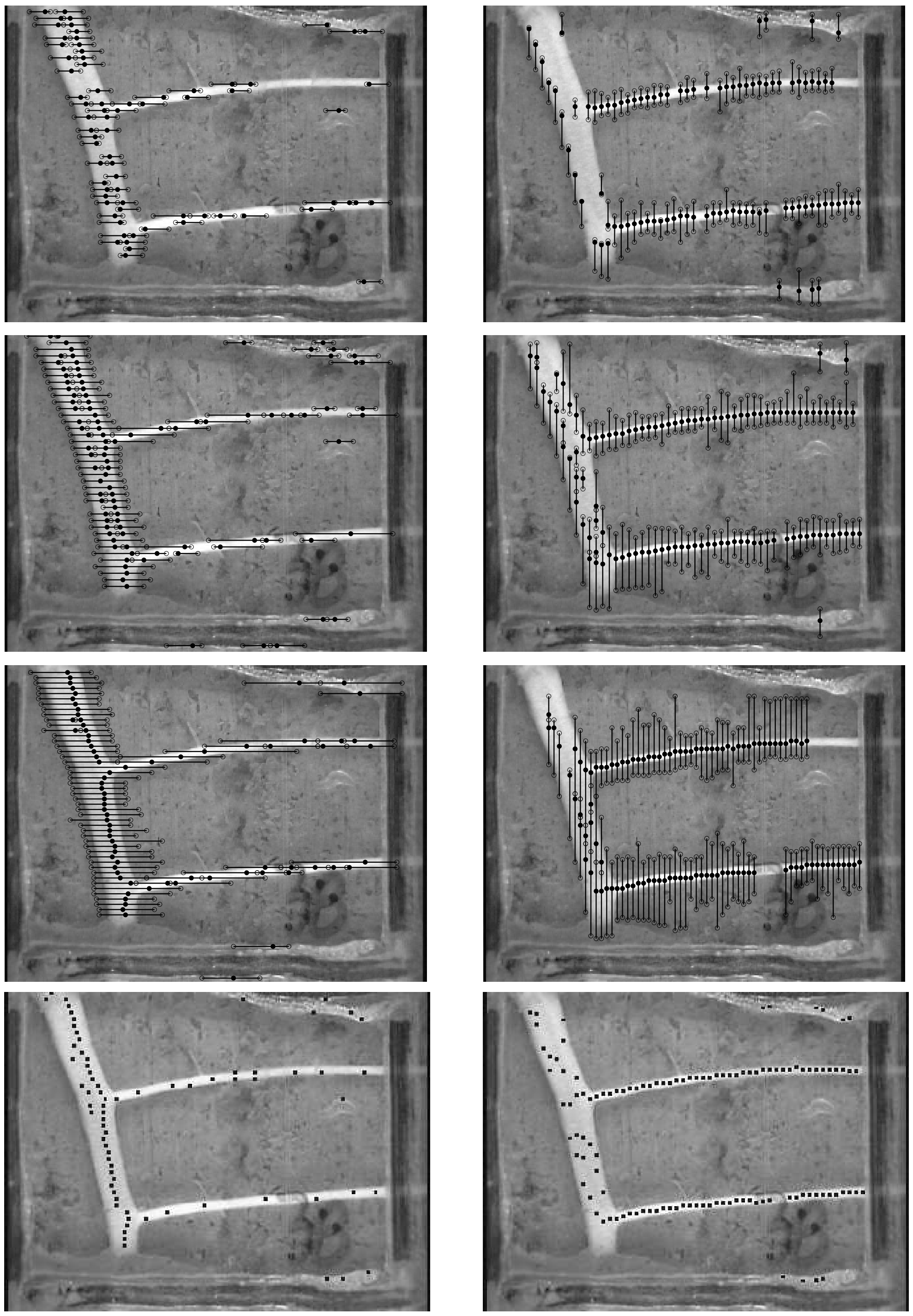


Figure 5.5: An example of seed points detected by rows (*left*) and by columns (*right*) at different scales in a sample minirhizotron root image, along with the result from combining the three scales. The dark lines are the initiated width of the seed points. From *top* to *bottom*: original size ($k = 0$), half size ($k = 1$), quarter size ($k = 2$), combined scales.

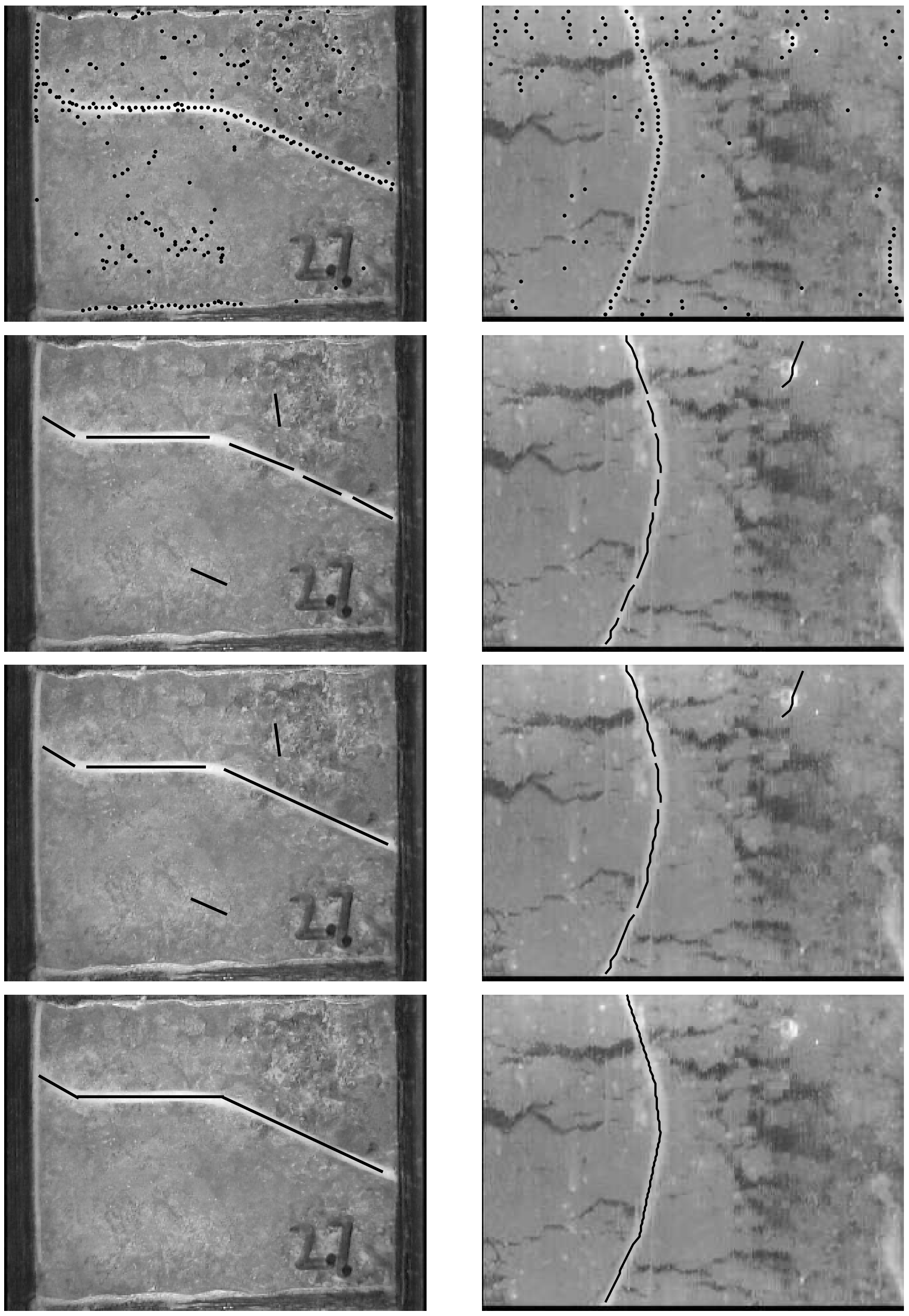


Figure 5.6: Examples of centerline detection in a *peach* root image (*left*) and a *maple* root image (*right*). *Top to bottom*: detected seed points, fitted centerline segments, combined centerline segments, and connected centerline segments. The right image also shows the removal of extraneous segments by the validation step.

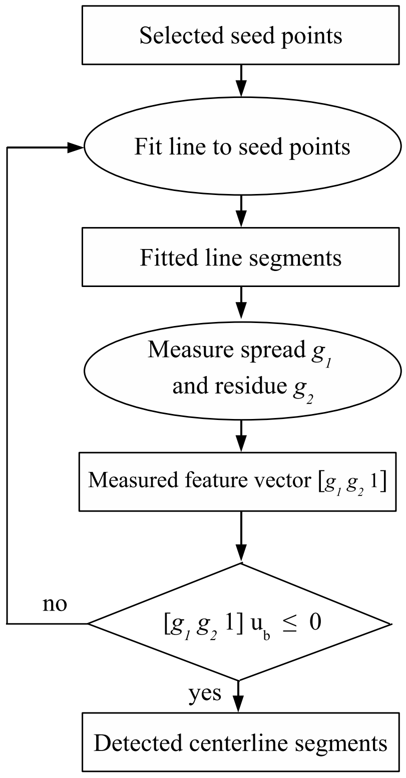

Selected seed points
Fit line to seed points
Fitted line segments
Measure spread g_1 and residue g_2
Measured feature vector $[g_1\, g_2\, 1]$
$[g_1\, g_2\, 1]\, u_b \leq 0$
no
yes
Detected centerline segments

Figure 5.7: The object process diagram of the centerline detection step.

seed points whose spread g_1 and residue g_2 are small, using the discriminator $\mathbf{u}_b$ shown in Figure 5.3b. The points belonging to the segment are removed from further consideration, and the process is repeated to the remaining points in order to detect additional segments in the same manner, terminating once all seed points have been examined. The OPD of this centerline detectio step is shown in Figure 5.7.

Because this procedure is based solely upon the coherence of the locations of local maxima in the image intensity function, it sometimes detects bright regions in the background in addition to the actual linear features. To remove these false positives, a seperate validation step is performed. For each linear feature s detected, the probability $P(D_{valid} \mid s) = \exp\{-\psi(D_{valid} \mid s)\}$ is computed, where the energy function is based on the support g_3 and width stability g_4: $\psi(D_{valid} \mid s) = w_3 g_3(s) + w_4 g_4(s)$. The weights are determined by applying a linear discriminator $\mathbf{u}_c$ to training

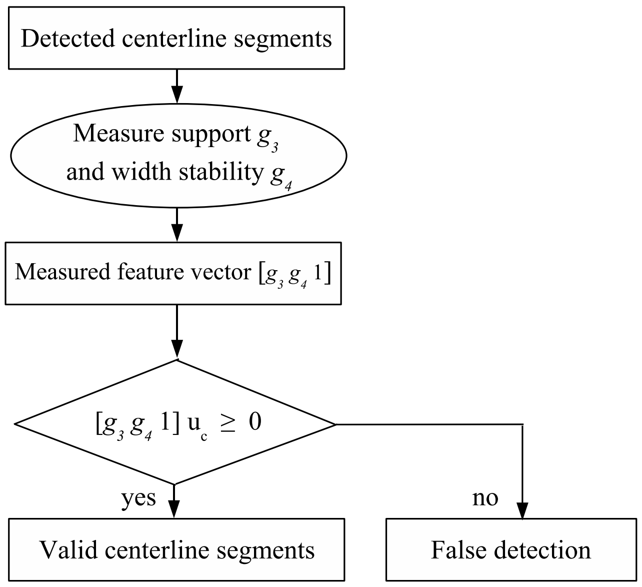

Detected centerline segments
Measure support g_3 and width stability g_4
Measured feature vector $[g_3\, g_4\, 1]$
$[g_3\, g_4\, 1]\, u_c \geq 0$
yes
no
Valid centerline segments
False detection

Figure 5.8: The object process diagram of the centerline validation step.

data, as shown in Figure 5.3c. The OPD of this centerline validation step is shown in Figure 5.8.

As introduced in Chapter 3, to measure the *width stability* g_4 of a centerline segment s, we need detect its corresponding boundary edges. In the presented algorithm, the Canny edge detector is used for edge detection. However, as shown in Figure 5.9, when applying Canny to a minirhizotron image containing roots with different width, the edges of roots smoothed by improper scale factor are sometimes ignored or very noisy. Therefore, Canny operators with different scale factors ($\sigma_1 = 1.0$ and $\sigma_2 = 2.5$) are applied for edge detection and the larger *width stability* value is chosen for linear feature validation.

Although the foregoing procedure quickly fits line segments to points, it suffers from oversegmentation. To remedy this problem, pairs of line segments are tested for *combinability.* Two line segments s_i and s_j are combined into a single segment s_c if the intensity energy of the combined segment is less: $U_D(s_c) \leq U_D(s_i) + U_D(s_j)$. In this equation, the support g_3 is the same on both sides, and we omit the width stability g_4

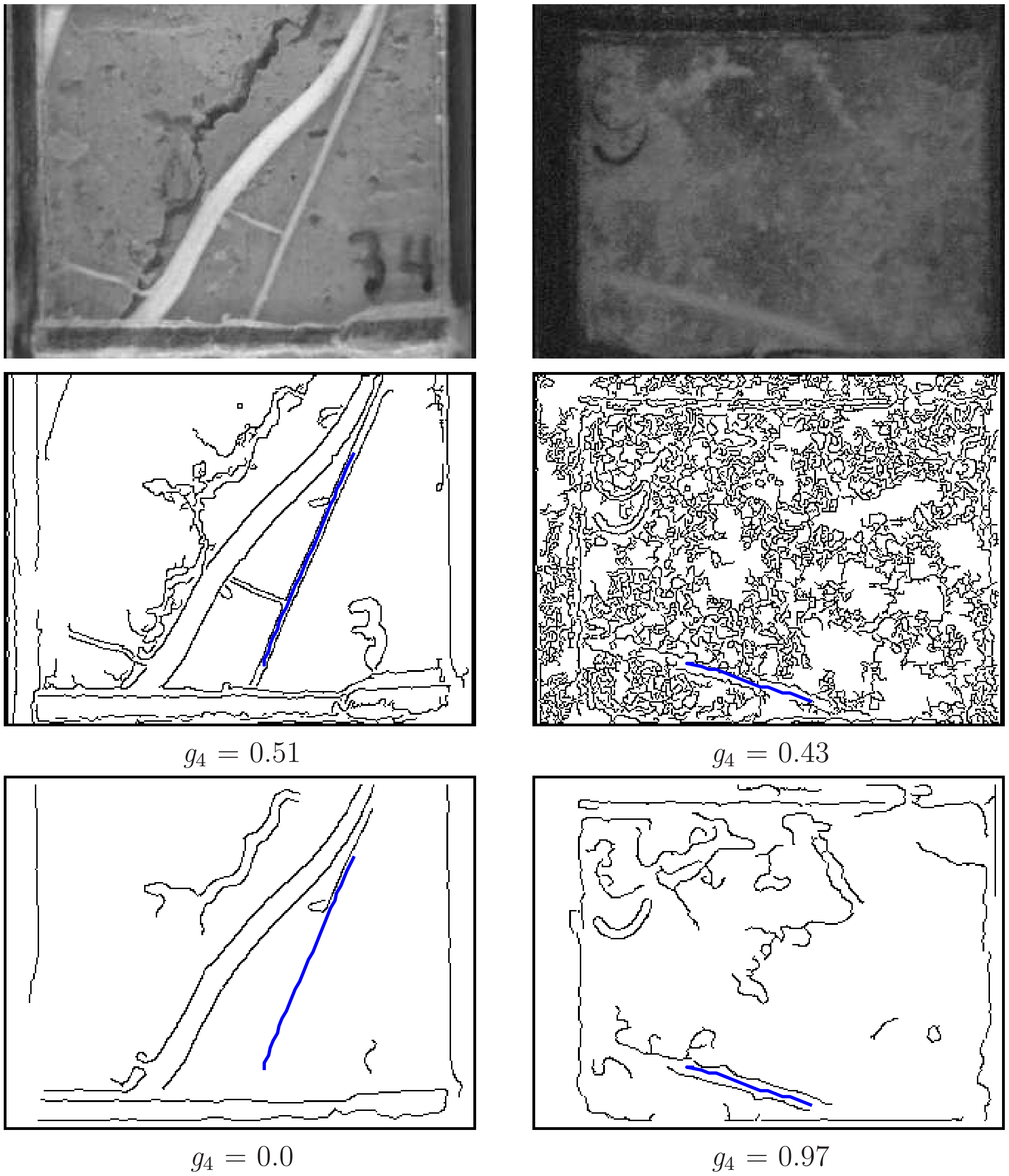


Figure 5.9: Examples of measuring *width stability* on a centerline segment (*blue*) with multi-scale factors in sample minirhizotron root images. *Top to bottom*: original image, detected edges using σ_1 (= 1.0), detected edges using σ_2 (= 2.5).

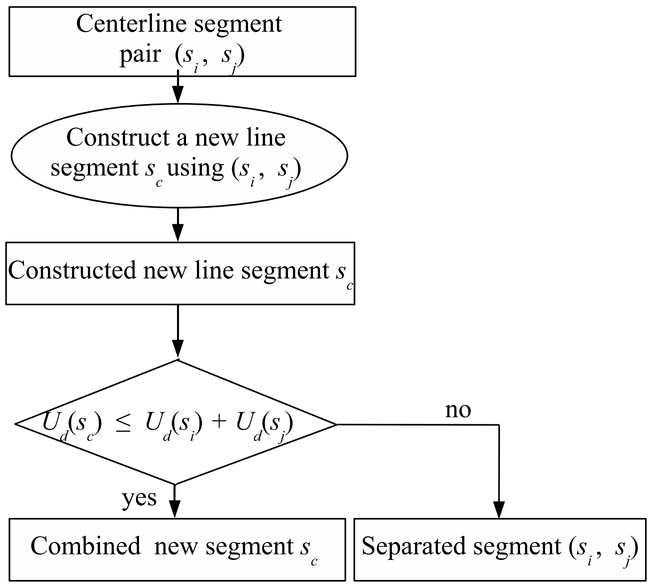

Centerline segment pair (s_i, s_j)
Construct a new line segment s_c using (s_i, s_j)
Constructed new line segment s_c
$U_d(s_c) \leq U_d(s_i) + U_d(s_j)$
no
yes
Combined new segment s_c
Separated segment (s_i, s_j)

Figure 5.10: The object process diagram of the centerline combination step.

for computational reasons. As a result, the intensity energy considered here is based only on spread g_1 and residue g_2, so that line segments are combined if they are near each other and well aligned, according to the weighting function described previously. The OPD of this centerline combination step is shown in Figure 5.10.

In addition to combinability, we introduce the *connectability* function to join attracted line segments that belong to a curved linear feature whose orientation is not constant based on their interaction energy. We define the connectability of two attracted line segments s_i and s_j as the probability $P(D_{conn} \mid s_i, s_j) = \exp\{-\psi(D_{conn} \mid s_i, s_j)\}$, where the energy function is based on the *proximity* and *alignment*: $\psi(D_{conn} \mid s_i, s_j) = w_1' h_1(s_i, s_j) + w_2' h_2(s_i, s_j)$. The weights are determined by applying a linear discriminator $\mathbf{u}_d$ to the training data, as shown in Figure 5.3d. The OPD of this centerline connection step is shown in Figure 5.11.

Due to the bifurcation and crossover of linear features, sometimes centerline segments from different linear features may be misconnected. To avoid the problem of misjoining, for any centerlines segment s_i with m (where $m \geq 2$) interacted line

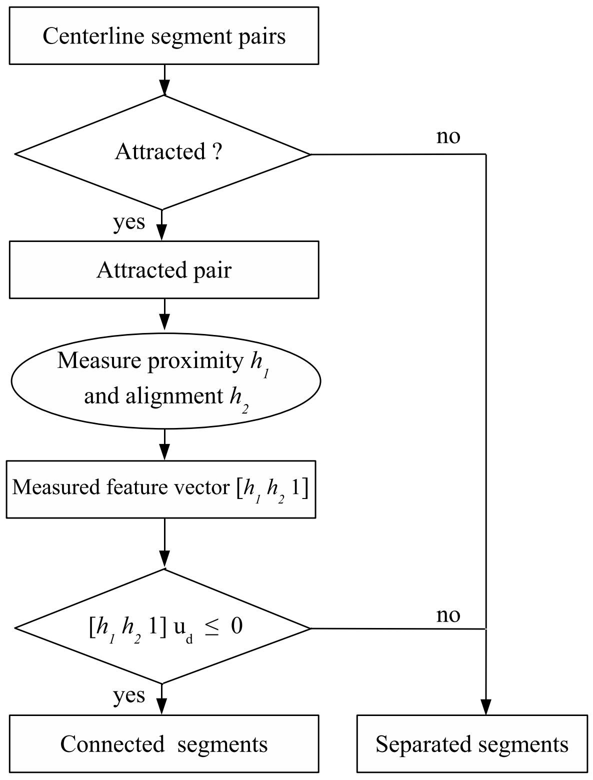

Centerline segment pairs
Attracted ?
no
yes
Attracted pair
Measure proximity h_1 and alignment h_2
Measured feature vector $[h_1\, h_2\, 1]$
$[h_1\, h_2\, 1]\, u_d \leq 0$
no
yes
Connected segments
Separated segments

Figure 5.11: The object process diagram of the centerline connection step.

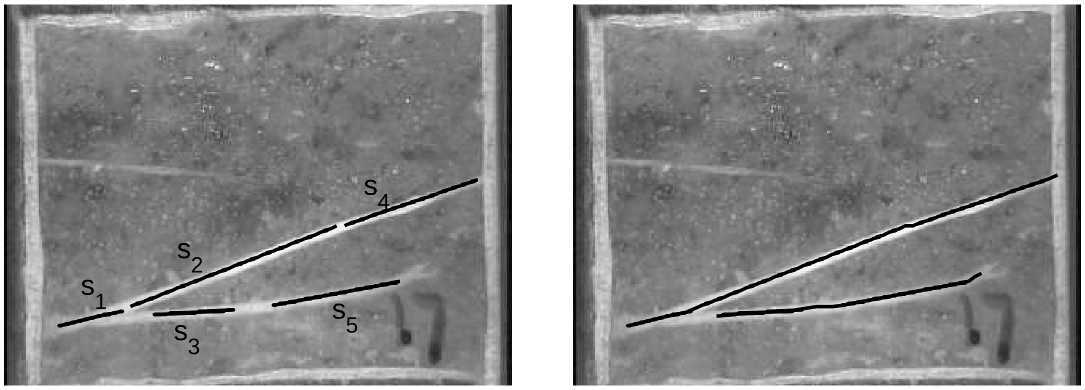


Figure 5.12: An example of connecting centerline segments with multi-interaction. *Left to right*: detected centerline segments, connected centerlines. Although both the two pairs (s_1, s_2) and (s_1, s_3) can be connected, only (s_1, s_2) is connected because $U_I(s_1, s_2) < U_I(s_1, s_3)$.

segments at the same end point, we connect it to the segment s_j which produce the minimum interaction energy $U_I(s_i, s_j)$:

$$\{s_i, s_j\} = \underset{l=1,\dots,m}{\operatorname{argmin}}\{U_I(s_i, s_l)\} = \underset{l=1,\dots,m}{\operatorname{argmin}}\{\psi(D_{conn} \mid s_i, s_l)\}. \tag{5.1}$$

As shown in Figure 5.12, although the two segments s_2 and s_3 interact with s_1 at the same end point, and both the two pairs (s_1, s_2) and (s_1, s_3) can be connected, we only connect (s_1, s_2) because $U_I(s_1, s_2) = 0.91$ is less than $U_I(s_1, s_3) = 1.31$.

5.3 Centerline Tracing

Because some linear features have low contrast to the background in images, we cannot collect enough seed points to build a centerline segment in this region. Meanwhile, because some linear features are covered by background noise in images, we cannot fit a centerline segment to the detected seed points in this region.

To remedy this problem, a modified centerline tracing method [4] is applied.

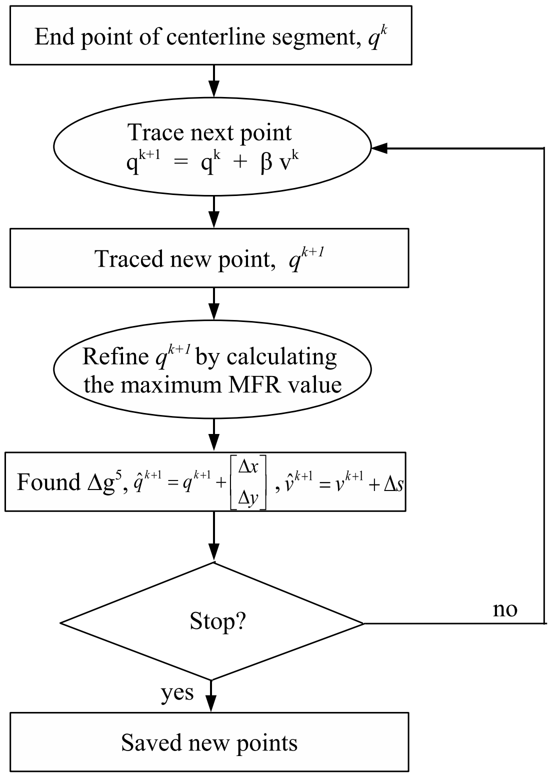

End point of centerline segment, q^k
Trace next point $q^{k+1} = q^k + \beta v^k$
Traced new point, q^{k+1}
Refine q^{k+1} by calculating the maximum MFR value
Found Δg^5, $\hat{q}^{k+1} = q^{k+1} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$, $\hat{v}^{k+1} = v^{k+1} + \Delta s$
Stop?
no
yes
Saved new points

Figure 5.13: The object process diagram of the centerline tracing step.

For each centerline segment, using its two endpoints as the initial points q^k and the orientation of the centerline s^k as the initial orientation (k is the iteration number, $k = 0$ for the initial point), this method can recursively grow next point q^{k+1} and its orientation s^{k+1} to extend the linear feature centerline. The OPD of this centerline tracing step is shown in Figure 5.13.

Denoting $\mathbf{v^k}$ as a unit vector along the linear feature centerline at point q^k as

$$\mathbf{v^k} = \begin{bmatrix} \mathrm{v_x^k} \\ \mathrm{v_y^k} \end{bmatrix} = \begin{bmatrix} \cos(\mathrm{s^k}) \\ \sin(\mathrm{s^k}) \end{bmatrix}, \tag{5.2}$$

we trace the next point q^{k+1} as

$$q^{k+1} = q^k + \beta \mathbf{v^k}, \tag{5.3}$$

where β is the step size.

Because of the curvature of the linear feature, the initial direction may not be along the actual orientation of the linear feature at a new position q^k. Therefore, the searching direction needs be adjusted at these positions. Once a new point q^{k+1} is found along the direction s^k, we calculate its matched filter response (MFR) values at s^k and the nearby directions. Meanwhile, to find the accurate location of a centerline point, the MFR values of the neighbor pixels in the 3×3 window centered on the detected point q^{k+1} are checked at the hypothesized direction.

For each tested point, the corresponding MFR value at the hypothesized direction is added to estimate the change of the *smoothness* feature g_5 of the centerline. To capture the information during the procedure, an "refinement vector" is defined as $\delta = [\Delta x \quad \Delta y \quad \Delta s]^T$. Therefore, we have

$$\Delta g_5 = \min_{\Delta s, \Delta x, \Delta y = -2, \ldots, 2} \{ | \ g_5^{init} - g_5(x^k + \Delta x, y + \Delta y, s^k + \Delta s) \ | \} \tag{5.4}$$

The refinement vector δ corresponding to the minimum increase of *smoothness* is chosen to calculate the new centerline point q^{k+1} and the new searching direction s^{k+1}.

$$\delta^k = \underset{\Delta s, \Delta x, \Delta y = -2, \ldots, 2}{\operatorname{argmin}} \{ | \ g_5^{init} - g_5(x^k + \Delta x, y + \Delta y, s^k + \Delta s) \ | \} \tag{5.5}$$

where g_5^{init} is the *smoothness* value of the original centerline.

With the refinement procedure, the recursion equation Equation 5.3 is modified as,

$$q^{k+1} = q^k + \hat{q}^k + \beta \hat{\mathbf{v}}^k \tag{5.6}$$

where

$$\hat{q}^k = \begin{bmatrix} \delta^k(1) \\ \delta^k(2) \end{bmatrix};\ \hat{\mathbf{v}}^k = \begin{bmatrix} cos(s^k + \delta^k(3)) \\ sin(s^k + \delta^k(3) \end{bmatrix}$$

Results of the centerline tracing procedure on sample minirhizotron images are shown in Figure 5.14.

The tracing procedure is stopped if one or more of the following criteria are satisfied:

1. The new point q^{k+1} is out of the image frame (Figure 5.14.c),
2. The extended centerline intersects with other centerlines (Figure 5.14.b),
3. The iteration number reaches the pre-set limit N^T,
4. The increase of the smoothness Δg_5 at the point q^{k+1} is larger than a pre-set threshold $\Delta {g_5}^T$ (Figure 5.14.a),

5.4 Region Detection

Up to now, we have modeled a linear feature as a sequence of centerline segments. In some research areas such as horticulture, it is also important to determine the width and linear extent of a linear feature. To accomplish this objective, a procedure that we call *constrained floodfill* is applied to the extracted centerlines. An intensity value v is defined as the minimum graylevel of all the seed points that lend support to the linear feature centerline. The region is grown from these seed points

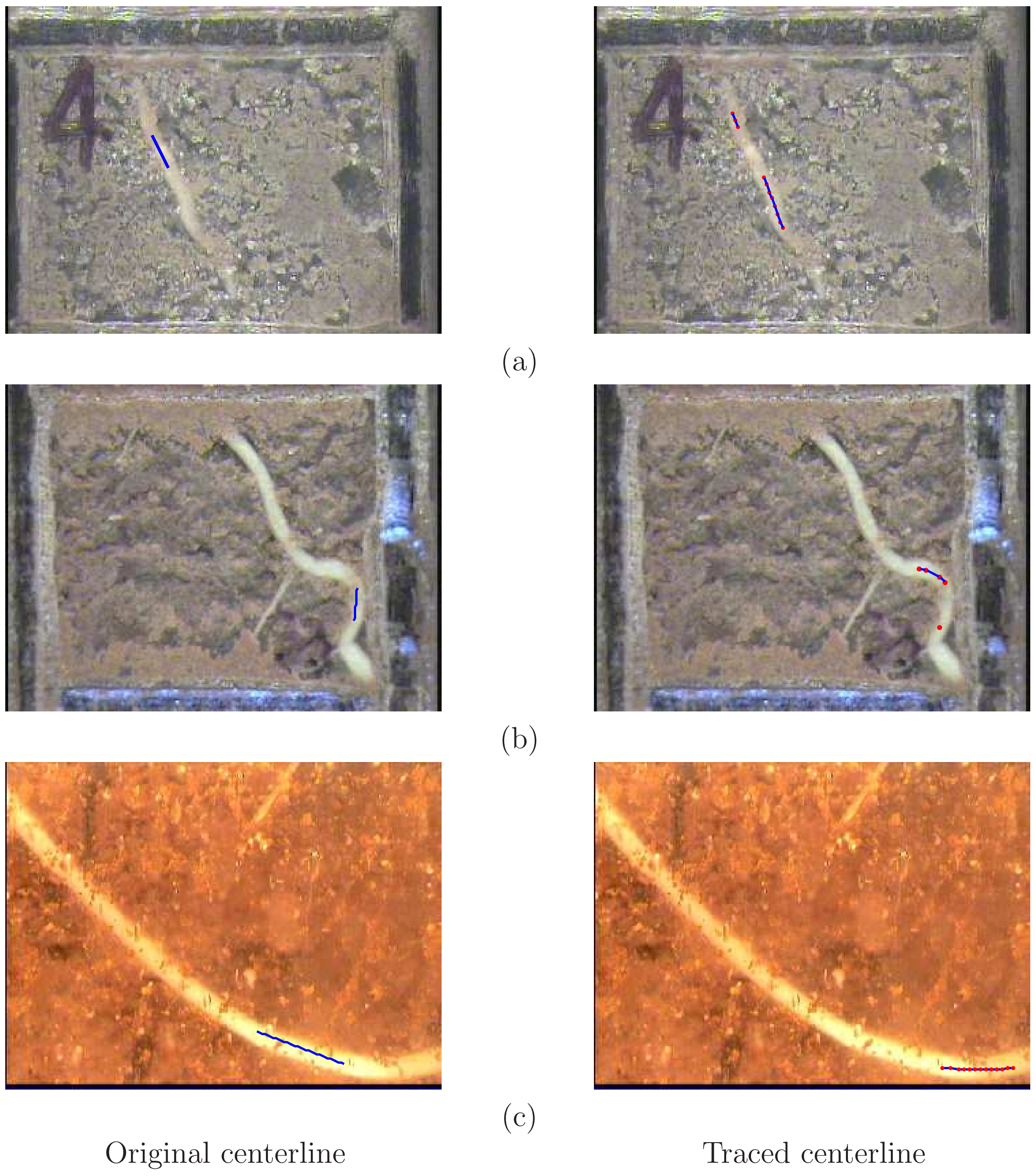


Figure 5.14: Examples of centerline tracing in sample minirhizotron root images. New centerline points *(dot, red)* are traced from the two end points of an original centerlines *(line, blue)*.

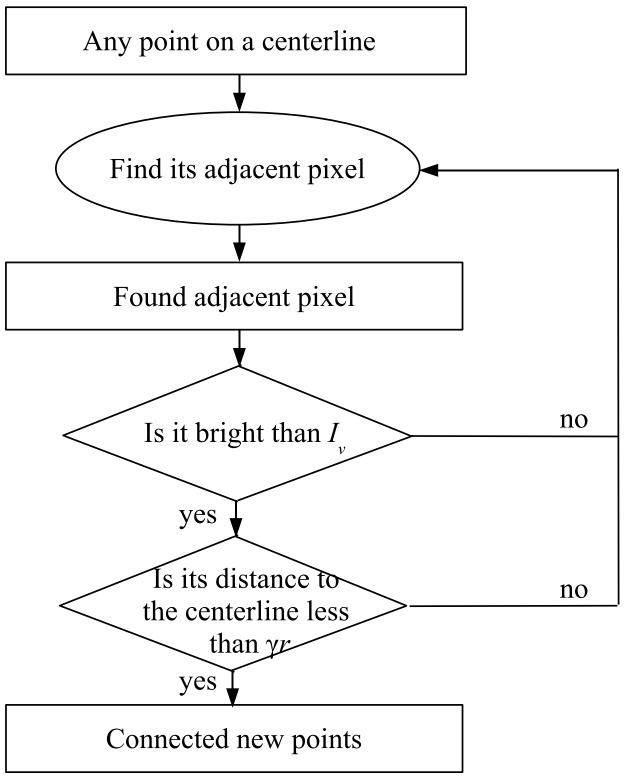


Figure 5.15: The object process diagram of the region detection step.

into all the connected pixels whose graylevel is at least as bright as I_v and whose location is within γr pixels of the centerline, where r is the average width of the linear feature measured during the validation step, and $\gamma = 0.5$ is a constant. The OPD of this region detection step is shown in Figure 5.15.

Figure 5.16 presents two examples of linear feature region detection from a centerline which contains one segment and multiple segments in sample minirhizotron root images. When this method is applied to a centerline segment containing multiple segments, for every detected pixel, its distance to each of these segments is calculated. If the minimum distance is less than γr, this pixel is filled to the linear feature region.

Although this simple procedure works much of the time, it fails in two cases shown in Figure 5.17. First, when the linear feature is partially occluded by dark background noise, disconnected regions may be produced for the same linear feature.

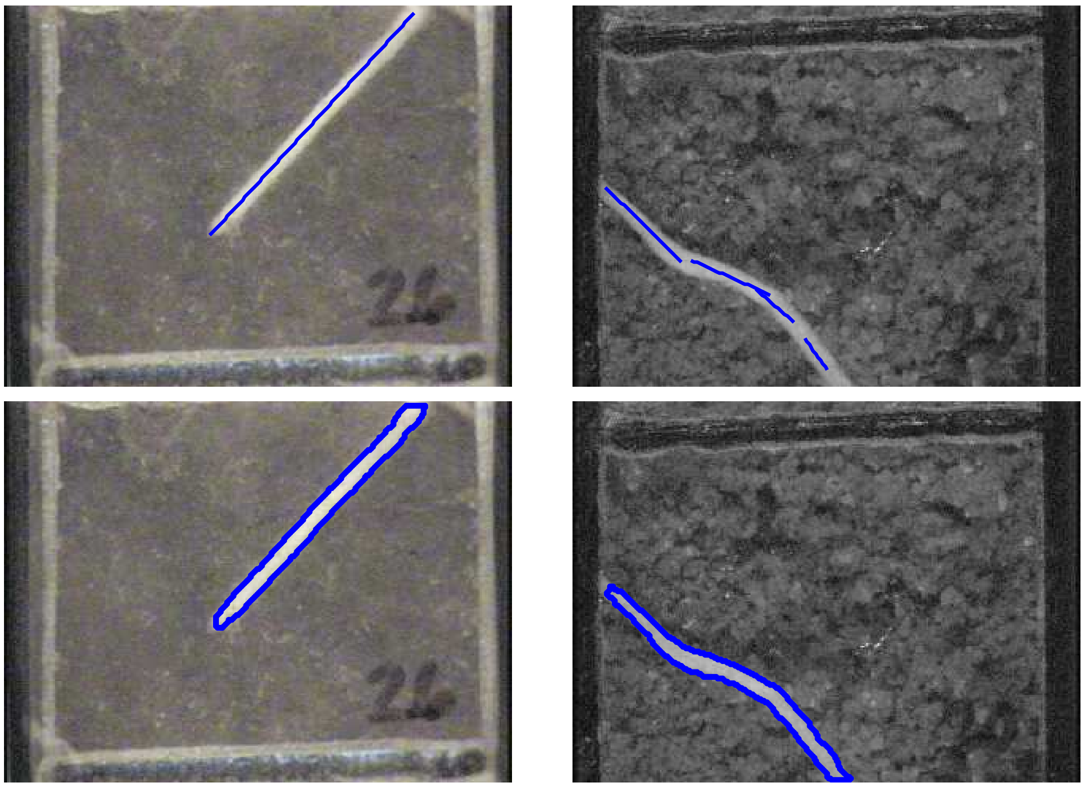


Figure 5.16: Examples of root region detection (*bottom*) from its centerline (*top*). *Left to right*: root region detection from a centerline containing a single segment, and from a centerline containing multiple segments.

To handle this situation, an occluded dark pixel is automatically selected between the two separate seed points along the centerline as the start point, then the floodfill algorithm is applied again to detect the occluded region. The region growing process stops when it reaches the previously detected regions.

Secondly, when the centerline algorithm of the previous section does not connect all the segments of a linear feature, the growing procedure will result in disconnected regions. To solve this problem, an additional linear feature connection process similar to the connectability function previously described is applied. As before, the intersection type of any two linear feature sections is determined as an *attraction* if they overlap near their ends.

Let R_1 and R_2 be two linear feature regions with an attraction intersection. The intersection $I = R_1 \cap R_2$ between the regions is computed, as well as the leftover regions $R_1' = R_1 \backslash I$ and $R_2' = R_2 \backslash I$. Let $\lambda = \frac{|I|}{\min(|R_1|,|R_2|)}$ be the overlap ratio, and let φ_{R_1,R_2} be the angle between the two lines connecting the centroid of I and the centroids of R_1 and R_2. Let $| R_1'' |$ be the size of the second-largest connected component of R_1', or 0 if R_1' has only a single component; and let $| R_2'' |$ be defined similarly. Then the two regions are combined if their overlap is significant or if their angles are compatible and no large leftover regions exist: $\lambda \geq \tau_\lambda$ or ($\varphi_{R_1,R_2} \leq \tau_\varphi$ and $| R_1'' | < \tau_r$ and $| R_2'' | < \tau_r$). We set $\tau_\lambda = 0.7$, $\tau_\varphi = 80$ degrees, and $\tau_r = 50$ pixels.

In the previous section, we introduced a method for tracing linear feature centerlines at regions have low-contrast to the background or region have curvature. If the traced points have no significant deviation from the original centerline, we simply add them to the original centerline for linear feature region detection. However, if the points are traced from a curvature of the linear feature (Figure 5.18), they may be ignored during the linear feature region detection step if their distances to the original centerline are larger than the threshold γr. Therefore, if the number of traced points

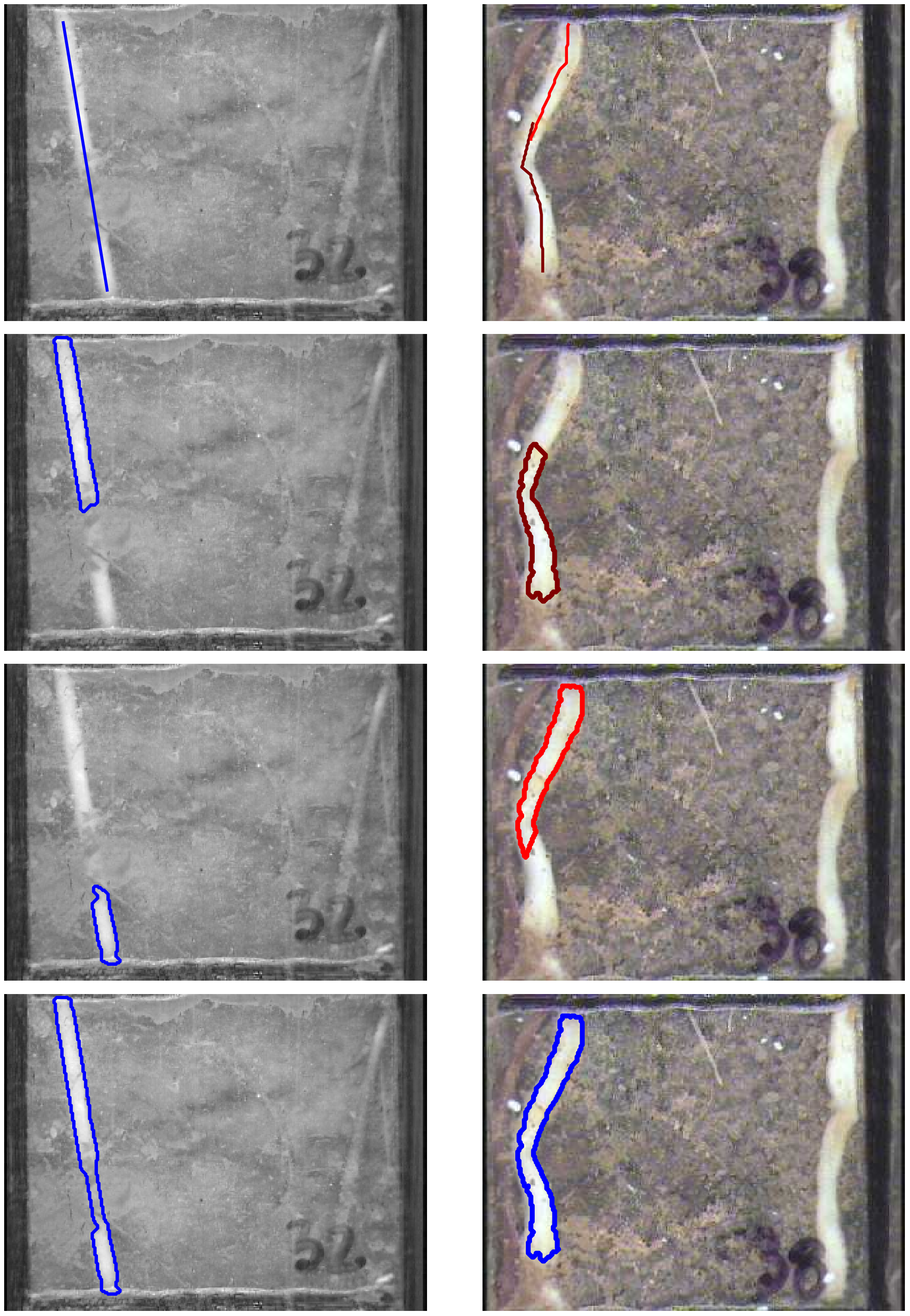


Figure 5.17: Examples of root region detection on sample images with background occlusion (*left*) and broken centerline (*right*). *Top to bottom*: detected centerlines, two detected root regions, and the refined root region.

$m \geq 5$, a new line s_n is fitted to the traced points. We compare the orientation of the new line, θ_{s_m}, with that of the original centerline θ_{s_i}. If $\mid \theta_{s_m} - \theta_{s_i} \mid \geq \theta_T$, our *constrained floodfill* procedure is applied to both of the two lines for linear feature region detection. The detected regions from the two lines are combined directly using logical *Or* operator. Here, we set $\theta_T = 15$ degrees.

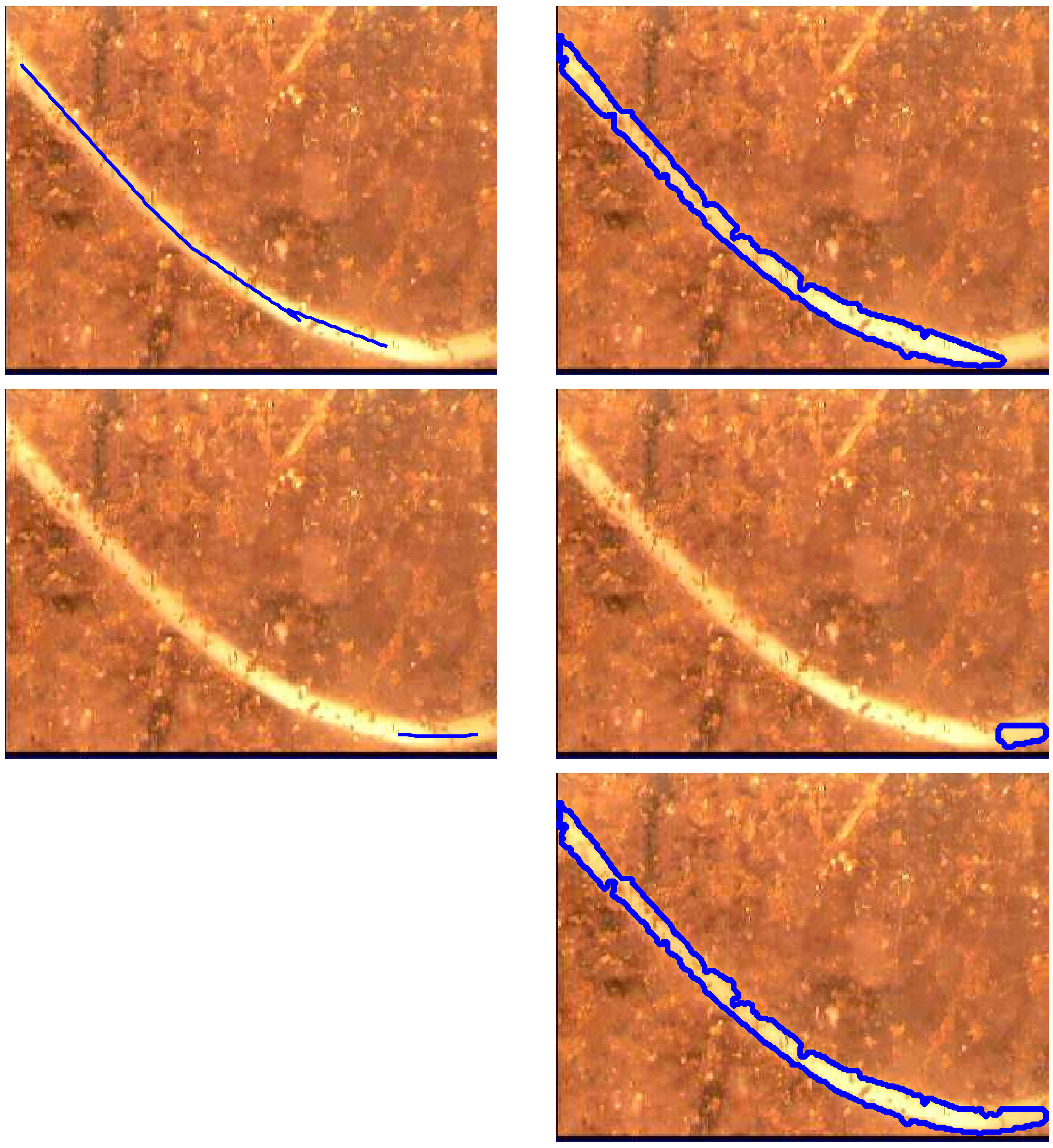


Figure 5.18: An example of root region detection (*right*) when a new segment is fitted to the points traced from the original centerline. *Top to bottom*: original centerline and the detected root region, new segment fitted to the traced points and its detected root region, combined root region.

Chapter 6

Linear Feature Discrimination

In the previous chapter, we described a Gibbs point process-based method for linear feature detection. However, in the case of images containing bright and elongate background objects, there is a need to discriminate the detected unwanted background objects. In the minirhizotron root images (See Figure 6.2), these falsely detected objects may be bright extraneous objects, light soil particles, water droplets or spots caused by uneven diffusion of light through the minirhizotron wall. The purpose of this chapter is to describe a strong classifier we built to discriminate linear feature from unwanted background objects [68]. The OPD of our linear feature discrimination method is shown in Figure 6.1.

6.1 Feature Classifiers

In this paper, we explored both geometric and intensity-based features. The geometric features include the following: (1) eccentricity, (2) approximate line symmetry, and (3) boundary parallelism. The intensity-based features include two additional methods: (4) histogram distribution and (5) edge detection.

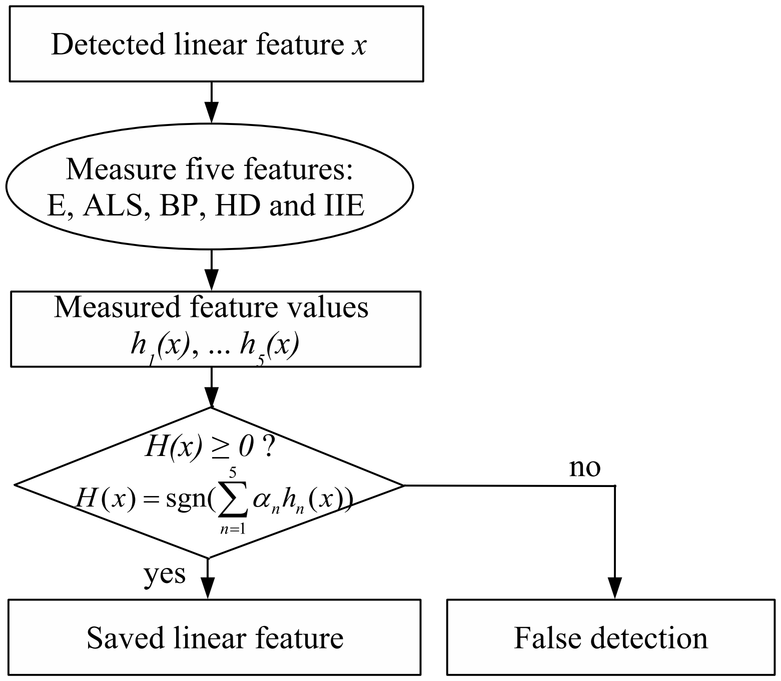

Detected linear feature x
Measure five features: E, ALS, BP, HD and IIE
Measured feature values $h_1(x), \dots h_5(x)$
$H(x) \geq 0$?
$H(x) = \mathrm{sgn}(\sum_{n=1}^{5} \alpha_n h_n(x))$
no
yes
Saved linear feature
False detection

Figure 6.1: The object process diagram of the linear feature discrimination method.

6.1.1 Histogram Distribution (HD)

The histogram of a digital image with gray levels in the range [0, G] is a discrete function $h(r_k) = n_k$, where r_k is the k^{th} gray level and n_k is the number of pixels in the image having gray level r_k [24]. For an 8-bit gray scale image, G is 255. Assuming that linear features are brighter than the background, regions corresponding to linear features should contain many bright pixels. We apply this test by measuring the percentage of pixels in the region with an intensity value greater than $0.8G$, which is accomplished by thresholding the graylevel histogram of the region. As shown in Figure 6.3 and Table 6.1, this percentage in minirhizotron images was low for non-root objects and significantly higher for actual roots.

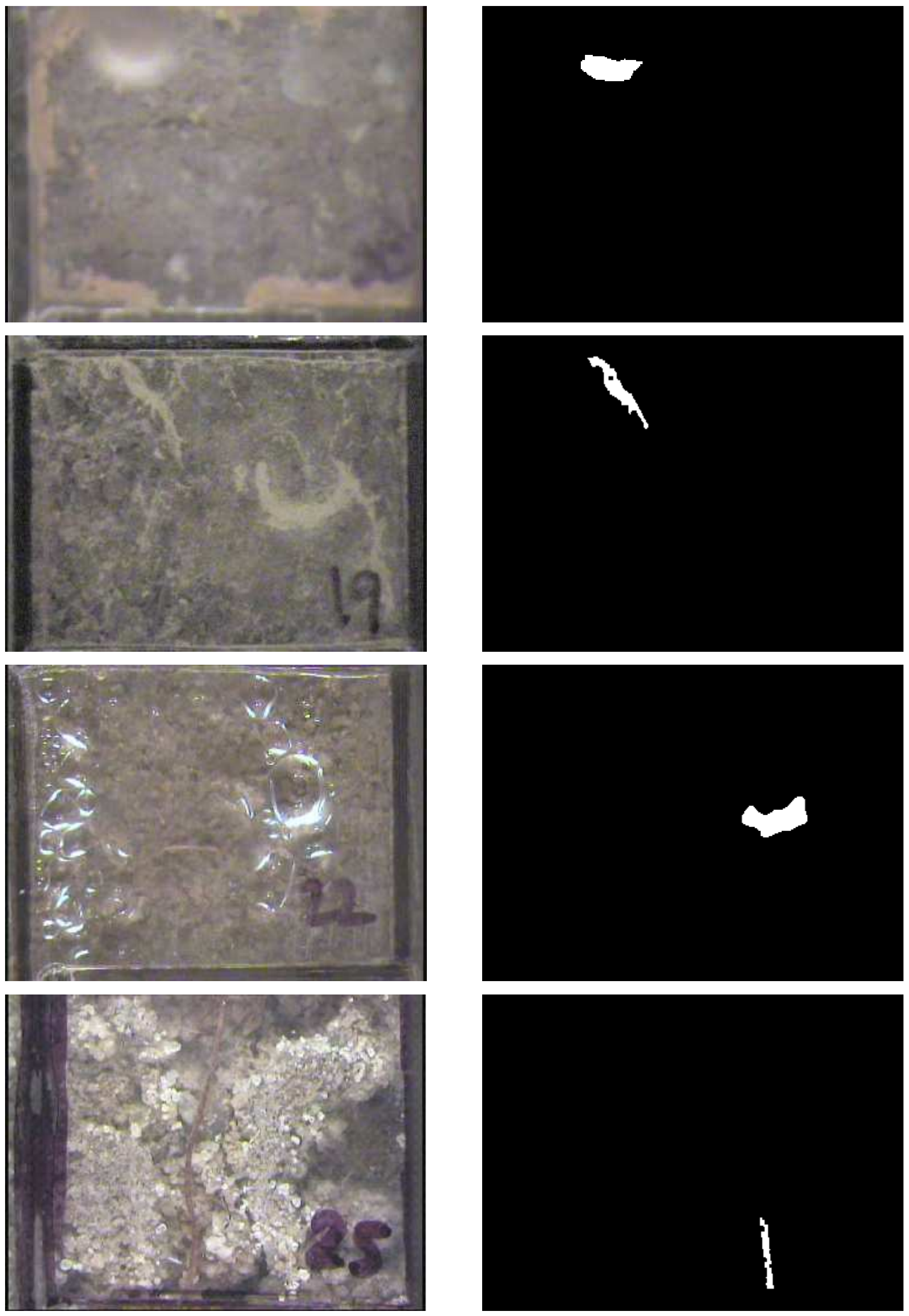


Figure 6.2: Examples of falsely detected roots. From *top* to *bottom*: bright extraneous object, light soil particles, water droplets, spots caused by the uneven diffusion of light through the minirhizotron wall.

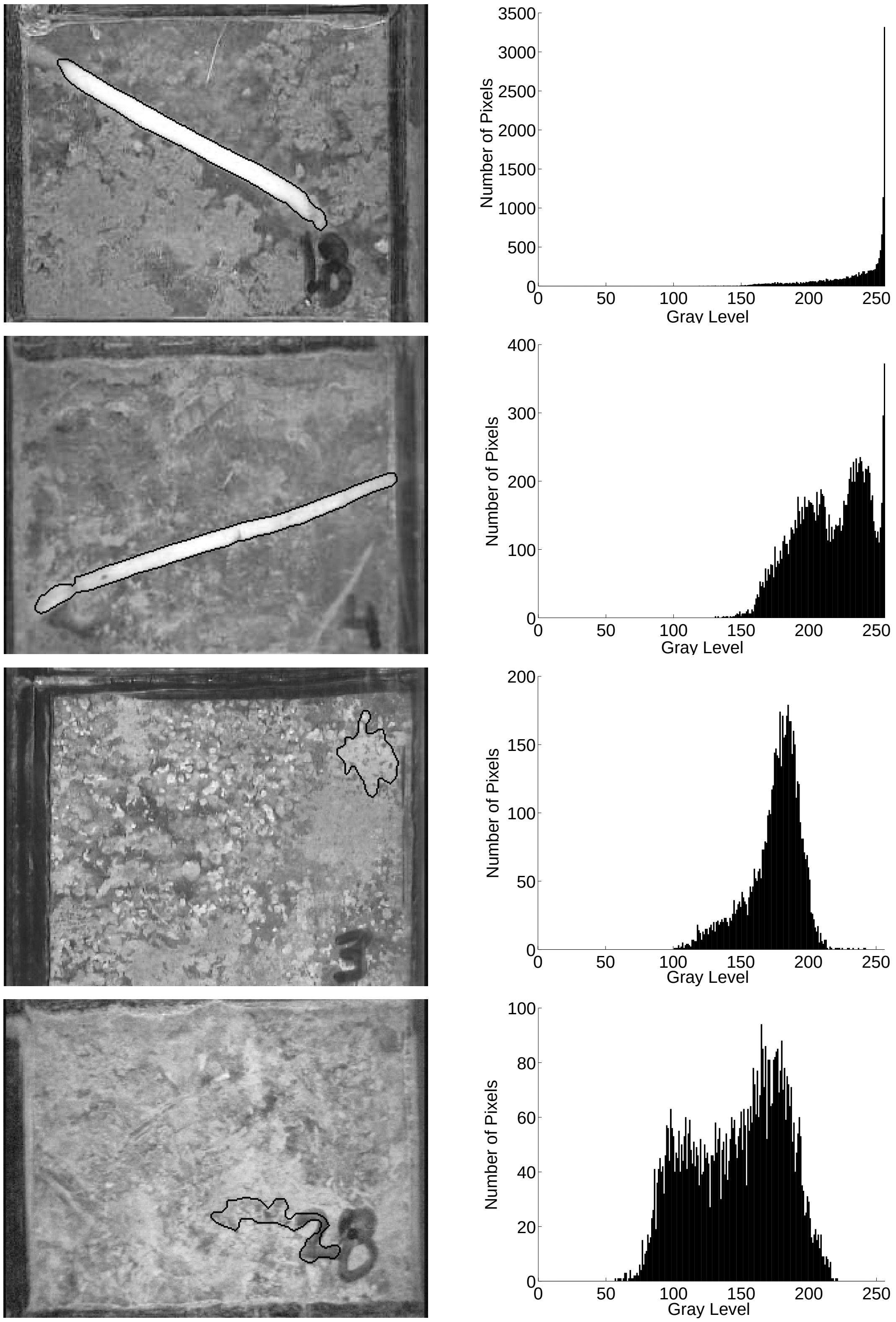


Figure 6.3: Two roots and two non-roots (*left*) with their gray-level histograms (*right*).

6.1.2 Interior Intensity Edges (IIE)

An intensity edge is a location in an image where the intensity function changes rapidly. Linear features tend to have smooth interiors with little intensity variation, while unwanted background objects often have appreciable variation in their interior. We convolve the image with a Sobel edge operator and compute the absolute value, to yield the magnitude of the intensity edges in the region. We then threshold these values and sum them to produce a count of the edge pixels in the region. The proportion of interior pixels was significantly greater for roots than for non-roots (Figure 6.4, Table 6.1) in minirhizotron images.

6.1.3 Eccentricity (E)

Given a $2 \times N$ matrix A containing the centralized 2D image coordinates of the N points in the region, the 2×2 covariance matrix K is formed as the outer product of A:

$$\mathrm{K} = \mathrm{AA}^T = \tfrac{1}{N-1}\sum_{i=1}^{N}(X_i - \mu)(X_i - \mu)^T \tag{6.1}$$

where X_i is 2×1 vector, and μ is the centroid of the region. According to principal components analysis (PCA) [55], the lengths of the major and minor axes of the best ellipse to fit the region are given by the square roots of the eigenvalues, λ_i, of K. The eccentricity of a region is the ratio of the lengths of these two axes: $\frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}}$, where $\lambda_1 \geq \lambda_2$. The eccentricity ranges from 1 to infinity, with 1 indicating a perfect circle. In minirhizotron images, roots tend to be long and narrow, giving them a higher value for eccentricity than most non-root objects (Figure 6.6, Table 6.1).

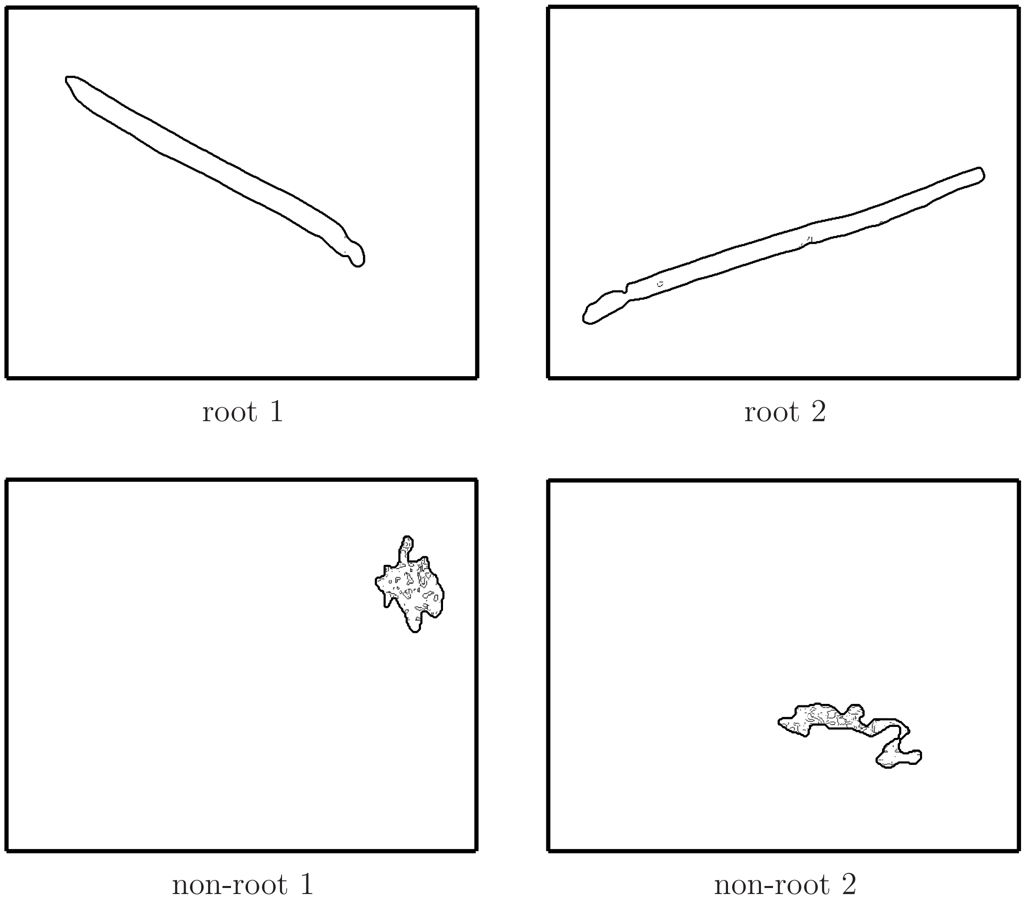


Figure 6.4: Two roots and two non-roots, with the results of edge detection. Edges are black.

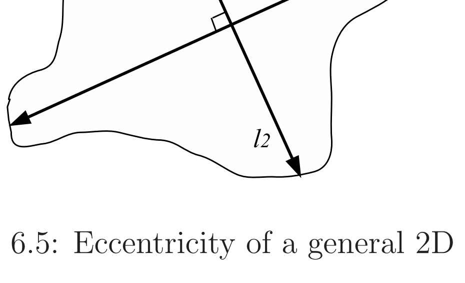


Figure 6.5: Eccentricity of a general 2D shape.

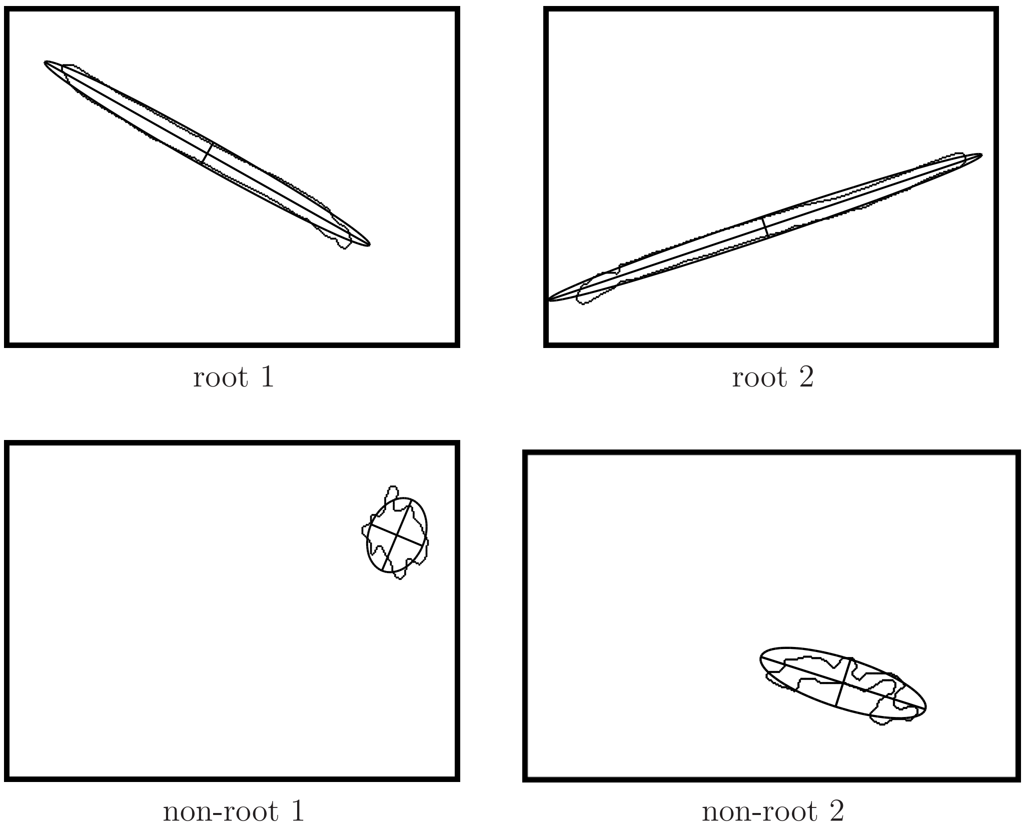


Figure 6.6: Two roots and two non-roots, with the results of the eccentricity computation.

6.1.4 Approximate Line Symmetry (ALS)

A geometric shape is said to be symmetric with respect to its central curve if the central curve bisects all line segments that are perpendicular to it and terminate at the shape outline. As shown in Figure 6.7, the 2D shape of the linear feature approximates an elongated rectangle and has approximate line symmetry with respect to its central curve, while the unwanted background objects tend to have irregular shape and show low symmetry.

To calculate line symmetry, we first extract the central curve of an identified region using Dijkstra's algorithm. For each point C_i on the central curve C we search along the line perpendicular to the central curve at C_i to find the two points that intersected the boundary. Let $b_{i,1}$ be the distance from C_i to one intersection point and $b_{i,2}$ the distance from C_i to the other intersection point. We calculate the proportion of points along C such that $max(b_{i,1}, b_{i,2})/min(b_{i,1}, b_{i,2}) \leq 1.05$. In minirhizotron images, this proportion is close to one for roots, while it is significantly lower for non-root objects (Table 6.1).

6.1.5 Boundary Parallelism (BP)

Because the width of a linear feature is approximately constant or varies continuously, the opposite boundaries of the linear feature should be nearly parallel. This test is very similar to the previous one. Making use of the found central curve C, for each point C_i on C, we find its corresponding opposite boundary point pair whose joining line is perpendicular to C at C_i as before, and then we compare the direction of the image gradient at the two point. The gradient is computed using the Sobel edge detector, as before. The proportion of points along C for which the angle between the lines was less than 10 degrees is used to assess the likelihood that the

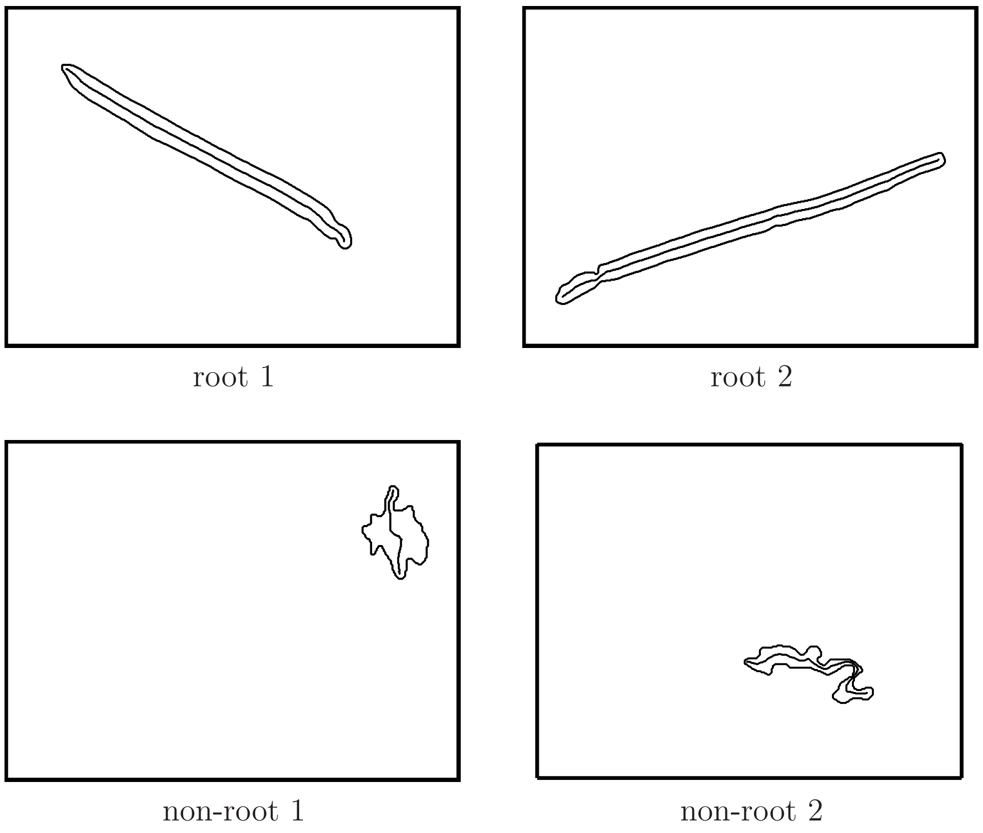


Figure 6.7: Two root images and two non-root images, with central axis displayed.

Classifier	root 1	root 2	non-root 1	non-root 2
Histogram Distribution	**0.87**	**0.66**	0.02	0.02
Interior Intensity Edges	**0.99**	**0.98**	0.89	0.86
Eccentricity	**14.62**	**20.72**	1.38	3.49
Approximate Line Symmetry	**0.99**	**0.99**	0.57	0.48
Boundary Parallelism	**0.92**	**0.88**	0.05	0.15

Table 6.1: The results of the five measurements on the sample peach images. For each row, the largest two numbers are in bold.

object was a root in minirhizotron images (Table 6.1).

6.1.6 Performance Evaluation

The ability of individual classifiers to discriminate linear features from unwanted background objects was assessed using receiver operating characteristic (ROC) curves. For each weak classifier, we plotted the true positive rate (TPR) against the false positive rate (FPR). The TPR is the ratio of correctly-identified linear features to the total number of images containing linear features. FPR is the ratio of unwanted background objects incorrectly identified as linear features to the total number of images without linear features.

Every point on the ROC curve represents a (TPR, FPR) pair created by a choice of threshold value for the classifier, i.e. the value used to separate linear features from unwanted background objects. We defined the optimal threshold (OT) as the value corresponding to the point of intersection between the ROC curve and the equal error rate (EER) line, i.e. the diagonal connecting (1,0) to (0,1).

Since our training set includes minirhizotron images of three species (*peach*, *sweetbay magnolia* and *Freeman maple*), optimal thresholds of each individual classifier are trained for each species separately (Figure 6.8 and Table 6.2).

As we can see in Table 6.2, values for individual geometric classifiers differed

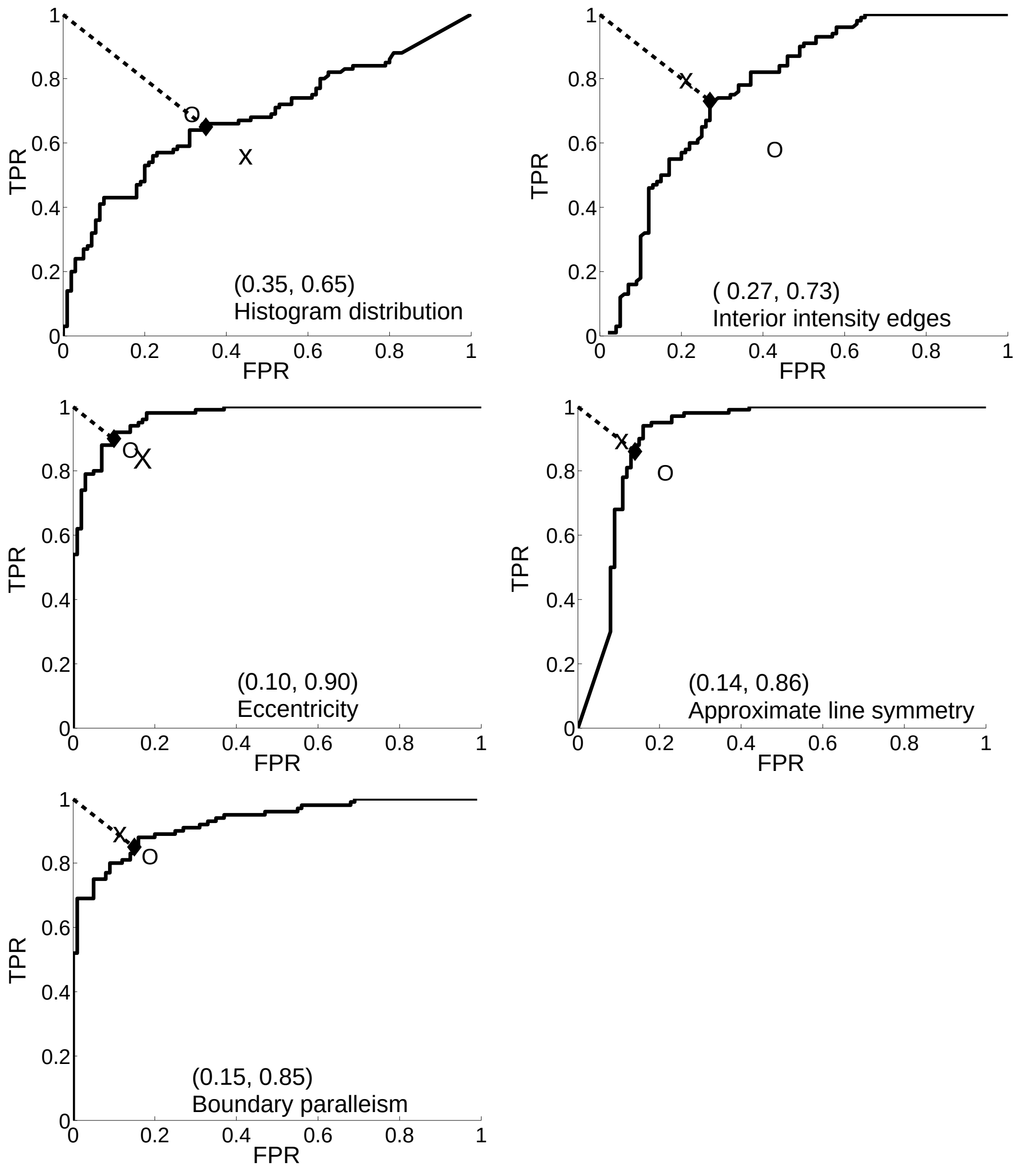


Figure 6.8: Receiver operating characteristic (ROC) curves for individual root classifiers evaluated with peach training set images. Values in parentheses represent the false positive rate (FPR) and the true positive rate (TPR), respectively, at the optimal threshold (OT). Points indicating FPR and TPR values for classifiers at their optimal thresholds evaluated with magnolia (cross) and maple (circle) training images are presented for comparison.

	Root means			Non-root means			Optimal threshold		
	Peach	Magnolia	Maple	Peach	Magnolia	Maple			
Root classifier	(n=100)	(n=50)	(n=60)	(n=100)	(n=50)	(n=60)	Peach	Magnolia	Maple
Histogram Distribution	0.24	0.11	0.23	0.16	0.024	0.058	0.18	0.0013	0.031
Interior Intensity Edges	0.97	0.98	0.94	0.91	0.92	0.91	0.96	0.97	0.95
Eccentricity	7.0	8.3	8.7	3.2	4.3	3.8	5.3	5.3	5.5
Approximate Line Symmetry	0.85	0.98	0.97	0.72	0.72	0.77	0.97	0.96	0.96
Boundary Parallelism	0.46	0.61	0.63	0.27	0.29	0.30	0.41	0.47	0.47

Table 6.2: Mean value and optimal thresholds for five individual root classifiers calculated from peach, maple and magnolia training set images. The optimal thresholds were determined from receive operating characteristic curve. An object with a value higher than the optimal threshold is considered to be a root.

significantly between roots and non-root objects across all image sets. A similar trend was observed for the intensity-based classifiers such as HD and interior intensity edges IIE, although there were several exceptions. Optimal thresholds for all individual classifiers except HD were highly similar across all image sets.

Peach roots differed significantly from both maple and magnolia roots in ALS and BP and differed from magnolia roots in IIE. It is important to note that maple and magnolia root images came from the same site, whereas peach root images came from a different site. Therefore, the effects of site and species cannot be completely separated in this work.

With the exception of HD in peach images, non-root artifacts exhibited no significant differences in any classifier across all image sets. Thus, background artifacts in all image sets appeared quite similar.

The accuracy of the geometric classifiers was greater than that of intensity-based classifiers for all species (Figure 6.8). However, among the geometric classifiers, no single classifier emerged as the most accurate for all species. The most accurate classifier for peach and maple images was eccentricity (E), whereas ALS and BP were most accurate for magnolia images. In training images, the TPR for the best individual classifier ranged from 85% for E in maple to 90% for E in peach. FPRs for the best individual classifier ranged from 15% for E in maple to 10% for E in peach. In general, accuracy of root discrimination with individual classifiers was lowest in maple images.

6.2 Classifier Boosting

Although each of the five methods is able to discriminate linear features from unwanted background objects to some extent, even better performance can be obtained by combining them into a single discriminator. Boosting is a way of generating a strong classifier from several weak classifiers, and AdaBoost [22] is a popular boosting algorithm that operates by considering the classifiers one at a time in a series of learning rounds and dynamically updating the associated weights on both the example data and the classifiers according to the errors in the previous round.

Adaboost operates on a labeled training data set $\{(x_1, y_1), ..., (x_m, y_m)\}$, where $x_i, i = 1, \ldots, m$ are the input vectors and $y_i \in \{-1, +1\}, i = 1, \ldots, m$ are the labels indicating whether the samples are positive (i.e., roots) or negative (non-roots). At round n, each sample $(x_i,\ y_i)$ has an associated weights $D_n(i)$ that indicates the influence that the sample will have on the training of that round. Initially, the weights are assigned according to a uniform distribution, $D_1(i) = 1/m, i = 1, \ldots, m$.

Each round of training involves three steps. First, a weak classifier h_n is applied to the input vectors of the training data to yield $h_n(x_i)$. Secondly, these output values are compared with the ground truth to generate the error ϵ_n of the weak classifier on the weighted samples, where $\epsilon_n = \sum_{i=1}^{m} D_n(i)[h_n(x_i) \neq y_i]$. Finally, this error is used to compute the importance of this weak classifier

$$\alpha_n = \frac{1}{2}\left(\ln \frac{1-\varepsilon_n}{\varepsilon_n}\right) \tag{6.2}$$

as well as the weights of the samples for the next round

$$D_{n+1} = \frac{D_n(i)}{z_n} exp(-(\alpha_n y_i h_n(x_i))) \tag{6.3}$$

where z_n is a normalization constant. In the above equation, notice that $y_i h_n(x_i)$ evaluates to $+1$ if the sample x_i is correctly classified and to -1 if the sample is incorrectly classified. Also notice that $\alpha_n > 0$ since $\epsilon_n < \frac{1}{2}$ without loss of generality, and that α_n gets larger as ϵ_n gets smaller. Thus, the influence of incorrectly evaluated samples increase in future rounds (their weights us multiplied by e^{α_n}), while the influence of correctly evaluated samples decrease (their weights is multiplied by $e^{-\alpha_n}$). After a predetermined number N of rounds, the final strong classifier is given by a linear combination of the weighted weak classifiers:

$$H(x) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n h_n(x)\right), \tag{6.4}$$

In Figure 6.9, we illustrate the Adaboost algorithm with a simple example. Three weak classifiers are applied to categorize ten points into two classes. After three rounds of training, weights are assigned to the classifiers according to their performance, and then a strong classifier is built by combining the three weighted classifiers.

Using the Adaboost algorithm, strong classifiers were developed separately using peach, maple, and magnolia root training images, and ROC curves were constructed to evaluate the performance of the classifiers. Weights of the five weaker classifiers in each of the strong classifier are updated based on their performance. The performance of strong classifiers developed for each species shows that the accuracy is markedly greater than any individual classifier alone (Figure 6.10).

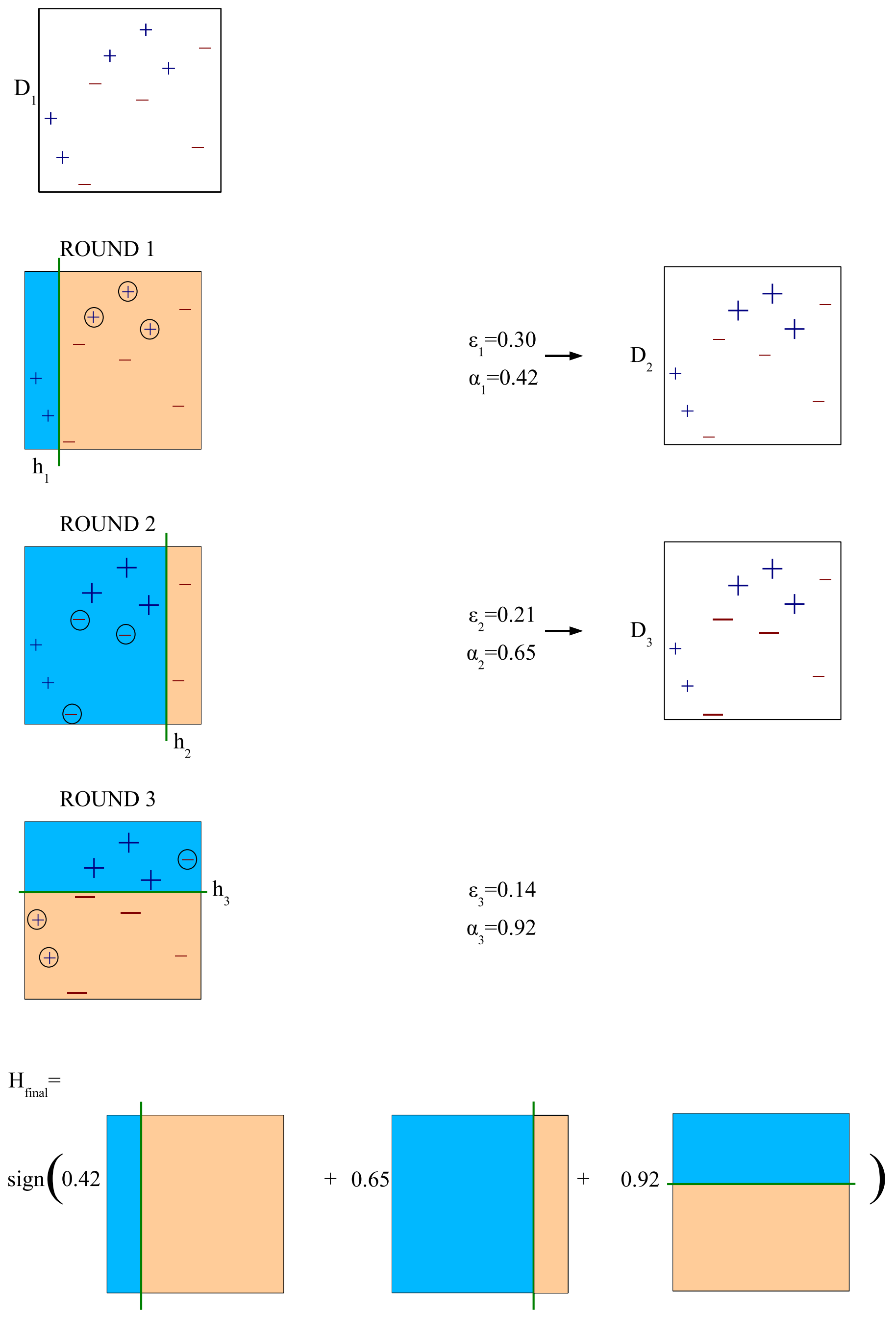


Figure 6.9: Training a strong classifier from three weak classifiers using the Adaboost algorithm. From *top* to *bottom*: Original training set (equal weights are assigned to all training samples), result of classifier h_1 and the weighted samples, result of classifier h_2 and the weighted samples, result of classifier h_3 and the weighted samples, trained strong classifier. Dark gray area is positive, light gray area is negative. Errors from each classifier are marked by *circle*. Size of each sample is proportional to its associated weight.

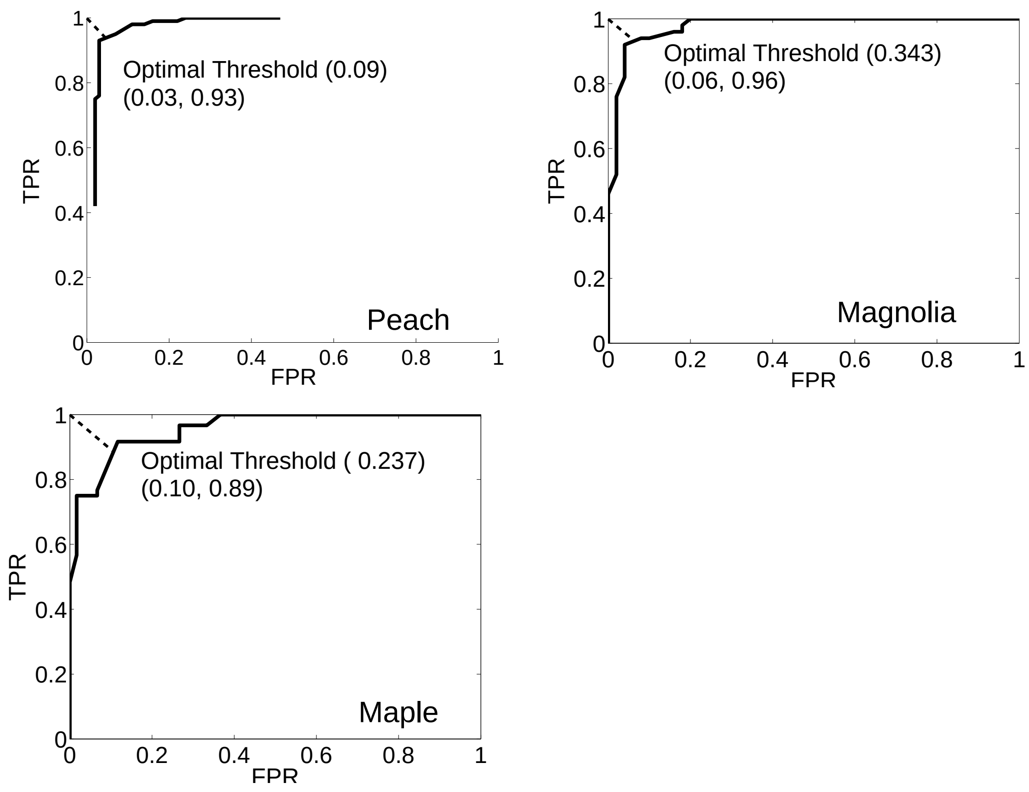


Figure 6.10: Receiver operating characteristic (ROC) curves of strong root classifiers generated by the Adaboost algorithm based on training images of peach, magnolia, and maple roots. Values in parentheses represent the false positive rate (FPR) and the true positive rate (TPR), respectively, at the optimal threshold.

Chapter 7

Linear Feature Measurement

Once a linear feature has been detected, it is necessary to measure its length and diameter. First, the morphological operation of thinning is applied to the binary linear feature image to yield the skeleton, also known as a skeleton tree [8]. Any irregularity in the shape of the linear feature will cause additional branches in the tree. To remove these undesirable artifacts, we apply Dijkstra's algorithm [11] to compute the minimum-length path between any pair of end points on the skeleton, and then the Kimura-Kikuchi-Yamasaki [30] algorithm is introduced to estimate the length of the central curve. The diameter of a linear feature is estimated by a robust average of the length of the line segments that are perpendicular to this curve and extend to the linear feature to background transition. The OPD of our linear feature length measurement method is shown in Figure 7.1.

7.1 Dijkstra's Algorithm

Dijkstra's algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph G with non negative edge path costs, outputting a

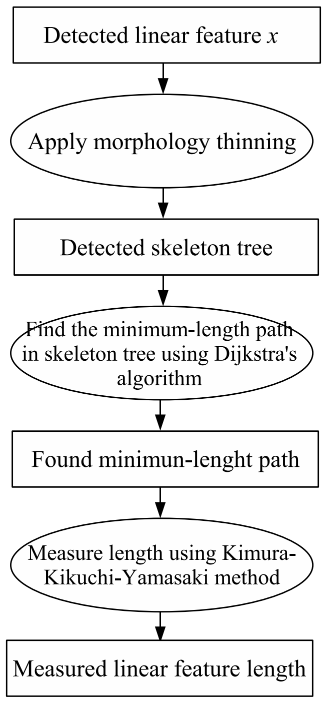

Detected linear feature x
Apply morphology thinning
Detected skeleton tree
Find the minimum-length path in skeleton tree using Dijkstra's algorithm
Found minimun-lenght path
Measure length using Kimura-Kikuchi-Yamasaki method
Measured linear feature length

Figure 7.1: The object process diagram of the linear feature length measurement method.

shortest path tree. The graph $G = (V, E)$ comprises of a set V of N vertices, $\{v_i\}$, and a set E of edges connecting vertices in V. For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined.

Dijkstra's algorithm is illustrated below in Figure 7.2. It maintains as T the set of vertices for which shortest path have not been found, and as d_i the shortest known path from vertex v_s to vertex v_i. Initially, $T = V$ and all $d_i = \infty$. At each step of the algorithm, the vertex v_m in T with smallest d value is removed from T. Each neighbor of v_m in T is examined to see whether a path through v_m would be the shorter than the currently best-known path.

Dijkstra procedure sequential

```
begin
    $d_s = 0$
    $d_i = \infty$, for $i \neq s$
    $T = V$
    for $i = 0$ to $N - 1$
        find $v_m \in T$ with minimum $d_m$
        for each edge $(v_m, v_t)$ with $v_t \in T$
            if $(d_t > d_m + \mathit{length}((v_m, v_t)))$ then $d_t = d_m + \mathit{length}((v_m, v_t)))$
        endfor
        $T = T - v_m$
    endfor
end
```

where *length*(v_m, v_t) is the cost of the edge between vertex v_m and v_t.

Once the skeleton tree of a binary root is extracted, all the end points can be found by checking whether the number of foreground-to-background transition $(1 \rightarrow 0)$ of the 8-connected neighbor of any pixel is one. For each pair of end points that are found, their shortest path is calculated using Dijkstra's algorithm, the pair whose minimum-length path is maximum are selected, along with the path, to yield the central curve of the root. Figure 7.3 shows the skeleton and the resulting central curve for four minirhizotron images.

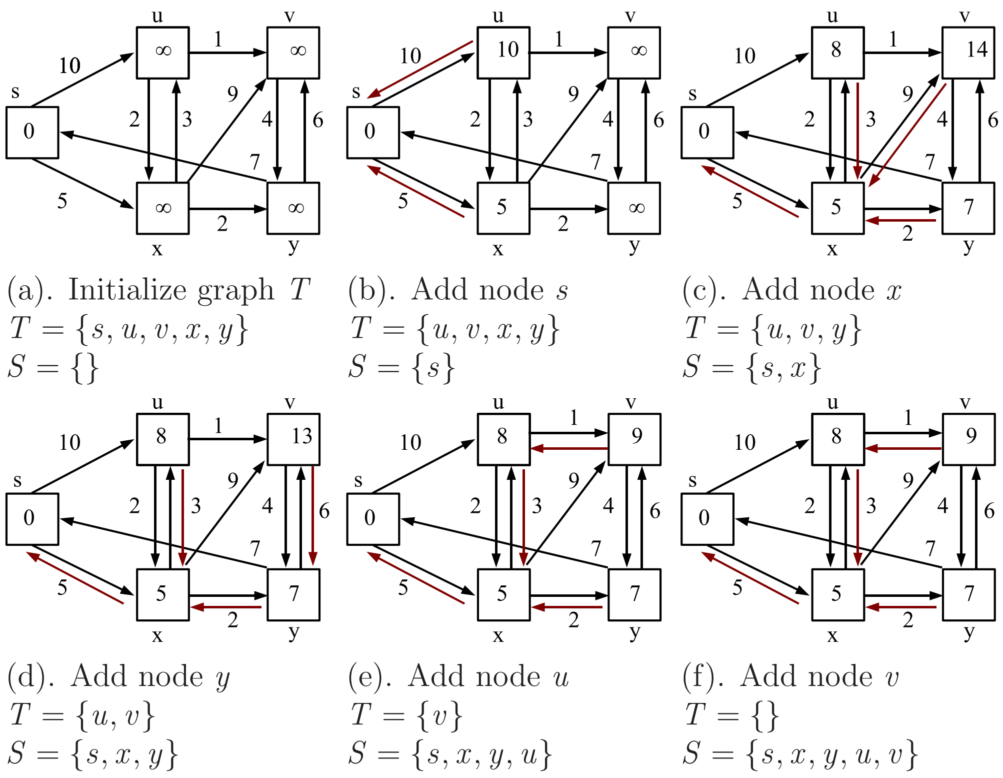


Figure 7.2: An example of shortest path searching using Dijkstra's algorithm.

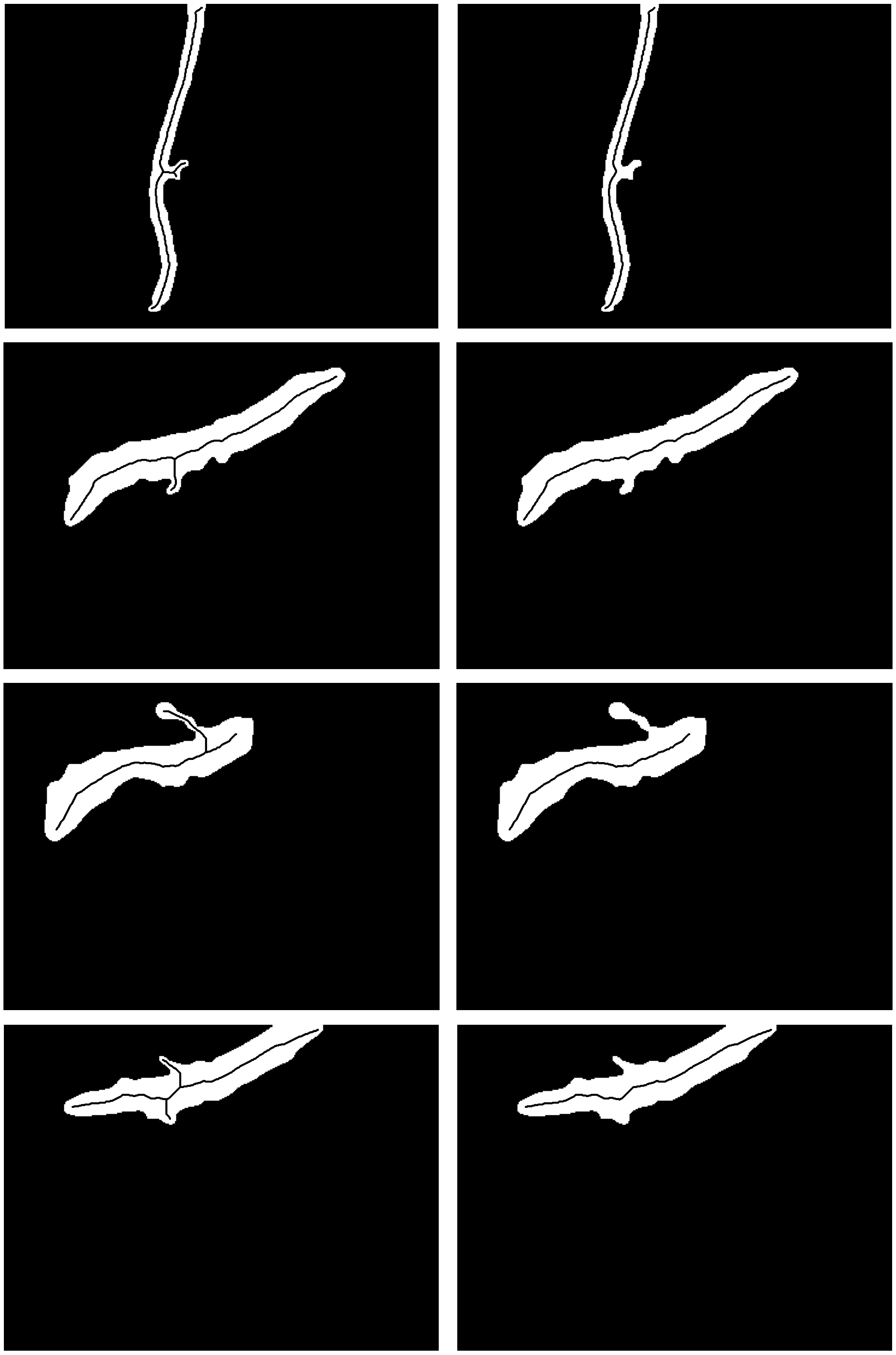


Figure 7.3: For four minirhizotron images, the skeleton tree (top) and the central curve computed by applying Dijkstra's algorithm to the skeleton (bottom).

7.2 Measurement using the Kimura-Kikuchi-Yamasaki method

The central curve is stored as a sequence of nodes (pixels) $\langle C_0, C_1, \ldots C_{N_c}\rangle$, where N_c is the number of nodes. The Euclidean distance along the discretized curve (i.e., the sum of the Euclidean distances between consecutive nodes) is given by

$$L = \sqrt{2}N_d + N_o, \tag{7.1}$$

where N_d is the number of consecutive node pairs (C_i, C_{i+1}) which are diagonally connected and N_o is the number of consecutive node pairs which are adjacent either horizontally or vertically. This equation is also known as the Freeman formula [21]. An alternate approach is to rearrange the node pairs (see Figure 7.4) and use the Pythagorean theorem to estimate the root length as the hypotenuse of the right triangle:

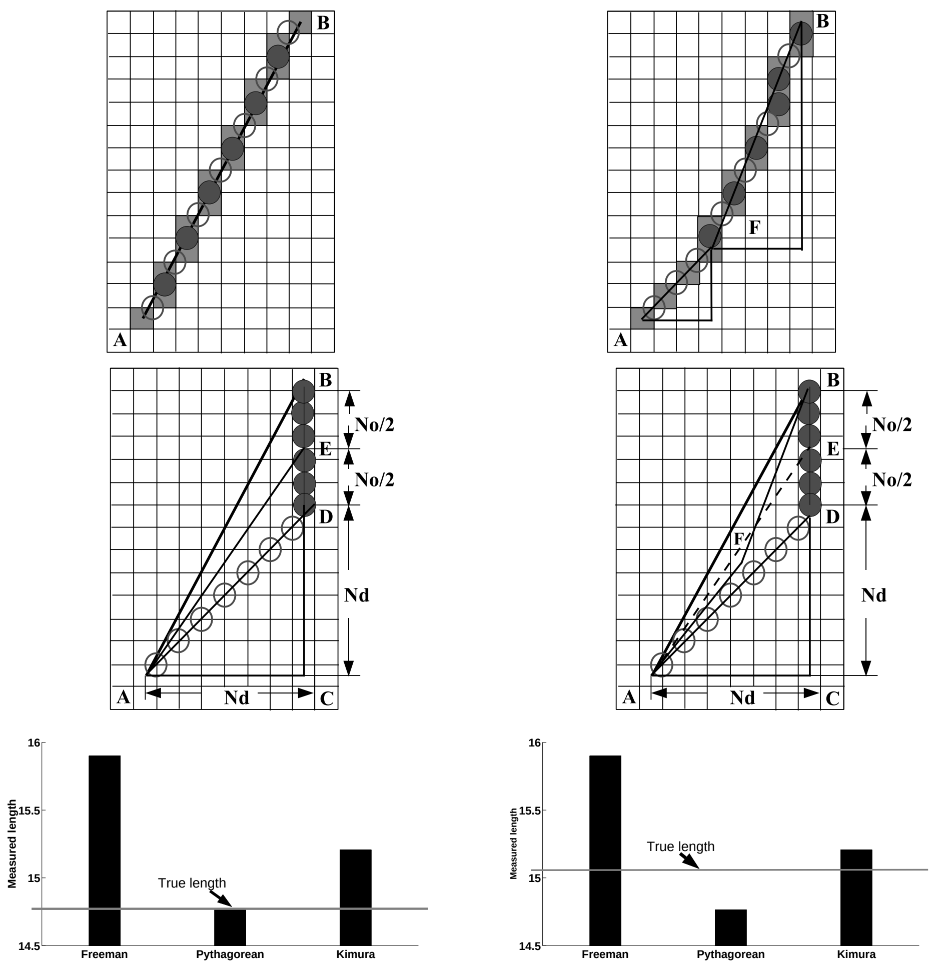
$$L = (N_d^2 + (N_o + N_d)^2)^{1/2}. \tag{7.2}$$

While the Freeman formula generally overestimates the length of a curve [23], the Pythagorean theorem usually underestimates it [33, 16].

Insight into the problem is obtained by noticing that the previous two equations can be written as special cases of the more general formula:

$$L = \left[{N_d}^2 + (N_d + cN_o)^2\right]^{1/2} + (1-c)N_o, \tag{7.3}$$

where $c = 0$ for the Freeman formula and $c = 1$ for the Pythagorean theorem. Kimura, Kikuchi, and Yamasaki [30] proposed a compromise between the overestimation and underestimation by setting c to the average between the two techniques:



One-line axis Two-line axis

Figure 7.4: A comparison of the three methods for root length measurement. TOP: A simple root with a straight line central curve (left), and a slightly more complicated with two line segments (right). Each gray pixel is on the curve, an open circle indicates a diagonally connection between a pair of pixels, and a closed circle indicating an adjacent connection. The total number and type of connections are the same in both roots. MIDDLE: The rearranged curves by grouping similar circles (the number of circles of each type remains the same). In both roots the length is estimated as AD+DB (Freeman), AB (Pythagorean), or AE+EB (Kimura). The true length is AB (left) and AF+FB (right). BOTTOM: A plot of the results. Freeman always overestimates, Pythagorean works perfectly for the linear curve but underestimates the more complex curve, and Kimura achieves a reasonable compromise.

Method	Measured Length (pixel)	Ground Truth (pixel)	Error (%)
Img. 1, Root. 1	454.0	456.8	0.6
Img. 1, Root. 2	104.0	97.3	6.8
Img. 2, Root. 1	394.0	394.5	0.1

Table 7.1: The results of the length measurement evaluation on sample images in Figure 7.5
.

$c = \frac{1}{2}$. See Figure 7.4 for a graphical illustration and comparison of the three techniques.

7.3 Measurement Evaluation

To evaluate the performance of the presented linear feature measurement method, the measured root lengths in minirhizotron images are compared with hand-labeled ground truth. After clicking a sequence of points along the roots, the ground truth of root length is estimated as the sum of the length of line segments linking the point sequence. All the images in training set and test set are labeled by graduate students from the Horticulture Department of Clemson University. Samples of hand-labeled ground truth are shown in Figure 7.5, and the measured results are compared in Table 7.1.

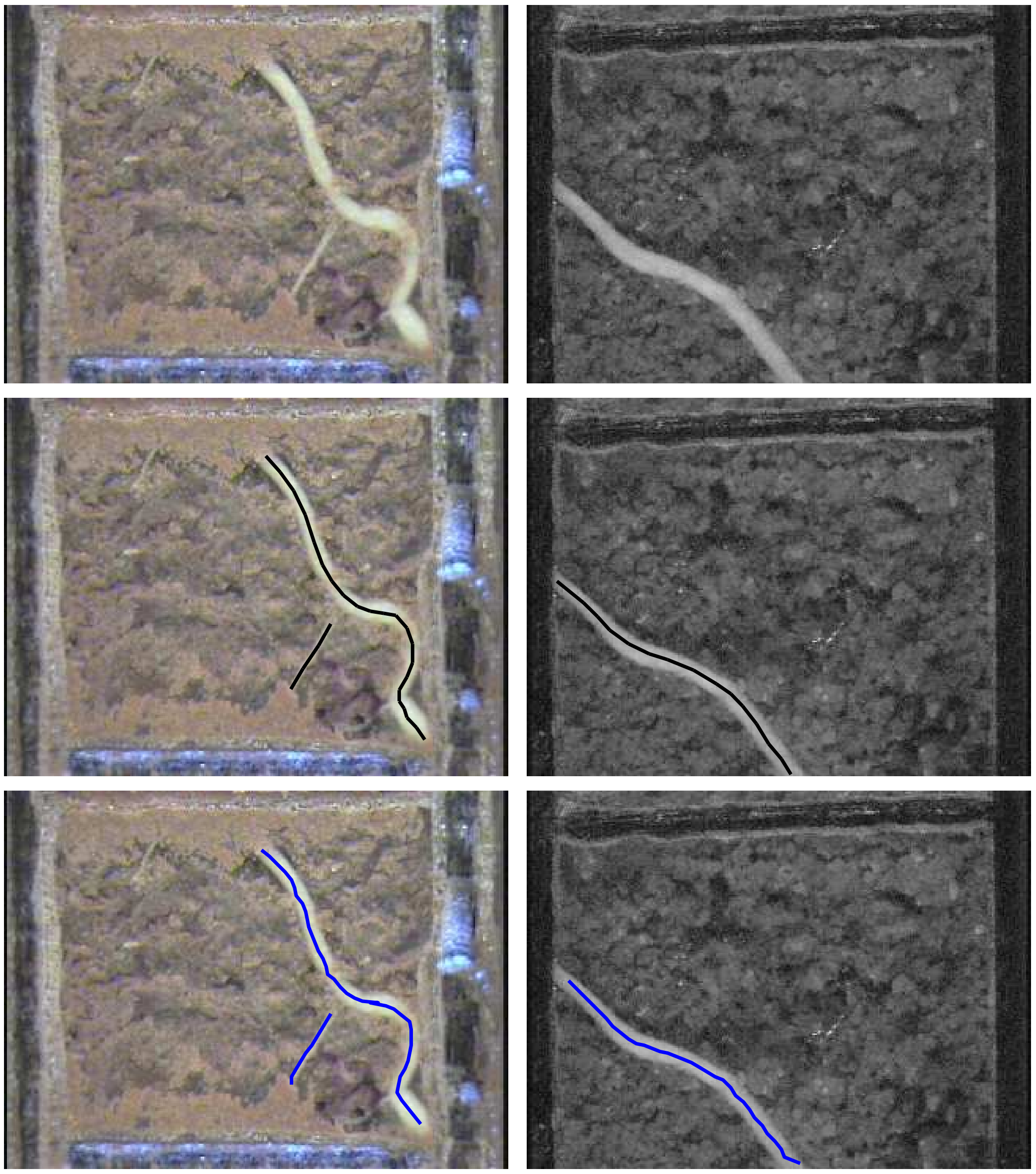

Img. 1 Img. 2

Figure 7.5: Examples of evaluating root measurement performance. *Top to bottom*: original image, hand-labeled ground truth, measured root length using the presented algorithm.

Chapter 8

Experimental Results

Our point process based linear feature extraction algorithm was developed and tested using a database of 450 minirhizotron images (640×480) from three different plant species: peach (*Prunus persica*), Freeman maple (*Acer x freemanii*), and sweetbay magnolia (*Magnolia virginiana*) [69].[1] Of these images, 200 contain no roots, while the rest contain one or more young roots of different type, size, shape, background composition, and brightness. The image backgrounds contain a wide variety of bright non-root objects including light soil particles and water droplets. We randomly selected 50 images for algorithm development and used the remaining 400 images as the test set. For ground truth, roots were labeled by hand if their diameter was greater than 0.3 mm and their length greater than 1.8 mm.

The results of the algorithm on selected peach images are shown in Figure 8.1. Multiple non-overlapping roots with curvature (#11, #36) cause no problem for the proposed algorithm, while the more difficult scenario of overlapping roots is also detected correctly (as in #82). One important challenge for root imagery is the occlusion of the roots by the soil which makes it difficult to detect the root as a

[1]The database is available at `http://www.ces.clemson.edu/~stb/research/horticulture/`.

single region using our previous work. By applying the floodfill method with a width limitation, the algorithm is able to correctly recovery the occluded region (#36, #37). For comparison, we also show the results of our previous matched filter / local entropy algorithm [67], augmented with the Adaboost discriminator trained on geometry- and intensity-based cues [68]. The previous algorithm has difficulty detecting small roots (#11), estimating the length of roots when encountering overlap (#82), and correctly handling occlusion (#36, #37).

Compared with the *peach* images, the *maple* and *magnolia* images are of considerably reduced image quality, thus presenting an even greater challenge to root detection. Nevertheless, as shown in Figure 8.2, the proposed algorithm achieves accurate results, correctly detecting roots with low luminance (#141, #149) or low contrast to the background (#129, #141). Meanwhile, water drops or bubbles (#110) are correctly ignored.

A quantitative comparison of the two algorithms on this database is shown in Table 8.2. The accuracy of the point process based algorithm ranges from 86% to 93% on all types of plants, whereas that of the previous approach achieves ranges from 60% to 92%. Moreover, the parameters for the new algorithm were the same for all experiments, whereas we manually selected different parameters for the three species for the previous algorithm. In addition, the proposed algorithm is significantly faster: 390 ms per image instead of 20 s on a 2.8 GHz Pentium 4 computer (Table 8.1).

For completeness, we show some examples in which the proposed algorithm fails in Figure 8.3. Roots with dark background noise(#35, #158), bright background noise (#114, #155), or excessive curvature in a root (#155) can confuse the algorithm and lead to erroneous results. Moreover, the algorithm tends to underestimate root length, as seen in Figure 8.1.

The accuracy of the Freeman, Pythagorean, and Kimura-Kikuchi-Yamasaki

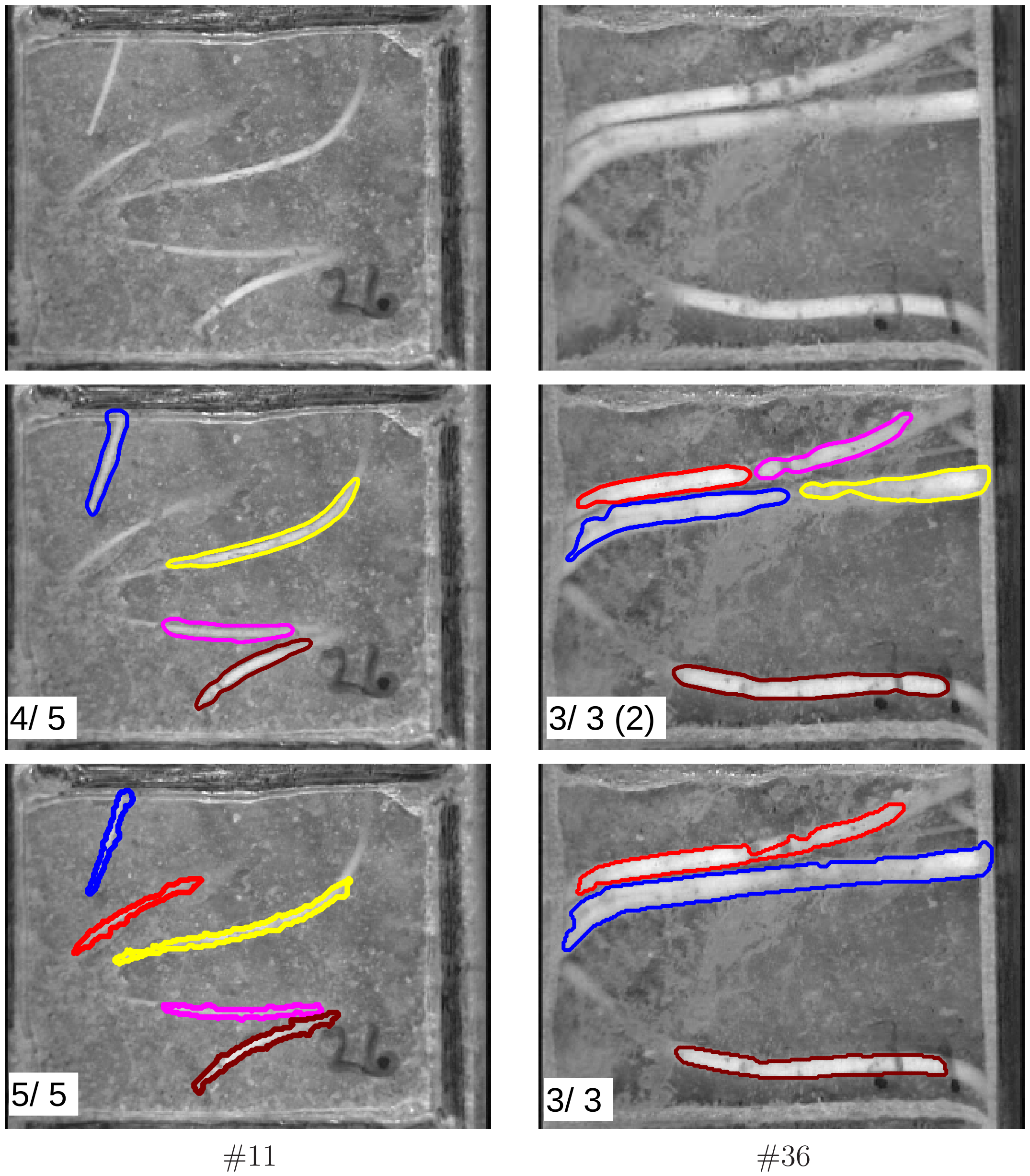


Figure 8.1: The results of our previous algorithm [67, 68] (*middle row*) and the proposed point process based algorithm (*bottom row*) on some images from *peach*. Overlaid on each image is the number of roots successfully detected (at least 50% of the region found) and the total number of roots in the image, along with the number of false positives (if any) in parentheses. The number below each image identifies it in the database.

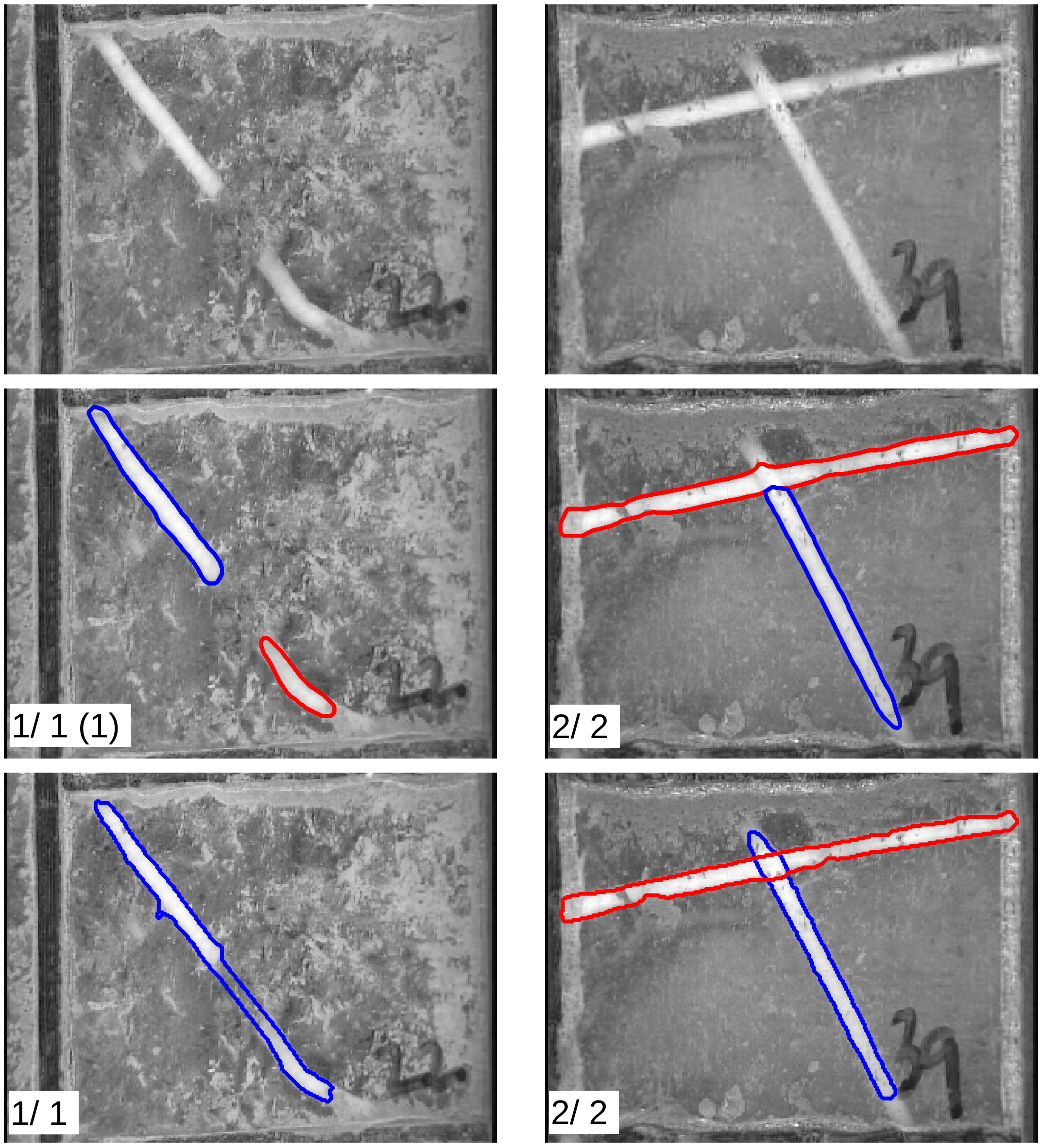
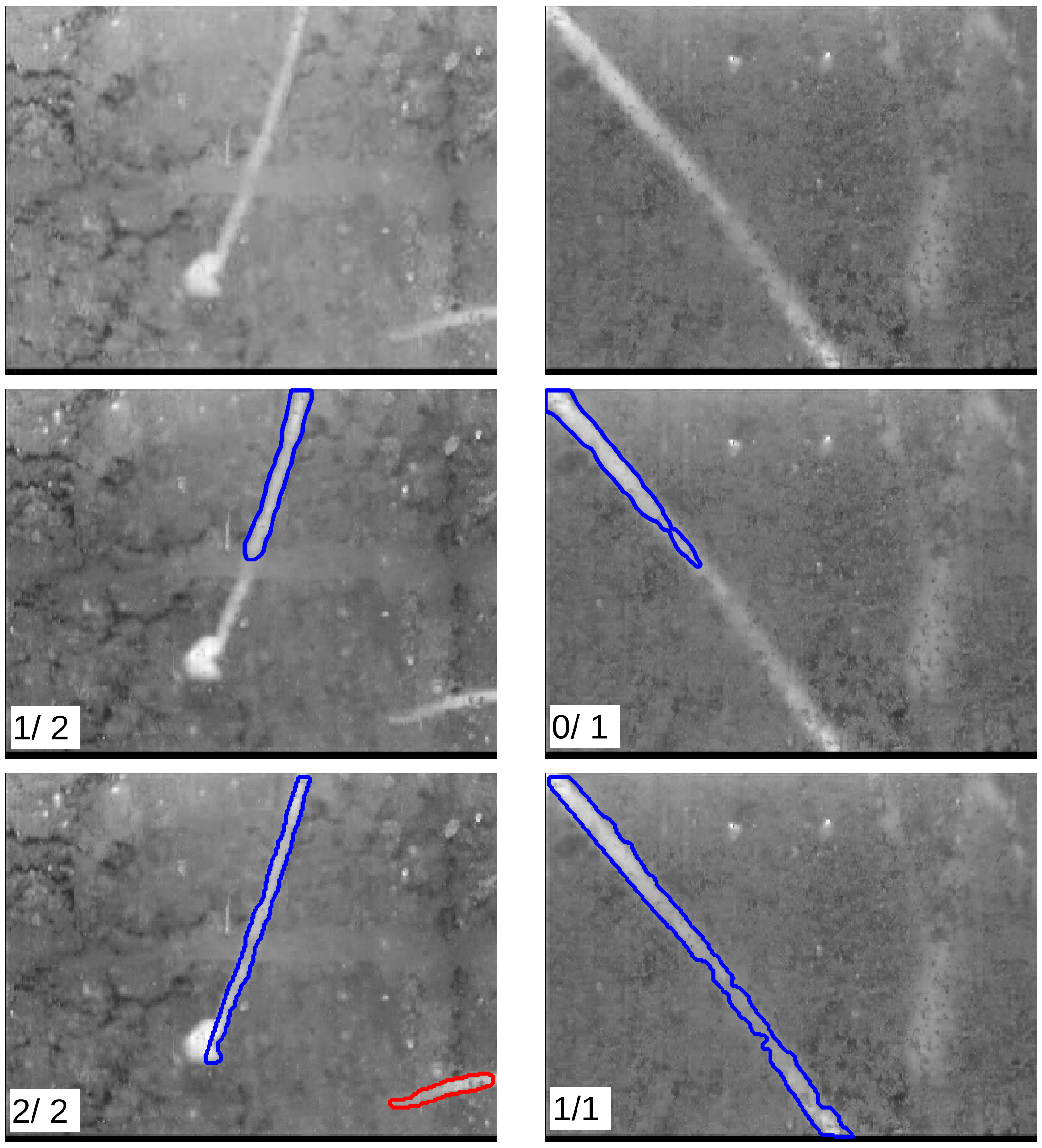


Figure 8.1: The results of our previous algorithm [67, 68] (*middle row*) and the proposed point process based algorithm (*bottom row*) on some images from *peach*. Overlaid on each image is the number of roots successfully detected (at least 50% of the region found) and the total number of roots in the image, along with the number of false positives (if any). The number below each image identifies it in the database. (cont.)



#110 #129

Figure 8.2: The results of our previous algorithm [67, 68] (*middle row*) and the proposed point process based algorithm (*bottom row*) on some images from the more difficult *maple* and *magnolia* databases.

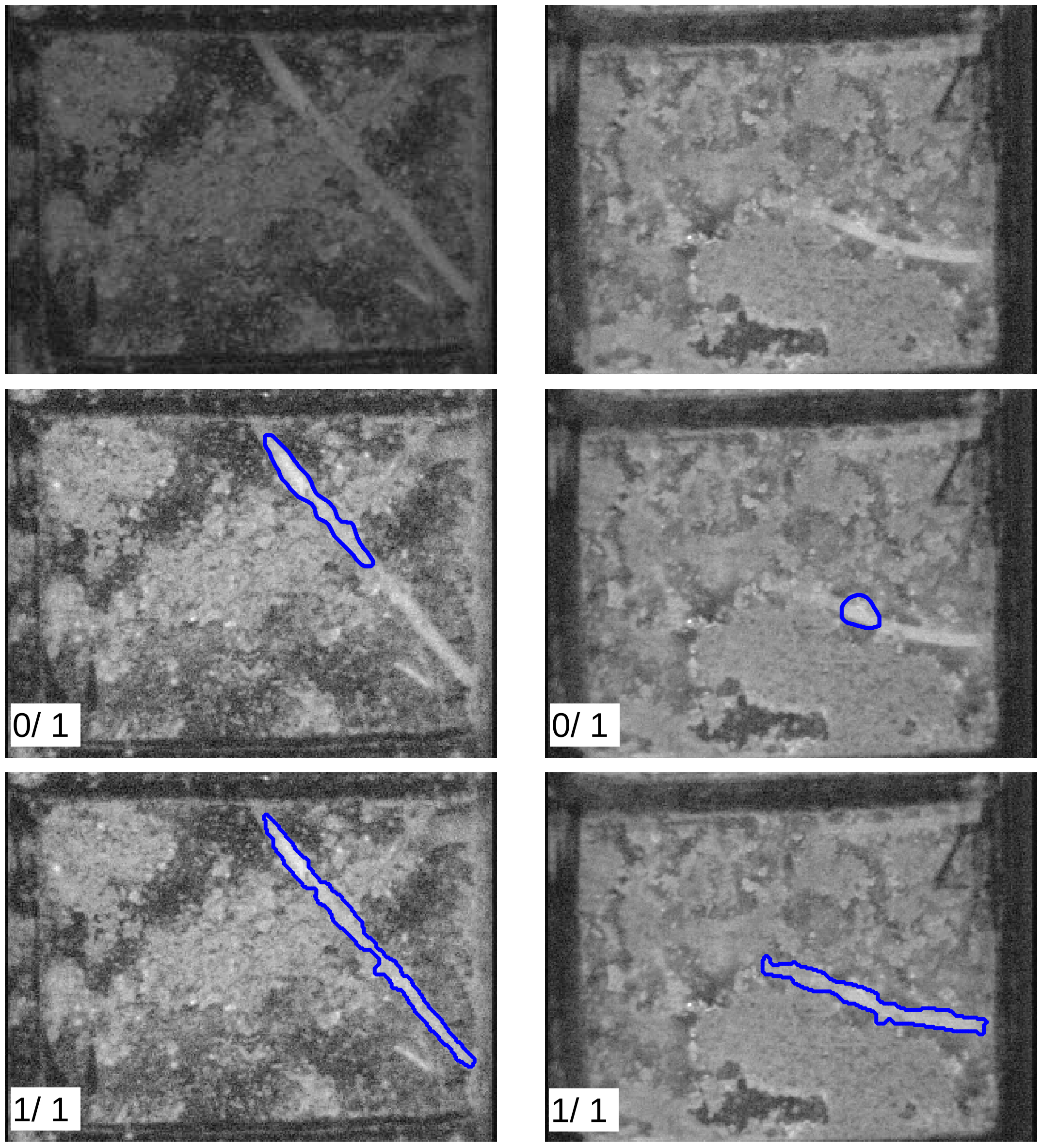
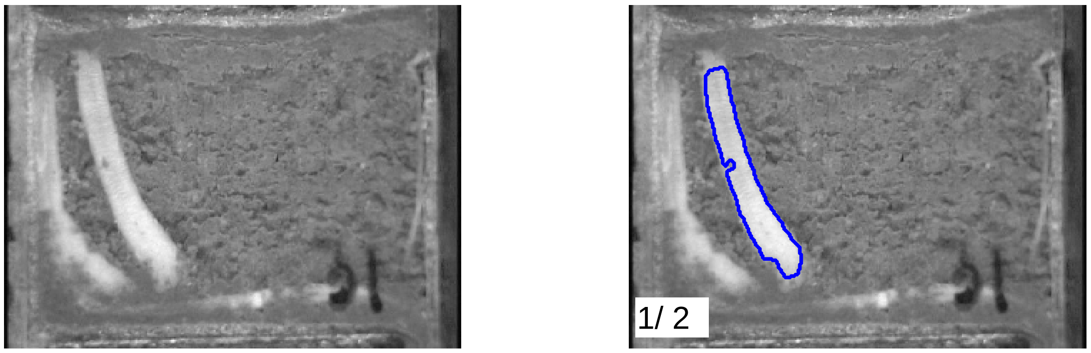
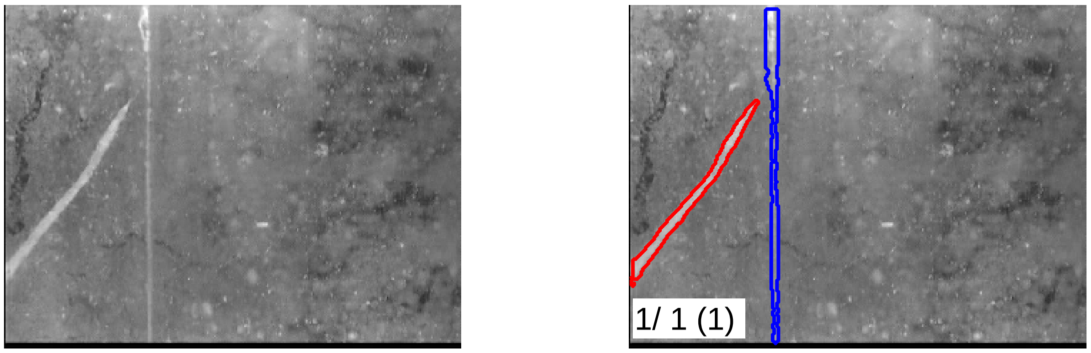


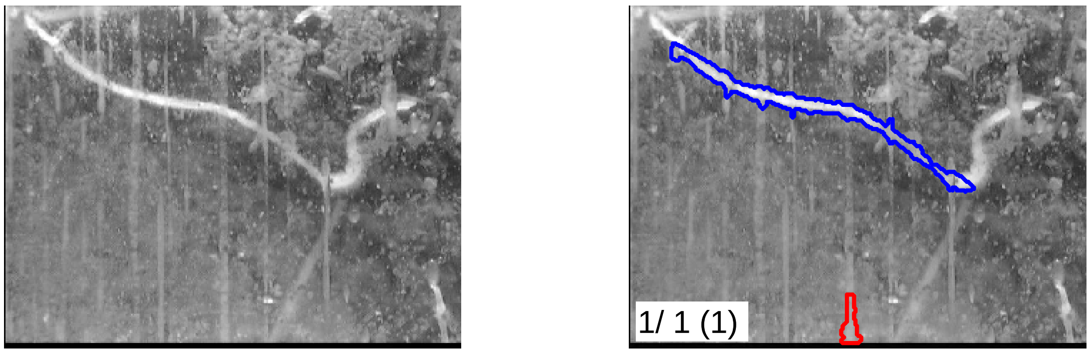
Figure 8.2: The results of our previous algorithm [67, 68] (*middle row*) and the proposed point process based algorithm (*bottom row*) on some images from the more difficult *maple* and *magnolia* databases. (cont.)



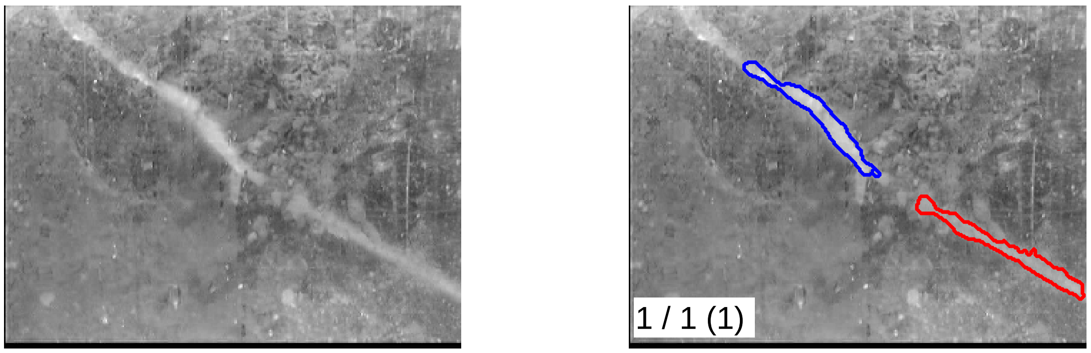
#35



#114



#155



#158

Figure 8.3: Examples of detection or measurement errors on difficult images.

Point Process Based Algorithm [69]	Time (s)	Template Based Method [67]	Time (s)
Seed Point Selection	0.047	Matched Filtering	15
Centerline Detection	0.031	Local Entropy Thresholding	2
Centerline Validation	0.141	Linear Feature Labeling	0.8
Centerline Tracing	0.026	Linear Feature Discriminating	2.1
Region Detection	0.125	Other	0.02
Other	0.02		
Total	0.39	Total	19.92

Table 8.1: Computation time of the point process based algorithm and the template based method.

Root type	Peach		Maple		Magnolia	
	TPR	FPR	TPR	FPR	TPR	FPR
Zeng et al. [67, 68]	92%	5%	60%	7%	68%	9%
Point process based	93%	6%	89%	8%	86%	6%

Table 8.2: The true positive rate (TPR) and false positive rate (FPR) of the two algorithms on three different types of roots.

methods for estimating root length are compared in Table 8.3. Roots are detected from the test set images using our template matching based algorithm. As described in chapter 7, the Freeman formula generally overestimates the length of a curve, obtaining an average error of 9.8%. The Pythagorean theorem performs slightly better with a 8.1% average error, but it is significantly affected by images with a non-straight central curve. Overall, Kimura-Kikuchi-Yamasaki's method is the most accurate, with an average error of 6.5%.

Method	Measurement error (%)		
	Average	Min	Max
Freeman formula	9.8	0.5	28.3
Pythagorean theorem	8.1	0.6	26.3
Kimura's method	6.5	0.1	23.1

Table 8.3: Length measurement errors using the three different methods.

In Chapter 6, we built strong classifiers for three plant species separately.

When applied to the test set image, their TPRs ranged from 89% to 94%, and FPRs ranged from 3% to 7% (Table 8.4). These rates compare favorably with those obtained on the training images.

Each classifier was tested not only on the species for which it had been trained, but also on the two additional plant species. The application of a classifier from one species to test images from another species did not necessarily reduce its accuracy (Table 8.4). In several cases, accuracy was unchanged (for example, peach images evaluated with the magnolia classifier). In cases where accuracy was reduced, the magnitude of the reduction was small. Overall, the maple classifier produced the lowest FPRs. No single classifier was associated with the highest TPRs.

	Training image species					
	Peach		Magnolia		Maple	
Test image species	TPR	FPR	TPR	FPR	TPR	FPR
Peach	0.89	0.03	0.89	0.04	0.90	0.04
Magnolia	0.88	0.10	0.94	0.06	0.94	0.06
Maple	0.95	0.12	0.95	0.12	0.93	0.07

Table 8.4: True positive rate (TPR) and false positive rate (FPR) of strong root classifiers applied to test set images from three woody plant species (peach, magnolia and maple).

The strong classifier tended to fail when presented with non-root objects that had the geometric characteristics of roots (#10, #69). An example of such a false positive result is shown in Figure 8.4. Also, false negative results were obtained when roots were partially occluded by soil, altering their apparent intensity and geometric properties ((#92, #95)). Such errors reduced the TPR of the algorithm. Under real-world conditions, a small percentage of incorrect results may be unavoidable, and an automated image processing system will need to provide a simple mechanism for verification and correction of measurements.

In Chapter 5, we introduced four linear features discriminators for minimizing

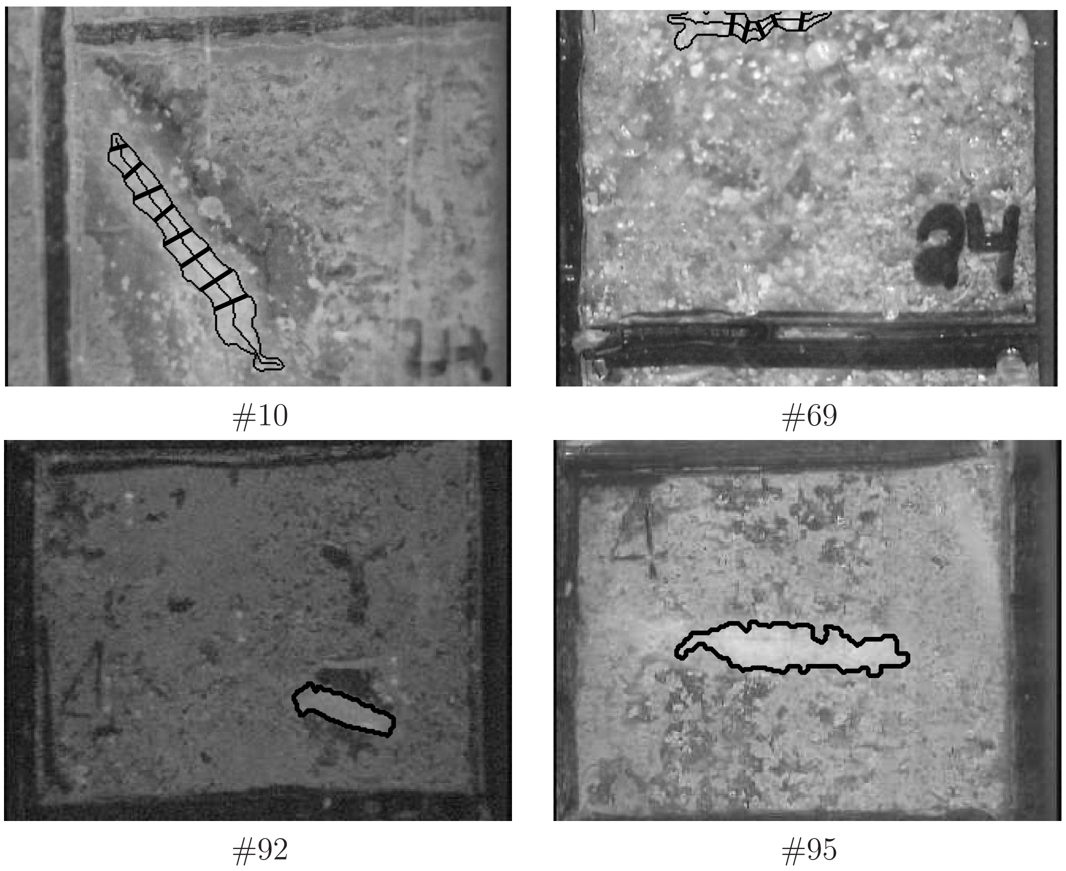


Figure 8.4: Examples of false positive detection (*top*) and false negative detection (*bottom*).

the proposed model energy. Parameters of the four off-line trained linear discriminators are shown below in Table 8.5. Other parameters we used for linear feature tracing and linear feature region detection are shown in Table 8.6.

Parameters	v_1	v_2	v_T
Seed point selection discriminator u_a	0.3	74.9	-82.4
Centerline fitting discriminator u_b	0.5	21.8	-176.0
Centerline validation discriminator u_c	8.5	8.8	-16.6
Centerline connection discriminator u_d	1.8	2.8	-1.2

Table 8.5: Parameters of the four linear discriminators we used for model energy minimization.

Method	Centerline Tracing			Region Detectio			
Parameter	β	N^T	$\Delta {g_5}^T$	γ	τ_λ	τ_φ	τ_r
Value	5	50	1.25	0.5	0.7	80	50

Table 8.6: Parameters used for centerline tracing and region detections.

In Chapter 6, we implemented the Adaboost algorithm for off-line training a strong classifier to discriminate linear feature from unwanted background objects based on five weak feature classifiers. The weights of the five weak classifiers in each of the Adaboost trained strong classifiers for peach, magnolia and maple are shown in Table 8.7.

Classifier	Peach	Magnolia	Maple
Histogram Distribution	0.00	0.55	0.44
Interior Intensity Edges	0.16	0.83	0.28
Eccentricity	1.10	0.68	0.87
Approximate Line Symmetry	0.56	1.00	0.90
Boundary Parallelism	0.78	0.94	0.45

Table 8.7: The weights of the five weak classifiers in each of the Adaboost trained strong classifiers for peach, magnolia and maple.

Chapter 9

Other Applications

There are many potential applications of the presented algorithm which we intend to explore. In this chapter, we describe the application of the algorithm to other two types of linear feature detection: urban road detection in satellite images and blood vessel stenosis estimation in Digital Subtraction Angiography (DSA) images.

9.1 Urban Road Detection

Digital road information is required for a variety of applications ranging from provision of basic topographic infrastructure over transportation planning, traffic and fleet management and optimization, car navigation systems, location-based services, tourism, to web-based emergency response applications and virtual environments. Unsupervised extraction of roads from satellite imagery eliminates the need for human operators to perform the time consuming and expensive process of mapping roads from aerial photographs. As increasing volumes of high spatial resolution satellite images (e.g., Ikonos, QuickBird, OrbView-3, etc.) become available, many of them have never even been viewed [34]. What is urgently needed is the automation for

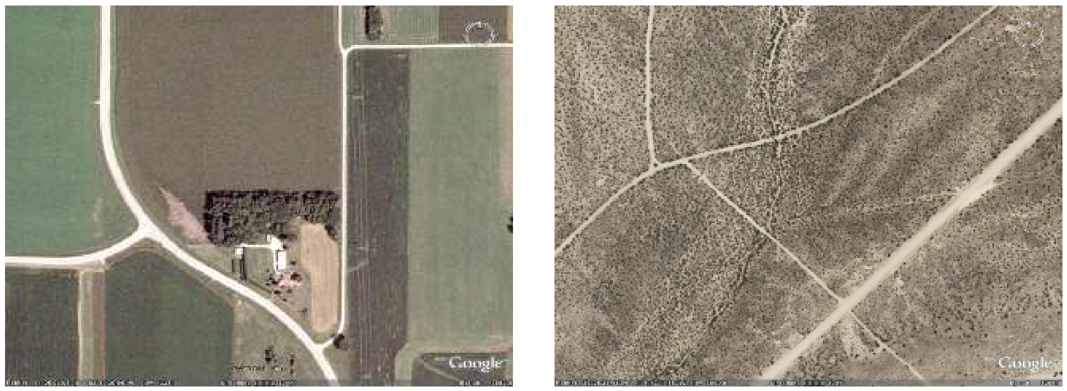


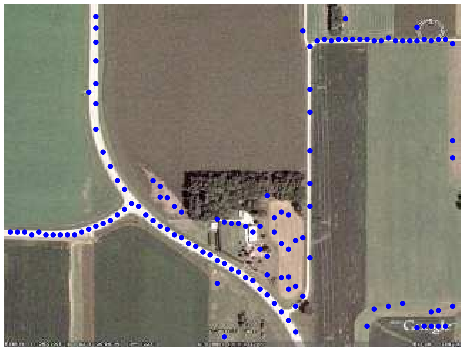
Figure 9.1: Sample satellite images obtained from Google Earth.

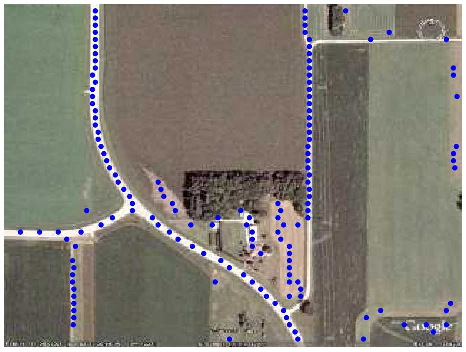
extracting information and analyzing image content. Here, the proposed algorithm is applied for automatic urban road extraction in satellite images. The test images extracted from Google Earth contain urban roads in different backgrounds and with different widths and orientations.

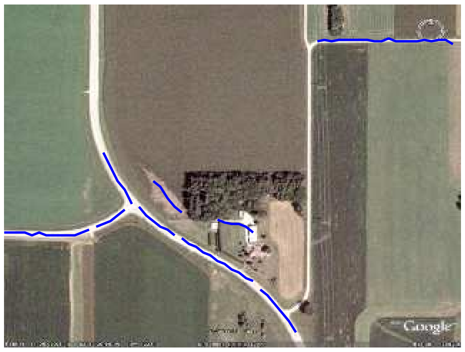
Adopting the previously developed energy model in Chapter 3, the road network is extracted by minimizing the energy through the steps introduced in Chapter 5. After extracting seed points from local maxima from two directions and three scales, centerline segments are fitted to the seed points. Unlike the plant roots in minirhizotron images, many bright linear objects can be found from the background of satellite images such as rectangular roofs, parking lots, and cropland (Figure 9.2.e). To remove the line segments from these background linear features, a greedy linear discriminator $\mathbf{u_c}$ is applied for centerline segment validation after measuring the features *support* g_3 and *width stability* g_4. Segments for which $[g_3(s_i) \quad g_4(s_i) \quad 1]\mathbf{u_c} \geq \mathbf{0}$ are retained as centerline, while the other segments are discarded. Once centerline segments from background features are removed, valid centerline segments are traced by adding new points along its direction if the points do not significantly increase the *smoothness* of the centerline segments. Here, the *smoothness* of the centerline segment is calculated as the standard deviation of the matched filter response (MFR) value along the centerline segment as we introduced in Chapter 5. The results of road extraction from satellite images of a crop field and a desert area are shown in Figure 9.2 and Figure 9.3 respectively.

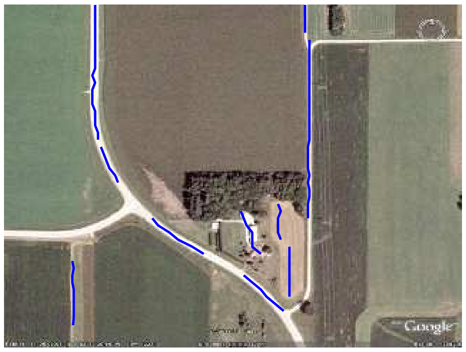
9.2 Stenosis Estimation

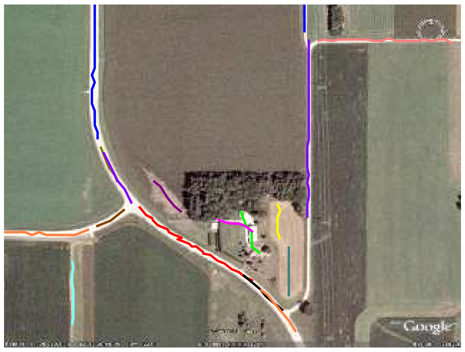
A stenosis is an abnormal narrowing in a blood vessel or other tubular organ or structure. Several surveillance techniques may be used in the timely detection of stenosis development such as color Doppler ultrasonography and Digital subtraction angiography (DSA). In DSA, an X-ray contrast agent (such as an iodine compound) is injected so as to increase the density (attenuation coefficient) of the blood within a certain organ or system of interest. A number of X-ray images are taken as the contrast agent spread throughout the arterial network and before the agent is dispersed via circulation throughout the body. An image taken before the injection of the agent is used as the "mask" of reference image, and subtraction from the "live" images obtained with the agent in the system to obtain enhanced images of the arterial system of interest [48]. DSA is widely used in the diagnosis and treatment of coronary arterial diseases such as arterial stenosis. In this section, the proposed method is modified to extract blood vessels in DSA images for stenosis estimation and applied to two sample images shown in Figure 9.4. Figure 9.4.a is the DSA image of a Shelley's carotid anthropomorphic vascular phantom containing symmetric 70% diameter stenosis [56]. It can be applied to evaluate a stenosis estimation algorithm by comparing the measured stenosis with the ground truth value (70%). Figure 9.4.b is a sample DSA image from real clinical data containing carotid stenosis.


(a) Detected seed points (Vertical)


(b) Detected seed points (Horizontal)


(c) Fitted segments (Vertical)


(d) Fitted segments (Horizontal)


(e) Connected centerline segments

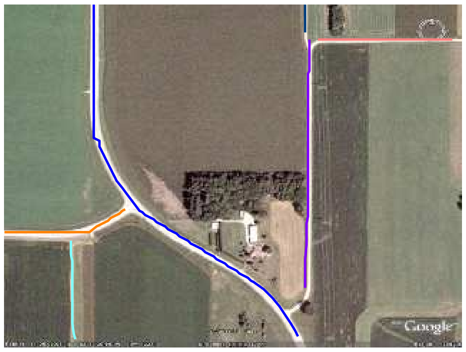
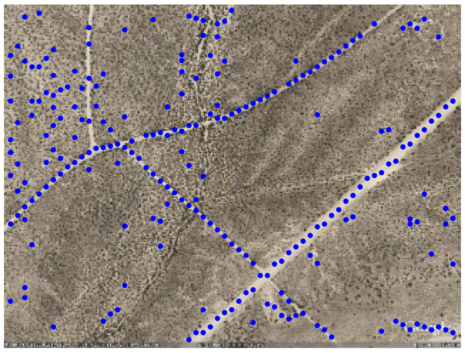
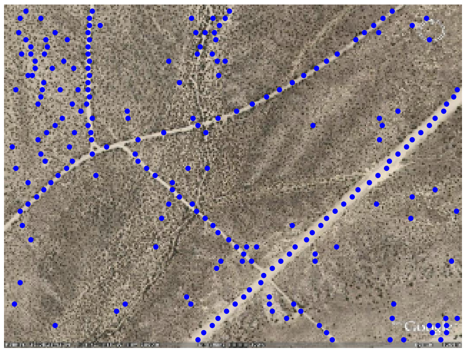
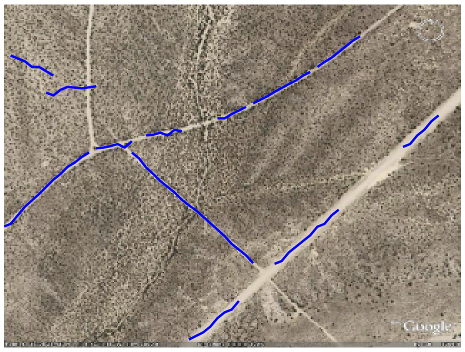
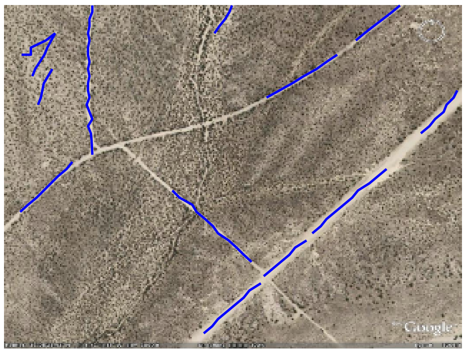

(f) Combined centerline segments

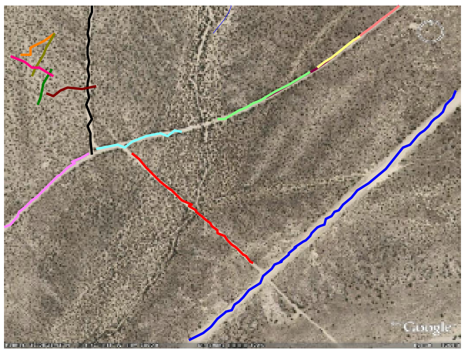
Figure 9.2: The results of applying the proposed method for road extraction from a satellite image of a crop field.


(a) Detected seed points (Vertical)


(b) Detected seed points (Horizontal)


(c) Fitted segments (Vertical)


(d) Fitted segments (Horizontal)


(e) Connected centerline segments

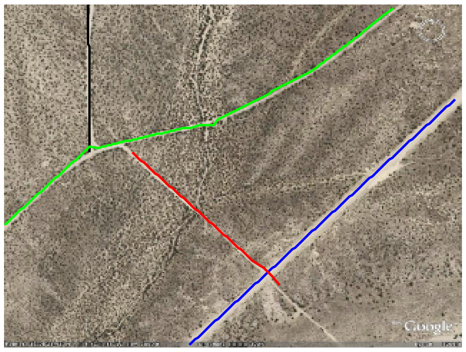
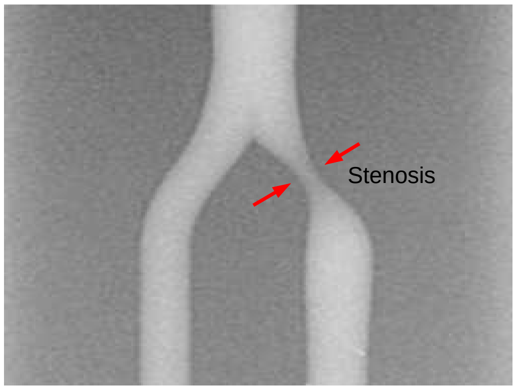
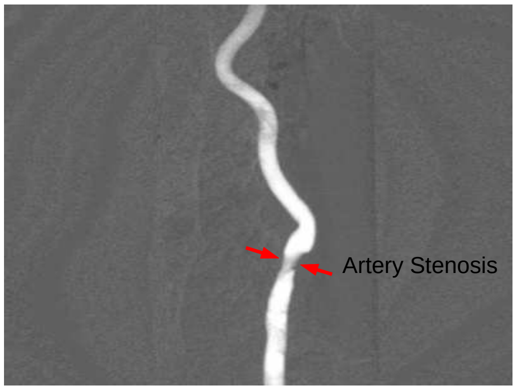

(f) Combined centerline segments

Figure 9.3: The results of applying the proposed method for vessel extraction from a satellite image of a desert area.



(a)



(b)

Figure 9.4: A vascular phantom image contains symmetric 70% diameter stenosis (*left*) and a DSA image from real clinical data containing carotid stenosis (*right*).

9.2.1 Energy Model

Using the Gibbs point process we introduced in Chapter 3, a probabilistic model is built for vessel network in DSA images, and then an energy minimization method is applied for vessel network extraction. Unlike other linear feature extraction problems such as root and road network detection which need to detect and measure each linear feature in the input image, the stenosis estimation problem focus on searching vessels with significant width variation. Therefore, several modifications are made to the previously developed energy model. Here, the intensity energy of a centerline segment s_i is expressed as the linear combination of four features:

$$U_D^{dsa}(s_i) = \sum_{j=1}^{N_D^{dsa}} w_j g_j(s_i), \tag{9.1}$$

where $g_1(s_i)$ and $g_2(s_i)$ are respectively *support* and *residue* of s_i as we defined early in Chapter 3.

The definition of another feature *width stability*, $g_3(s_i)$, we use here is slightly different from our previous definition for root and road detection. After applying

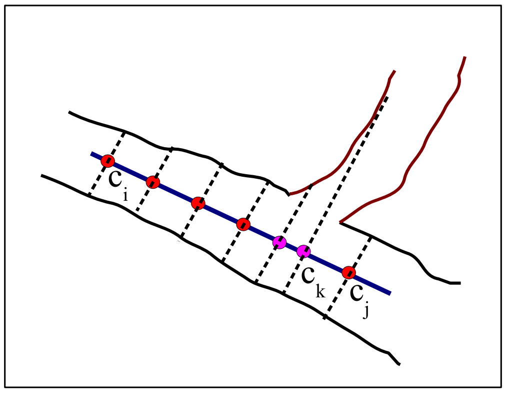


Figure 9.5: An illustration of calculating width stability of a centerline segment. Good points (*red*) are used for width stability calculation, while the unbalanced points (*magenta*) are removed.

Canny edge detector ($\sigma = 3.0$) to the original gray level image, and then removing the spurs, we acquire the edge map of the image by labeling each remained edge. For each point c_k on a centerline segment s_i, we search along the line perpendicular to the centerline segment at c_k to find its two nearest points B_k^{left} and B_k^{right} that intersect the boundary edge at each side, their corresponding labels L_k^{left} and L_k^{right} in the edge map are also stored. Since we simply assume that each vessel has a pair of continuous boundary at each side, we measure g_3 as the ratio of points meet all the following requirements to the total number of points on s_i. Meanwhile, any points along the centerline segment that can not meet the requirements are removed.

(a). Each point c_k on s_i can find a pair of boundary points.

(b). The distances from the point c_k to its boundary points are less than half of the breadth of c_k.

(c). The found boundary points at each side in the edge map have the same label.

An illustration of calculating the width stability of a centerline segment is shown in Figure 9.5.

Because most of the blood vessels in DSA images represent stronger intensity contrast to the background, instead of defining the fourth feature *smoothness* g_4 as the standard deviation of the matched filter response values along the points of s_i, we simply define it as the standard deviation of measured edge magnitudes.

To estimate the interaction of two attracted centerline segments s_i and s_j, their interaction energy $U_I^{dsa}(s_i, s_j)$ is measured as

$$U_I^{dsa}(s_i, s_j) = h(s_i, s_j), \tag{9.2}$$

where the feature $h(s_i, s_j)$ measures the residue of the new line built using all the points of s_i and s_j.

9.2.2 Energy Minimization

Energy of the vessel model is minimized using a greedy algorithm involves five steps: seed points selection, centerline fitting, validation, combination and tracing. The parameters used in these steps are learned from the data.

After local maxima are selected from the input DSA image at each N^{th} column and row from two directions and three scales, their height e and breadth b are measured, and then applied to the linear discriminator $\mathbf{u_a}$ we learned in Chapter 5. Points for which $[b \quad e \quad 1]\mathbf{u_a} \geq \mathbf{0}$ are retained as *seed points*, which the other points are discarded.

Once the seed points are decided, centerline segments are fitted to the seed points using the same manner we introduced in Chapter 5. Sometimes this procedure may detect the centerline of bright artifacts in the background in addition to the centerlines of actual vessels. On the other hand, because of the vessel bifurcation, the fitted line segments may include seed points belong to different vessels. Therefore, a validation step is performed firstly to remove the false positives. For each vessel centerline segment, the probability $P(D_{valid} \mid s) = \exp\{-\psi(D_{valid} \mid s)\}$ is computed, where the energy function $\psi(D_{valid} \mid s)$ is based on the width stability $g_3(s_i)$. Only centerline segments with $\psi(D_{valid} \mid s)$ larger than the predefined threshold $\psi_T(D_{valid})$ are remained. Meanwhile, the interaction of any attracted centerline pairs $< s_i, s_j >$ are measured by $U_I(s_i, s_j)$. The interaction energy is minimized by replacing s_i and s_j with a new line consists of all points belongs to s_i and s_j if $h(s_i, s_j) \leq g_2(s_i) + g_2(s_j)$.

To detect the complete centerline of a vessel with low contrast to the background, we trace the centerline along its current direction s^k from its end point q^k. New points $q^{k+1}(= q^k + \beta\mathbf{v})$ are added to the centerline if we can find a pair of its boundary points $(B^{k+1}_{left}, B^{k+1}_{right})$ along the direction perpendicular to s^k at each side whose (1) edge magnitudes cause no significant increase to the *smoothness* value g_4 of the centerline and (2) distances to q^{k+1} are similar. The search direction s^{k+1} is updated at each step. Meanwhile, we calculate the width of the vessel w^{k+1} at the new position q^{k+1} as $\| B^{k+1}_{left} - B^{k+1}_{right} \|$. The tracing procedure is stopped if any of the following criteria is met:

(a). The new point q^{k+1} is out of the image boundary.

(b). The extended centerline intersects with another previously detected centerline.

(c). The variation of vessel width converge ($| w^{k_1} - w^k | \leq w_T$).

(d). The iteration number reaches the pre-set limit N_T.

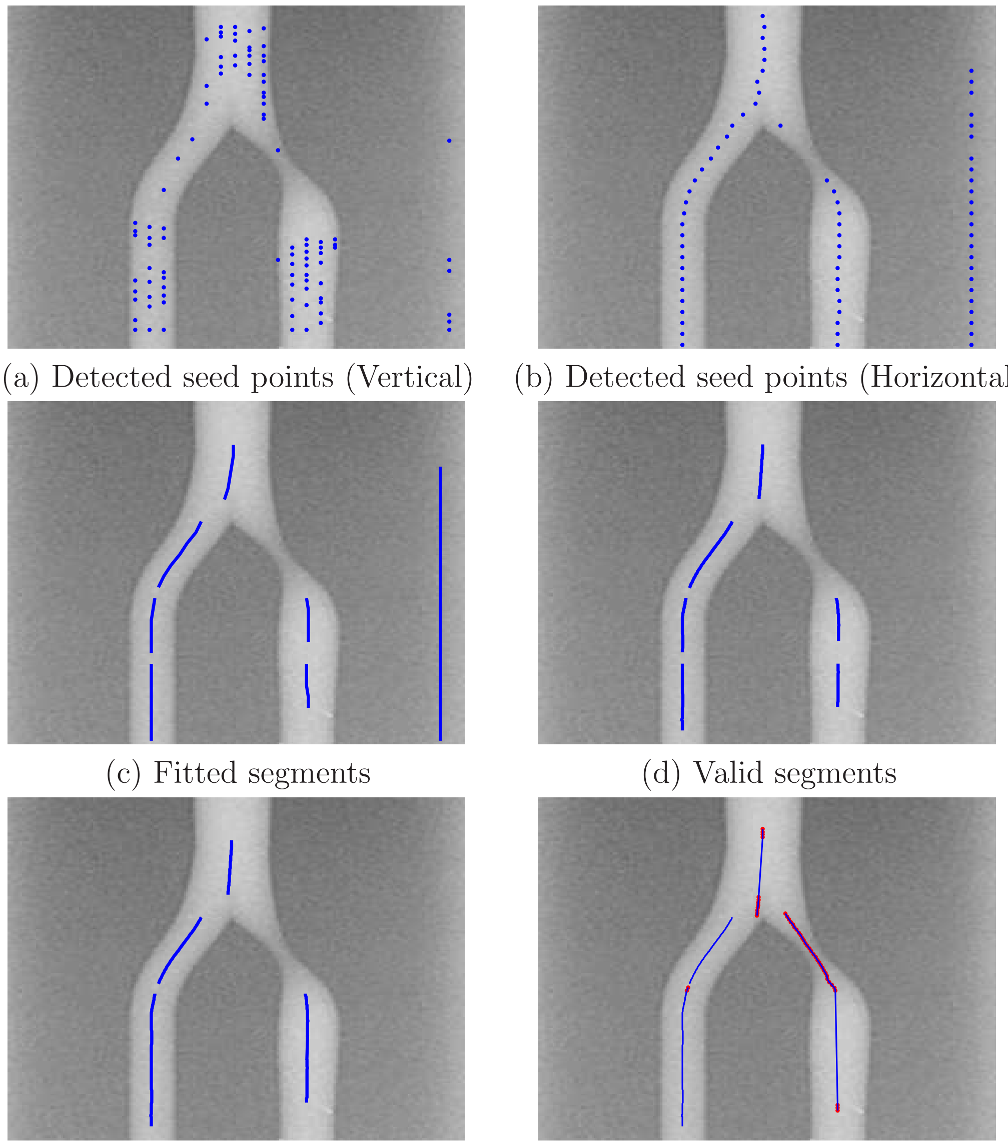
The results of centerline extraction in the phantom image and the sample image from clinical data in Figure 9.4 are shown below. As we can see in Figure 9.6 and Figure 9.7, after seed points (*blue dot*) are detected from two directions, centerline segments are fitted (*blue line*). The false positive line segments are removed by the validation step. Once oversegmented centerline segments are combined by comparing their similarity, new points (*red dot*) are traced from its two end points to detect the complete centerline. In Figure 9.7.f, although there are no seed points found at the stenosis area, the presented method can successfully trace the centerline to the stenosis area.

9.3 Experimental Results

To evaluate the performance of our algorithm on urban road detection, a test set containing 20 images (640×480) is built by selecting satellite images from Google Earth. The test images contain roads from different geological conditions. Among the 20 images, the overall detection rate is 92%. As we can see in Figure 9.8, multiple roads with intersection and curvatures are extracted correctly (#1, #2). Roads from different background cause no problem (#2, #3). Meanwhile, roads occluded by the shadow of trees (#3) and clouds (#4) are also detected successfully.

To evaluate the performance of our algorithm on blood vessel extraction, we built a test set containing 20 DSA images of renal artery (#1, #4), carotid artery (#3) and cerebral artery (#2) selected from clinical data. After selecting a region of interest in an input DSA image, vessel segments in the ROI are extracted. The detected vessel centerline segment on sample images are shown in Figure 9.9.

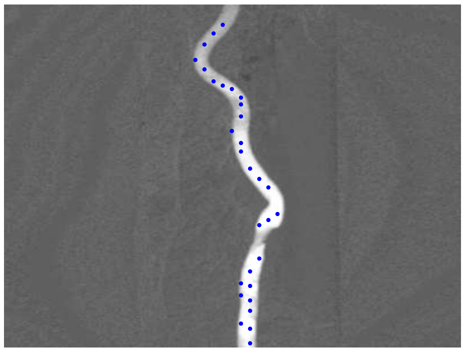
In our experiments, we use the same set of parameters as shown in Table 8.5 and Table 8.6 for urban road detection and discrimination except a new centerline validation discriminator $\mathbf{u_c} = [8.53 \quad 8.78 \quad -18.72]$, while the parameters we used for blood vessel extraction in DSA images are little different and shown below in Table 9.1 and Table 9.2.

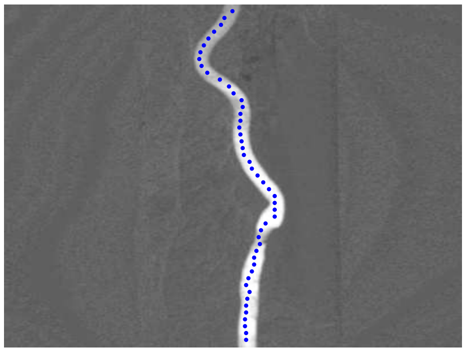

(a) Detected seed points (Vertical) (b) Detected seed points (Horizontal)

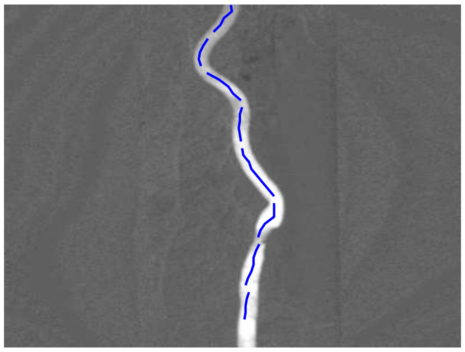
(c) Fitted segments (d) Valid segments

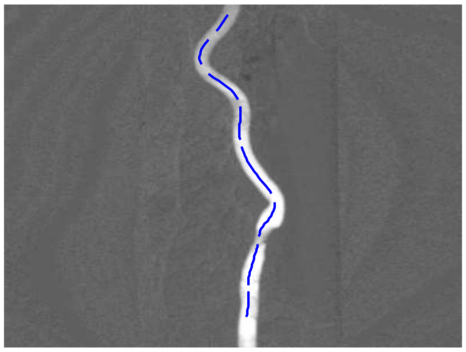
(e) Combined centerline segments (f) Traced centerline segments

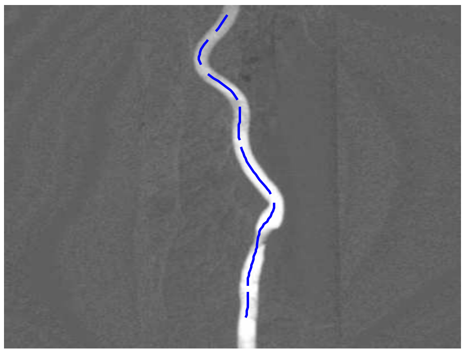
Figure 9.6: The results of applying the proposed method for vessel extraction in a phantom image.


(a) Detected seed points (Vertical)


(b) Detected seed points (Horizontal)


(c) Fitted centerline segments


(d) Valid centerline segments


(e) Combined centerline segments

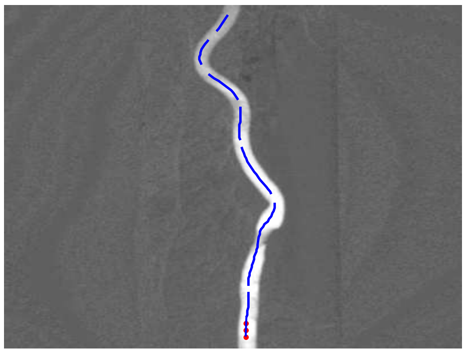

(f) Traced centerline segments

Figure 9.7: The results of applying the proposed method for vessel extraction in an image from clinical data.

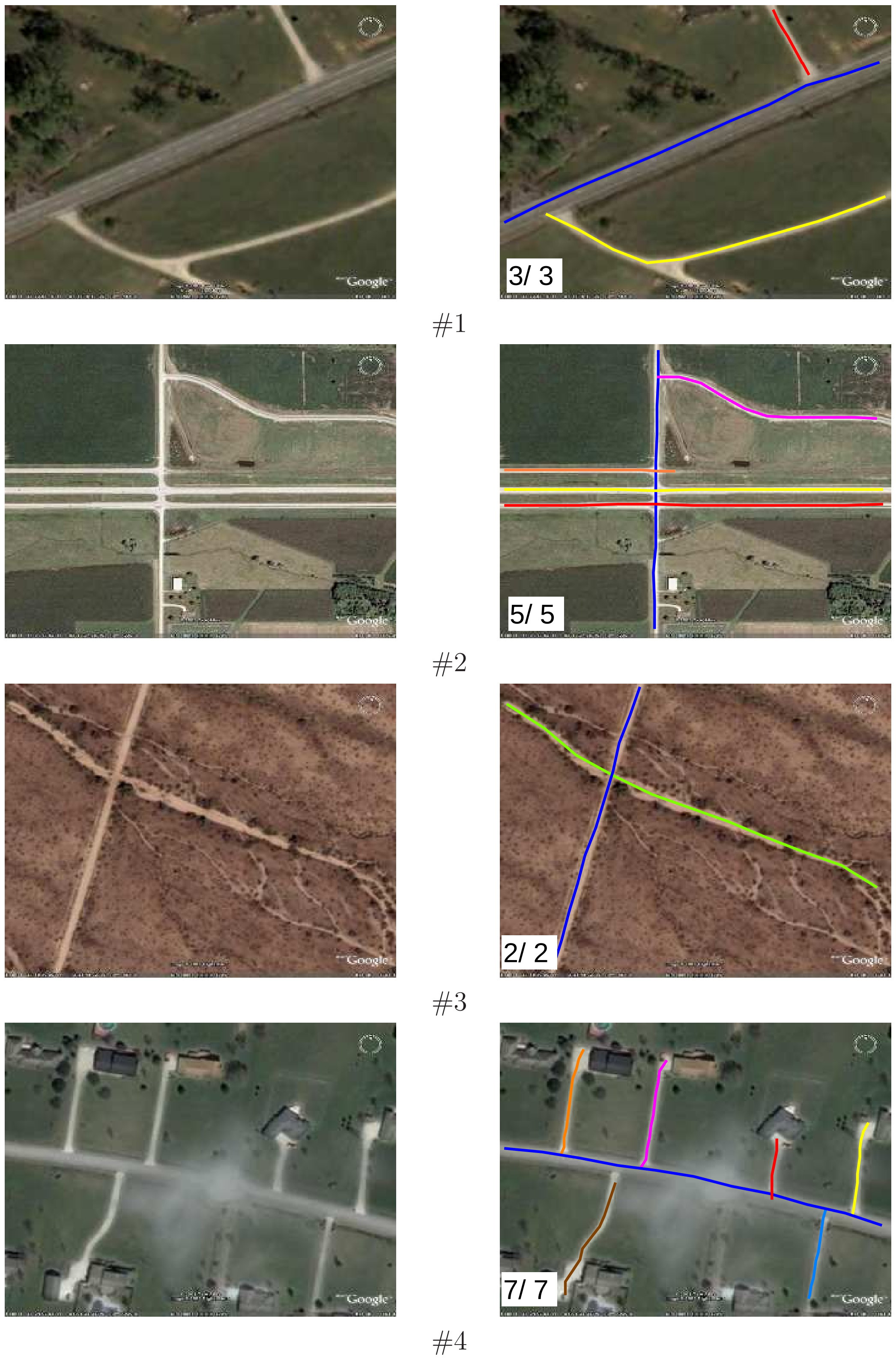
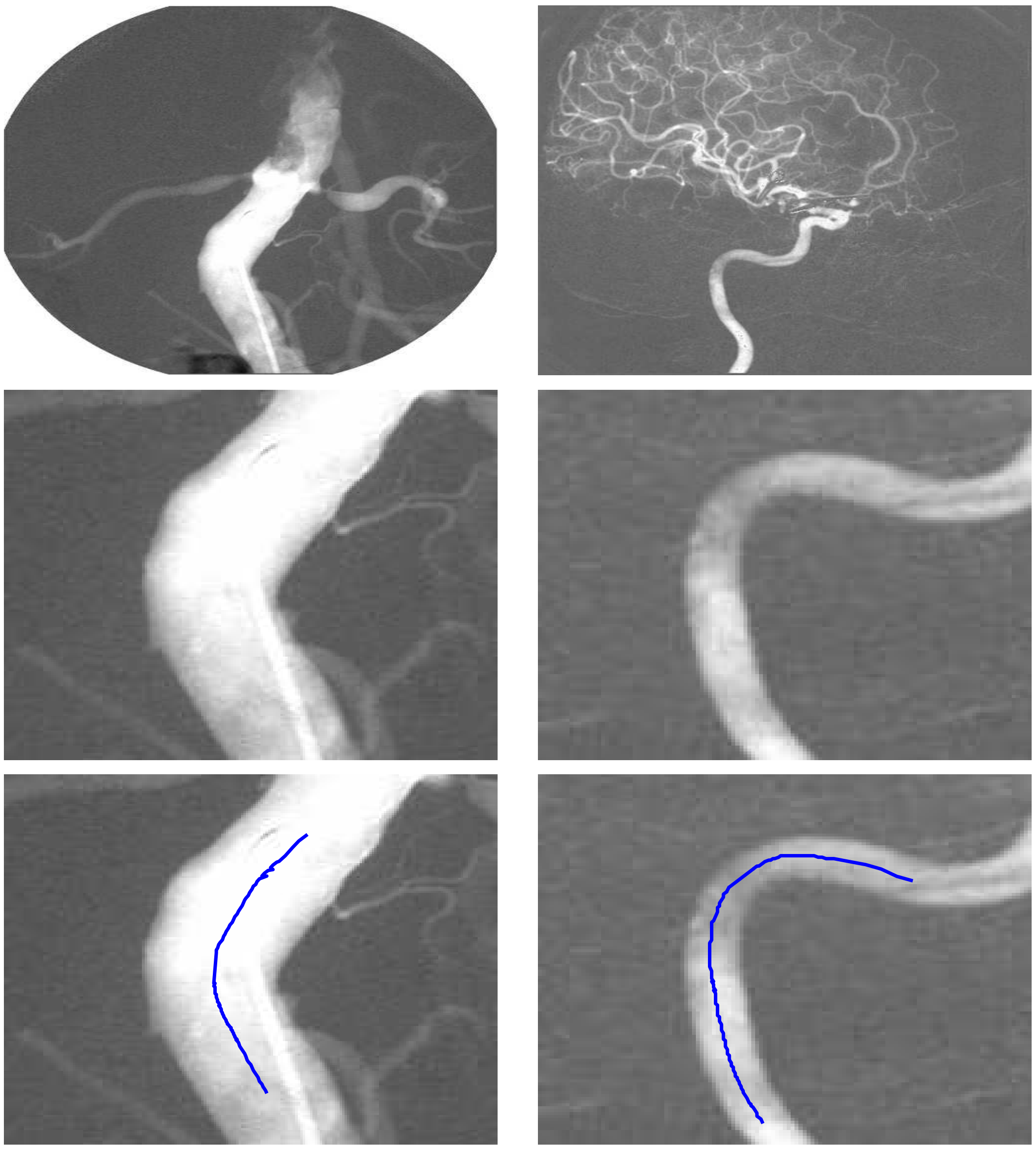
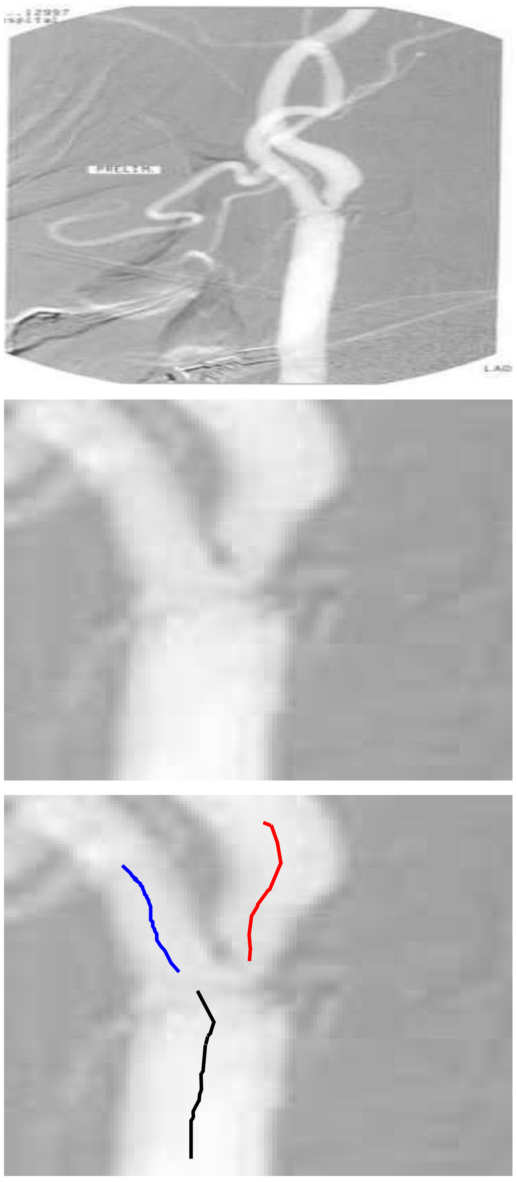


Figure 9.8: The results of urban road detection in sample satellite images. Centerlines of different roads are marked with different colors.

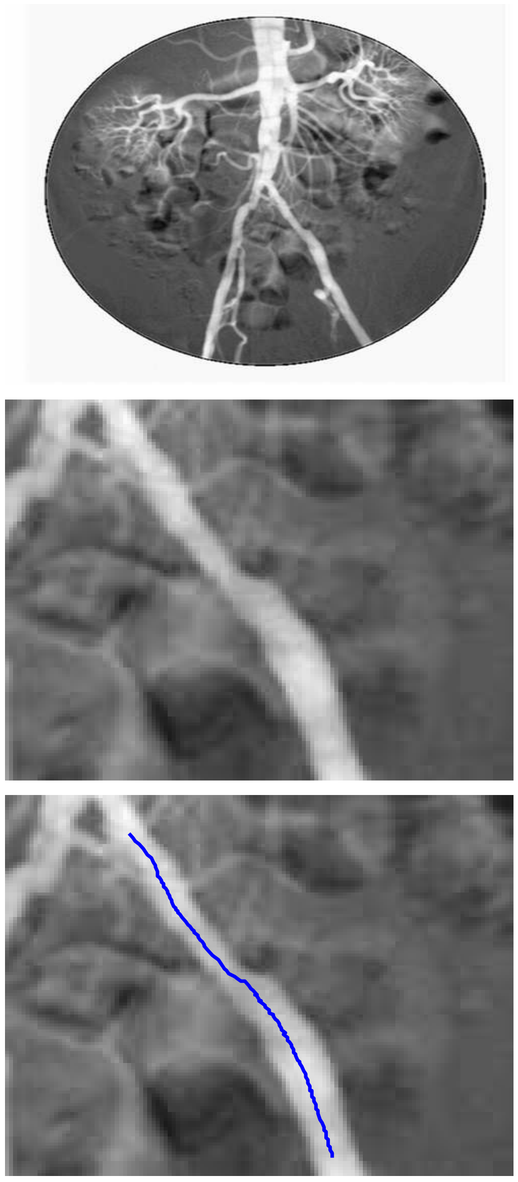


#1 #2

Figure 9.9: Detected centerline of vessel segments from selected ROI in DSA images. From *top* to *bottom*: original DSA image, user selected ROI, detected vessel segment.



#3



#4

Figure 9.9: Detected centerline of vessel segments from selected ROI in DSA images. From *top* to *bottom*: original DSA image, user selected ROI, detected vessel segment. (cont.)

Parameters	v_1	v_2	v_T
Seed point selection discriminator u_a	0.3	74.9	-82.4
Centerline fitting discriminator u_b	0.5	21.8	-176.0

Table 9.1: Parameters of the two linear discriminators we used for model energy minimization.

Method	Centerline Tracing		
Parameter	β	N_T	ω_T
Value	8	30	1.5

Table 9.2: Parameters used for centerline tracing.

Chapter 10

Conclusion

Linear or elongated feature detection is a very important issue in the areas of image analysis, computer vision, and pattern recognition. It has a wide range of applications such as in the medical or biometrics areas for retinal vessel extraction, and fingerprint analysis; in the biotechnology area for neurite outgrowth detection; in areas related to horticulture such as tree bark, tree branches, plant roots, and leaf vein detection; and in the infrastructure areas for road crack detection, roads, and valley detection in satellite images.

Our work focused on developing a fast and automatic system for detection of linear features in images, producing accurate length measurement, charactering the performance and making comparison with our previously developed algorithms for linear feature detection. This work was motivated by the increasing need for the development of machine related detection algorithm for automated underground image analysis, especially with the recent increase in the volume of minirhizotron data collected and the resulting complexity of data analyzed by humans.

We have presented a framework for automatic linear feature extraction from images. Linear features in images are modeled as realizations of a spatial point process of geometrical shapes. More specifically, we present a model based on the assumption that linear features form connected line segments in the image. A probabilistic model is defined, favoring connections of segments, alignments of segments, as well as a relevant interaction between segments. The estimation is performed by minimizing the energy of the model using a greedy algorithm whose parameters are determined in a data-driven manner.

One major contribution of the presented work is the greedy algorithm we designed for energy minimization. In contrast to most of previous algorithms which use a simulated annealing algorithm, based on a Monte Carlo dynamics (RJMCMC) for the model optimization, our method is much faster by extracting linear feature from the model using several linear discriminators trained by different type of linear objects.

Another contribution of our work is that our algorithm can extract each of linear features in the image instead of extracting just the entire linear feature network. After the linear feature extraction, properties of each linear feature can be measured separately which is important for horticulture analysis such as plant roots measurement and medical image analysis such as vessel stenosis estimation.

A previously developed matched filtering based algorithm is also described in this paper (see the appendix). Although this approach is computationally expensive, it has several noteworthy features. A technique known as local entropy thresholding is applied to segment the matched filter response (MFR) images. Instead of combining the matched filter responses from different orientation before thresholding, we choose to threshold each of the MFR images separately, followed by combining the outputs. A robust classifier built from a series of weighted weak feature classifiers is applied to discriminate linear features from unwanted background objects in thresholded binary images.

Two main directions may be outlined for future work. From a theoretical point of view, the linear discriminator we designed for removing bright background noise need to be more deeply investigated to reduce false negative decisions, especially for short segments. A stronger linear discriminator includes features which correspond to the stability of gray level intensity distribution of a linear feature may help to remedy this problem. From an application point of view, future work may include linear feature tracking in video sequence which can provide important information for horticulture researchers studying such as estimating plant root growth speed and computer-aided diagnosis such as measuring vessel stenosis severity.

We propose to apply the presented algorithm for detecting some other types of linear features in images such as blood vessel in DSA images and urban road detection in satellite images. Preliminary results shown in Chapter 9 is a good starting point. For blood vessel extraction, work will focus on reducing the noise caused by the uneven distribution of the dye in the vessel wall and false positive detection caused by the catheter inside the vessel lumen. For urban road detection, we expect to solve problems such as occlusion caused by shadow of trees and buildings, low contrast between the road and the background caused by the texture of road pavement materials, and false positive detection caused by other rectangular shape objects in the image.

Appendices

Appendix A

Template Matching Based Linear Feature Detection

In addition to the point process based method just described, we also developed a template matching based method for linear feature detection [67] earlier. This method was also used to detect bright young roots in minirhizotron images. To our knowledge, this method was the first attempt to go beyond merely classifying pixels as root or background and instead to actually detect an individual root by classifying a group of contiguous pixels as belonging to the same root. Experimental results from a collection of 200 minirhizotron images demonstrated the effectiveness of the approach. Although this approach is computationally expensive compared to our more recent algorithm, it is included here for the sake of completeness and to highlight some of its noteworthy characteristics.

This algorithm involves a series of processing steps. After initial preprocessing to enhance contrast between the young root and the background, matched filters at several orientations and two scales are applied to the image, utilizing assumptions about root color and shape. The resulting images are then separately thresholded using an automated technique known as *local entropy thresholding*. A robust root classifier is applied to discriminate roots from unwanted background objects in thresholded binary images, and a root labeling step used to identify individual roots.

A.1 Matched Filtering

Because roots generally have low curvature and their two edges run parallel to one another, a root can be represented by piecewise linear segments of constant width. Moreover, because young roots generally appear brighter than the surrounding soil (see Figure A.1), the gray level profile of the cross section of each segment can be approximated by a scaled Gaussian curve offset by a constant:

$$f(x, y) = A\left(1 + ke^{-\frac{d^2}{2\sigma^2}}\right), \tag{A.1}$$

where d is the perpendicular distance between the point (x, y) and the central axis of the root, σ defines the spread of the intensity profile, A is the gray level intensity of the local background, and k is the measure of reflectance of the plant root relative to its neighborhood.

Due to similarities between roots and blood vessels, the two-dimensional matched filter kernel developed by Chaudhuri et al. [7] for blood vessels is adopted here for detecting roots. (Similar approaches have been adopted by various researchers for detecting roads, canals, hedges, and runways in aerial images [47, 3, 17, 25, 2].) We convolve the image with a family of scaled Gaussian kernels at different orientations and scales:

$$K_{\theta,\sigma}(x, y) = \begin{cases} e^{-\frac{y_\theta^2}{2\sigma^2}} & \text{if } |x_\theta| \leq \frac{L}{2}, \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

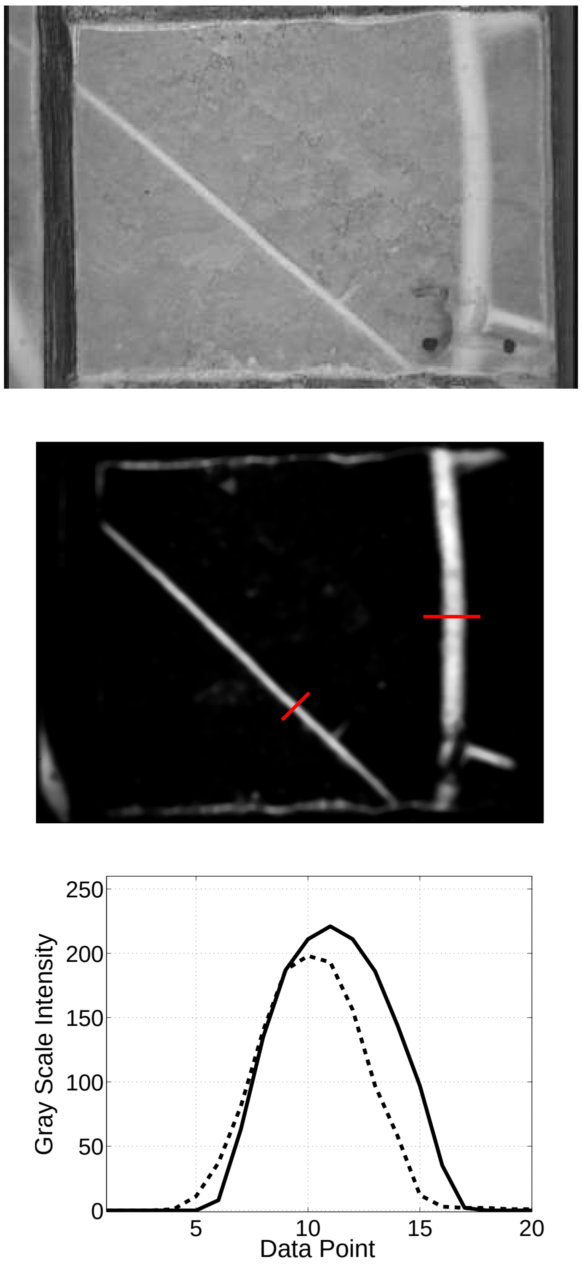


Figure A.1: The graylevel profile of the cross section of a young root approximates a Gaussian curve, with the peak of the Gaussian in the center of the root. From *top* to *bottom*: A minirhizotron image of two roots, the preprocessed version, a plot of the intensity profiles of the root cross sections in the preprocessed image along the line shown. (Dashed line for the oblique root on the left and solid line for the vertical root on the right) Note that this is a particularly clean image with little clutter, so the preprocessing alone goes a long way toward separating the root from the background.

where $x_\theta = x\cos\theta + y\sin\theta$ and $y_\theta = -x\sin\theta + y\cos\theta$ are the coordinates along and across the segment, respectively, and L is the length of the segment for which the root is assumed to have a fixed orientation.

If the background is viewed as pixels having constant intensity with zero mean additive Gaussian white noise, its ideal response to the matched filter should be zero. Therefore, the convolution kernel is modified by subtracting its mean value:

$$K'_{\theta,\sigma}(x,y) = K_{\theta,\sigma}(x,y) - \mu_{\theta,\sigma}, \tag{A.3}$$

where $\mu_{\theta,\sigma}$ is the mean of the values in the kernel $K_{\theta,\sigma}$. For computational efficiency, the coefficients in the kernel are multiplied by 100 and rounded to the nearest integer. To reduce the effect of background noise where no root segments are present, the mean value of the resulting kernel is forced to be slightly negative. Comparing Equations (A.1), (A.2), and (A.3), we see that $A = -\mu_{\theta,\sigma}$ and $k = 100/A$.

We apply the matched filter at 12 different orientations, spaced 15 degrees apart, and at two different scales ($\sigma = 2$ and $\sigma = 4$ pixels). Based on experimentation, we set $L = 11$ and the size of each kernel to 81×81 pixels, with pixels beyond the length L set to zero. A sample kernel is shown in Figure A.2. For computational efficiency, the larger sigma is achieved by downsampling the image by a factor of two in each direction and applying the same set of kernels to the downsampled image. Shown in Figure A.3 are the results of applying five of the matched filter kernels (at every other orientation) to the preprocessed image of Figure A.1. Notice that as the angle increases, the response to the vertical root on the right becomes stronger, while the response to the oblique root on the left becomes weaker.

	c1	...	c34	c35	c36	c37	c38	c38	c40	c41	c42	c43	c44	c45	c46	c47	c48	...	c81
r1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...									.	.	.								
r34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r36	-7	-7	-7	-6	-3	6	25	53	81	93	81	53	25	6	-3	-6	-7	-7	-7
...									.	.	.								
r40	-7	-7	-7	-6	-3	6	25	53	81	93	81	53	25	6	-3	-6	-7	-7	-7
r41	-7	-7	-7	-6	-3	6	25	53	81	93	81	53	25	6	-3	-6	-7	-7	-7
r42	-7	-7	-7	-6	-3	6	25	53	81	93	81	53	25	6	-3	-6	-7	-7	-7
...									.	.	.								
r46	-7	-7	-7	-7	-3	6	25	53	81	93	81	53	25	6	-3	-6	-7	-7	-7
r47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...									.	.	.								
r80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure A.2: The 81×81 matched filter at 180 degrees, with $L = 11$ and $\sigma = 2$.

A.2 Local Entropy Thresholding

In order to properly segment the roots from the background, we threshold the matched filter response (MFR) images. The threshold for each image is determined using a technique known as local entropy thresholding (LET) [45], which applies Shannon's classic notion of entropy [12] to the image co-occurrence matrix.

Let t_{ij} be the (i,j)th element of the co-occurrence matrix, i.e., t_{ij} is the number of pixels in the image with graylevel i whose immediate neighbor to the right or below has graylevel j. Thus, t_{ij} is defined as:

$$t_{ij} = \sum_{l=1}^{M} \sum_{k=1}^{N} \delta(l,k), \tag{A.4}$$

where

$$\delta(l,k) = \begin{cases} 1, \quad \text{if } f(l,k) = i \quad \text{and} \quad \begin{cases} f(l,k+1) = j \\ \text{or} \\ f(l+1,k) = j \end{cases} \\ \\ 0, \quad \text{otherwise,} \end{cases} \tag{A.5}$$

and where M and N are the image dimensions.

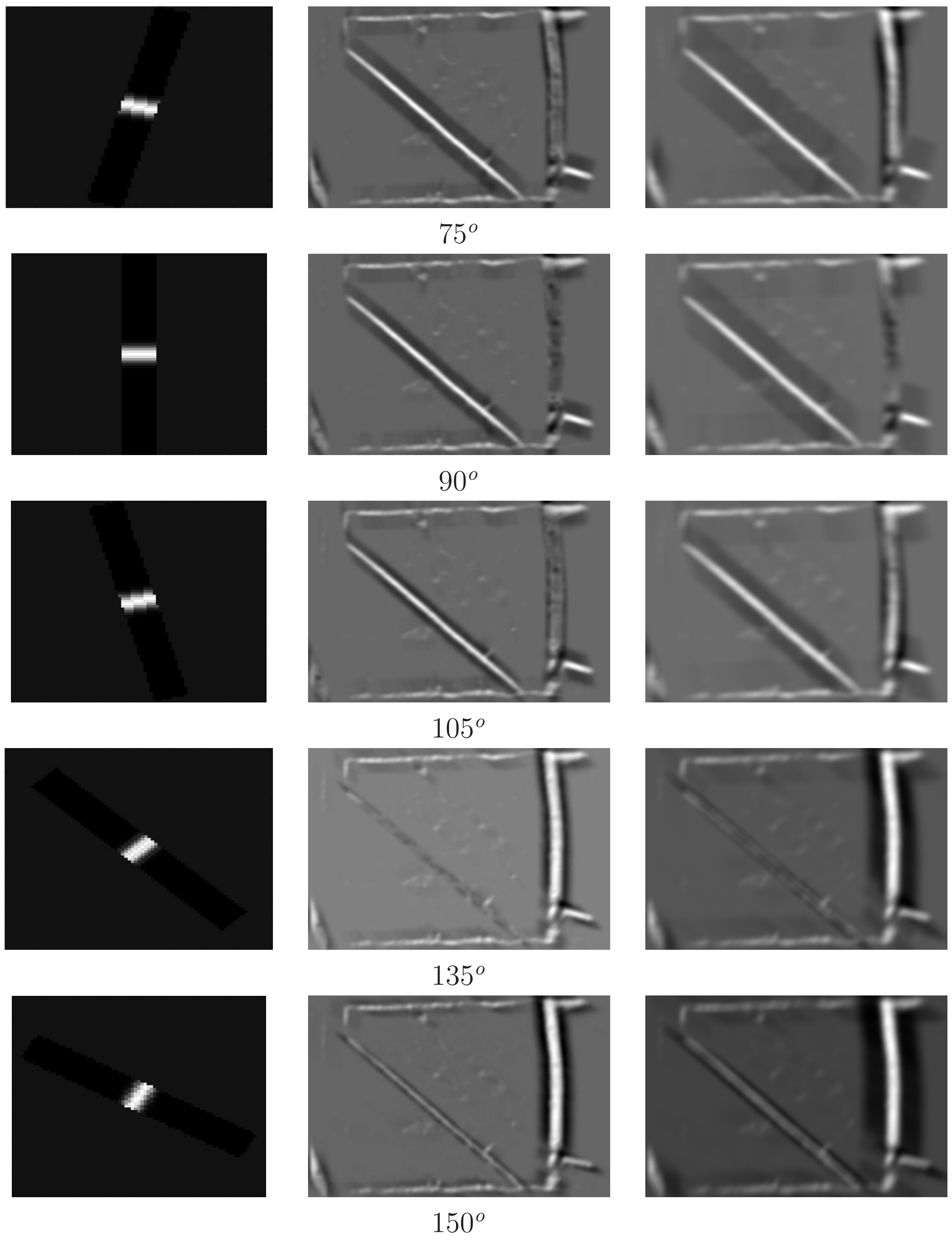


Figure A.3: Five matched filter kernels (left), along with the output matched filter response (MFR) images at five different angles for full size (middle) and half size (right). The half size images have been scaled for display.

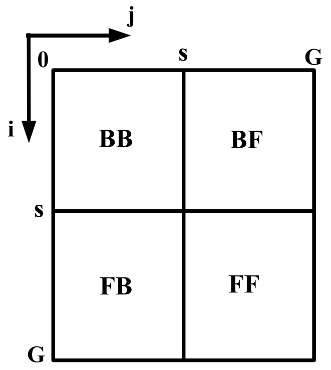


Figure A.4: Quadrants of the co-occurrence matrix.

The threshold s, $0 \leq s \leq G$ (where $G = 255$ is the maximum graylevel), partitions the co-occurrence matrix into four quadrants, namely BB, BF, FB, and FF, as shown in Figure A.4. Assuming that the foreground is lighter than the background, these quadrants correspond, respectively, to the transition from background-to-background, background-to-foreground, foreground-to-background, and foreground-to-foreground.

Quadrants BB and FF are used to define the local entropy. Treating the normalized co-occurrence matrix as a probability distribution, the probability of each $i \rightarrow j$ transition, conditioned upon the quadrant BB or FF, is computed as

$$\begin{aligned} P_{ij}^{BB} &= \frac{t_{ij}}{\sum\limits_{i=0}^{s}\sum\limits_{j=0}^{s} t_{ij}} \\ P_{ij}^{FF} &= \frac{t_{ij}}{\sum\limits_{i=s+1}^{G}\sum\limits_{j=s+1}^{G} t_{ij}}. \end{aligned}$$

The local entropy method uses the spatial correlation in the image as the criterion for selecting the optimal threshold by attempting to distribute the transition probabilities within each quadrant. The threshold is chosen to maximize the sum of the background-to-background entropy and the foreground-to-foreground entropy:

$$H_T(s) = H_{BB}(s) + H_{FF}(s), \tag{A.6}$$

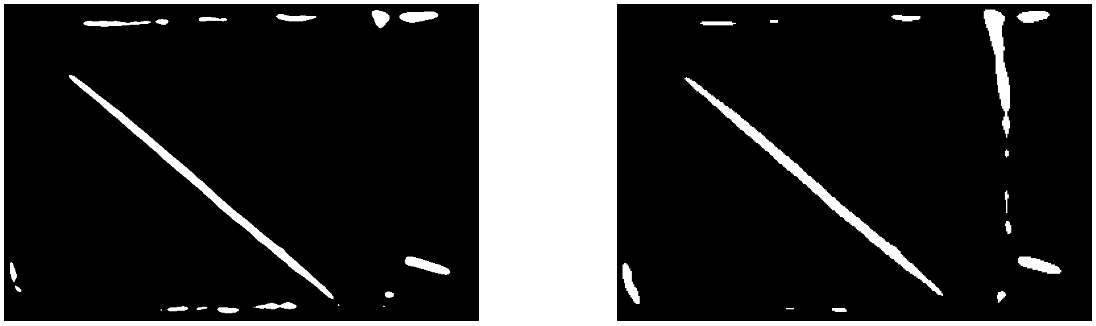
where

$$\begin{aligned} H_{BB}(s) &= -\frac{1}{2}\sum_{i=0}^{s}\sum_{j=0}^{s} P_{ij}^{BB}\log P_{ij}^{BB} \\ H_{FF}(s) &= -\frac{1}{2}\sum_{i=s+1}^{G}\sum_{j=s+1}^{G} P_{ij}^{FF}\log P_{ij}^{FF} \end{aligned}$$

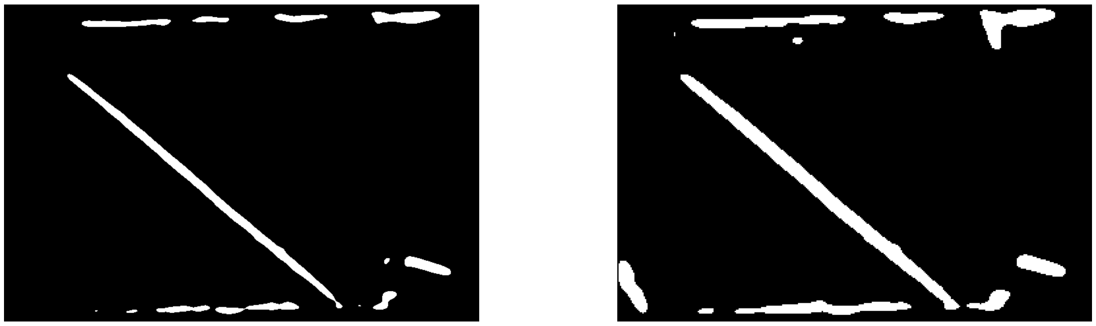
are the entropies of the two quadrants. Local entropy thresholding can be thought of as a simple form of texture segmentation in which there are exactly two objects separated by their graylevels. The results of applying LET to the matched filter response (MFR) images are shown in Figure A.5.

Because it takes spatial information into account, local entropy thresholding is superior to common thresholding techniques such as Otsu's method [43] that operate only on the graylevel histogram of the image. Figure A.6 compares LET with Otsu's method using two synthetic images sharing identical histograms. Ignoring all spatial information, Otsu's method incorrectly computes the same threshold in both cases, whereas LET is able to correctly segment both images by taking into account the spatial relationships of the pixels. The advantage of using LET versus Otsu's is clearly seen on several real images in Figure A.8. Even when the histogram is unimodal, LET is able to compute a threshold that successfully retains the roots and attenuates the distracting background pixels.

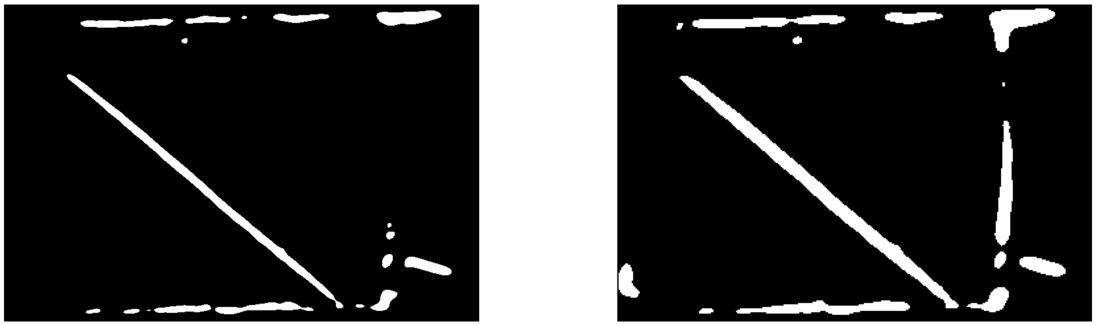
Traditionally, matched filter responses are combined (e.g., using a pixelwise maximum operator) before thresholding [6]. The drawback of this approach, however,



75^o

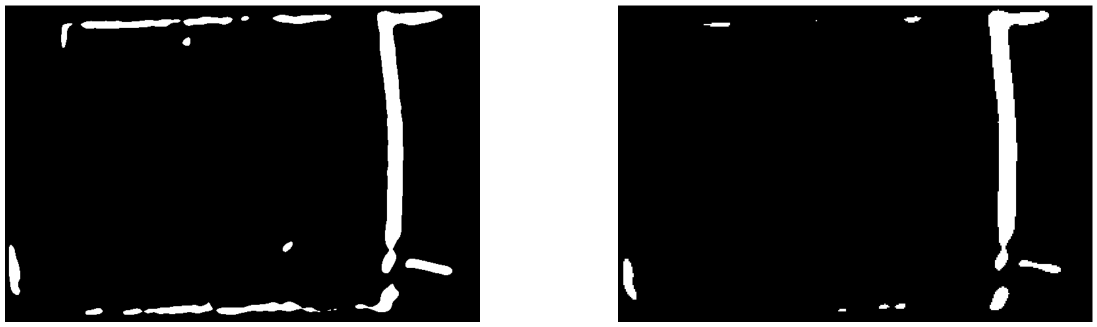


90^o

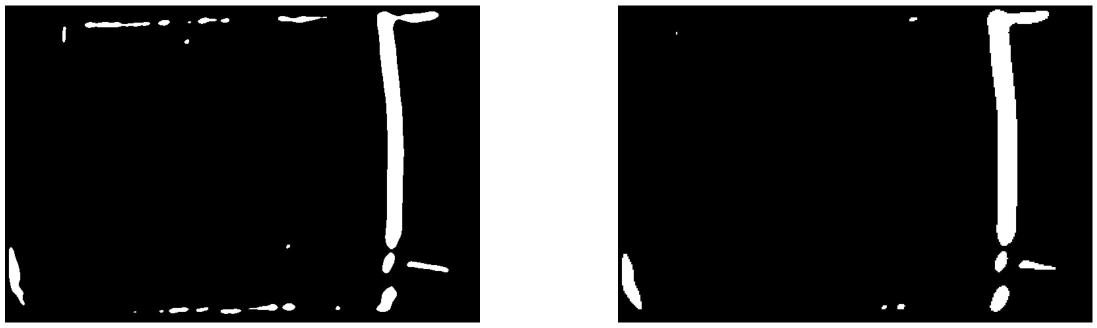


105^o

Figure A.5: T
he result of LET thresholding on the matched filter response (MFR) images The result of LET thresholding on the matched filter response (MFR) images from full scale *left* and half scale *right*.



135^o



150^o

Figure A.5: The result of LET thresholding on the matched filter response (MFR) images from full scale *left* and half scale *right*. (cont.)

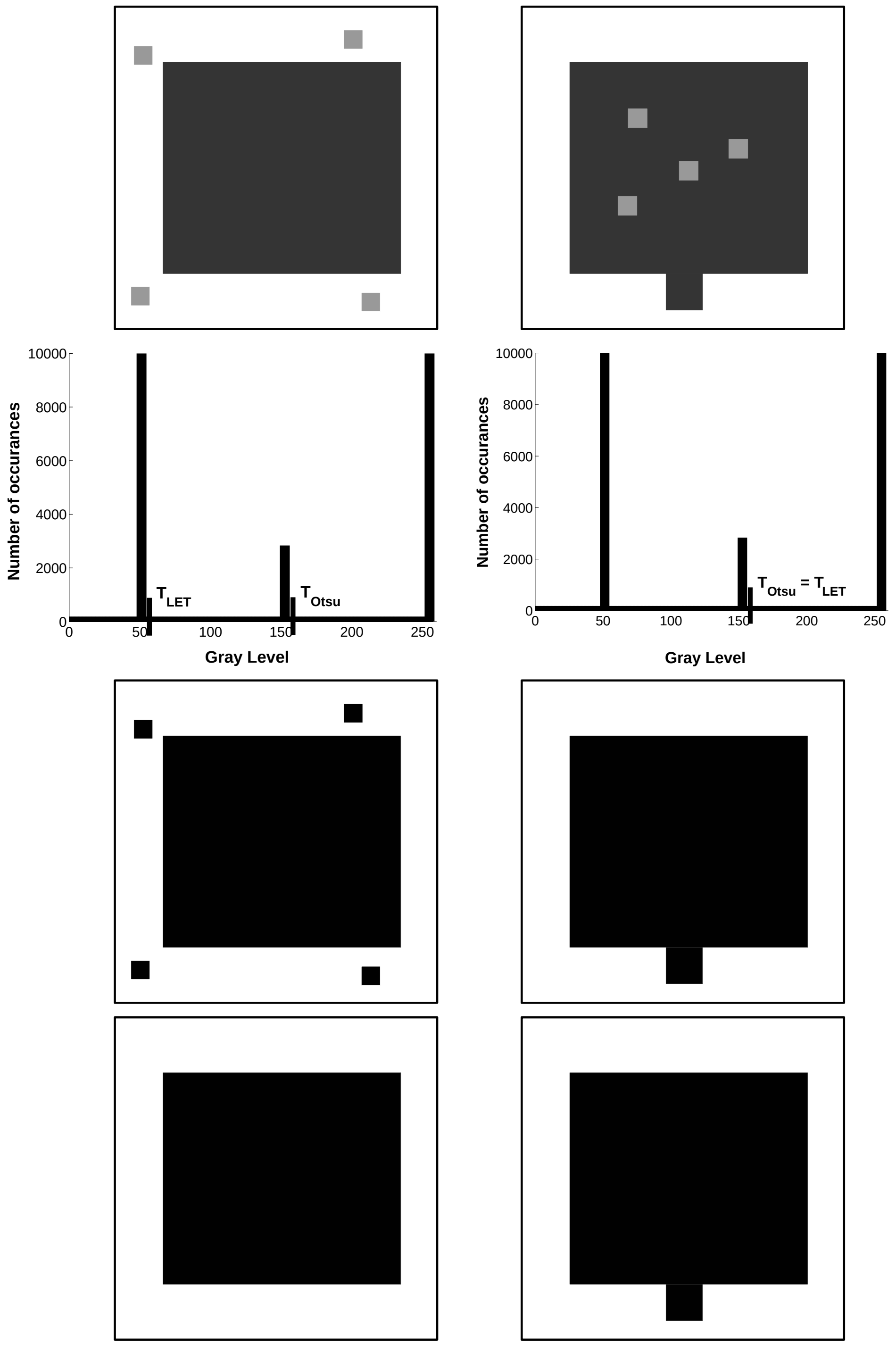


Figure A.6: A comparison of Otsu's method and local entropy thresholding (LET) on two synthetic images sharing the same graylevel histogram. The former incorrectly computes the same threshold value in both cases, while the latter successfully computes the correct thresholds.

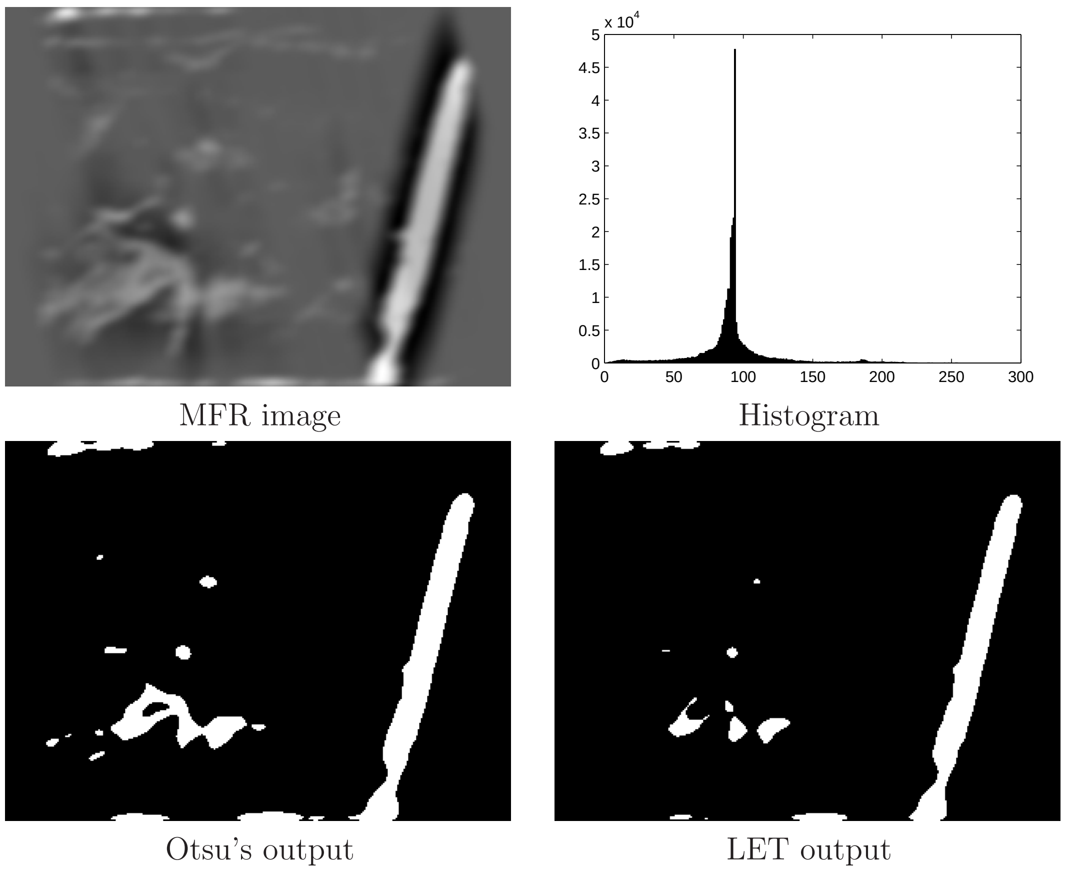
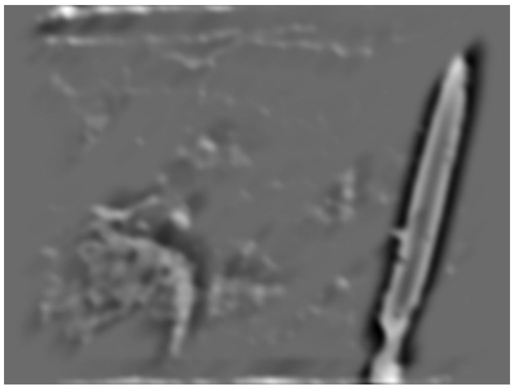
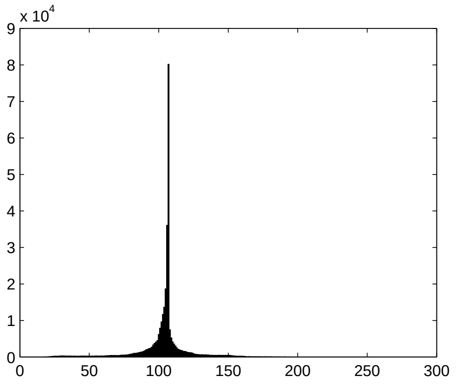
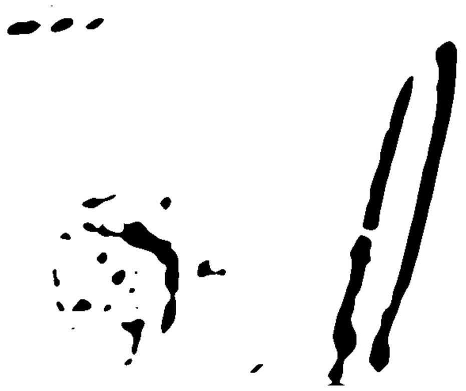


Figure A.7: A comparison of Otsu's method and local entropy thresholding (LET) on matched filter response (MFR) images. The MFR image is filtered at 105^o.


MFR image



Histogram


Otsu's output

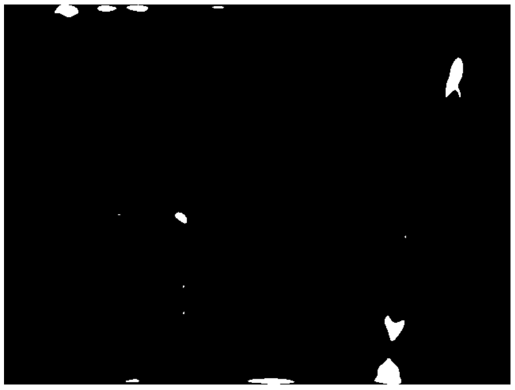

LET output

Figure A.7: A comparison of Otsu's method and local entropy thresholding (LET) on matched filter response (MFR) images. The MFR image is filtered at 45^o (cont).

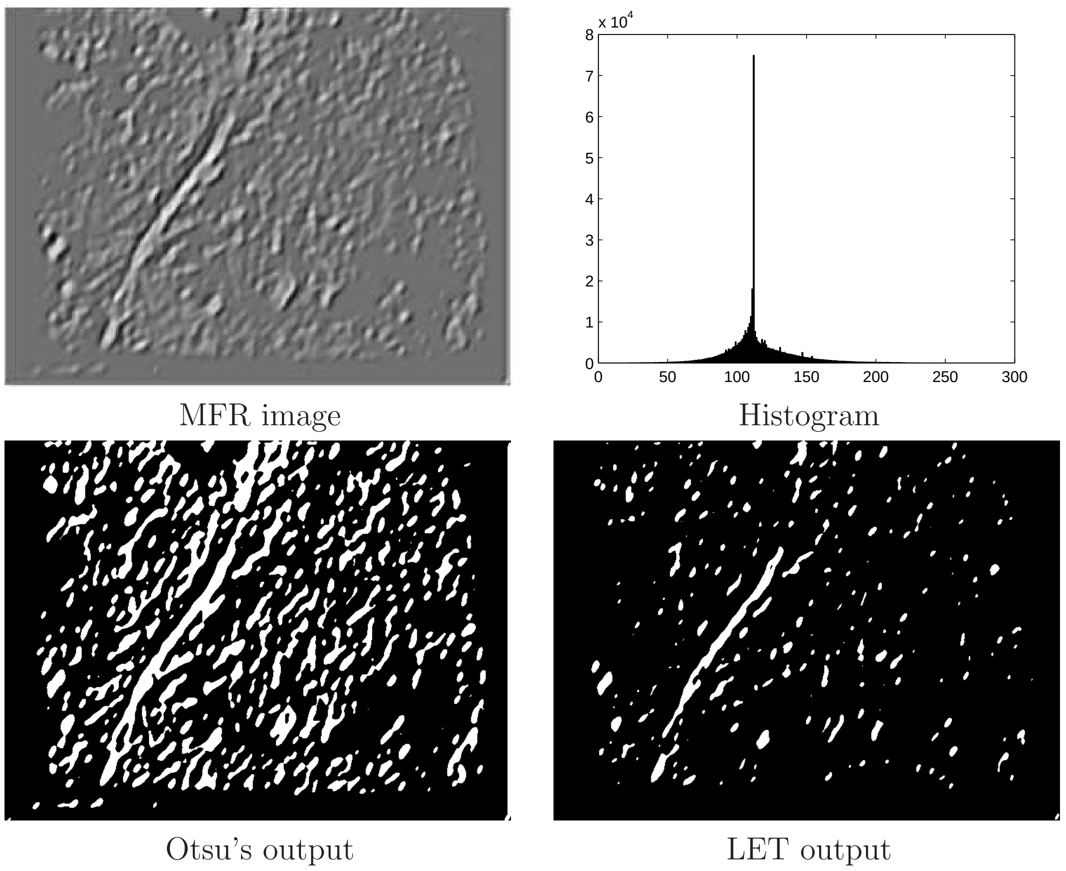


Figure A.7: A comparison of Otsu's method and local entropy thresholding (LET) on matched filter response (MFR) images. The MFR image is filtered at 150^o (cont).

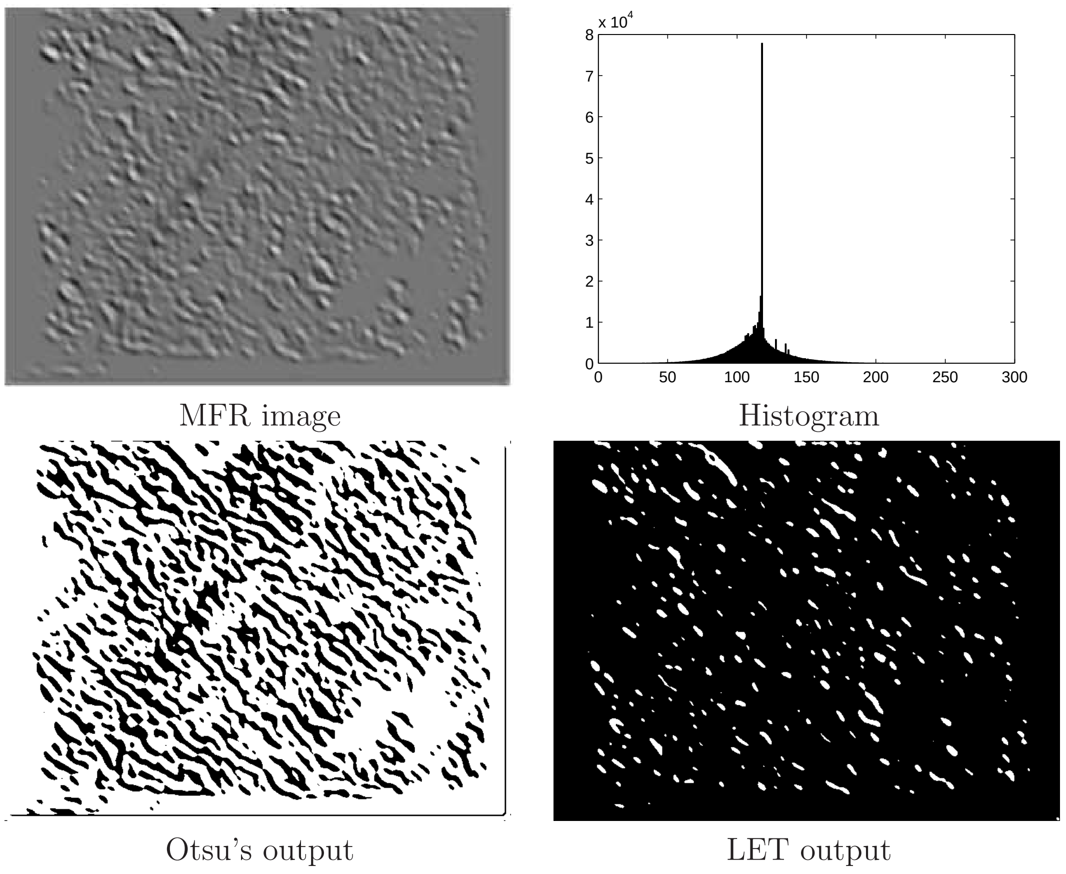


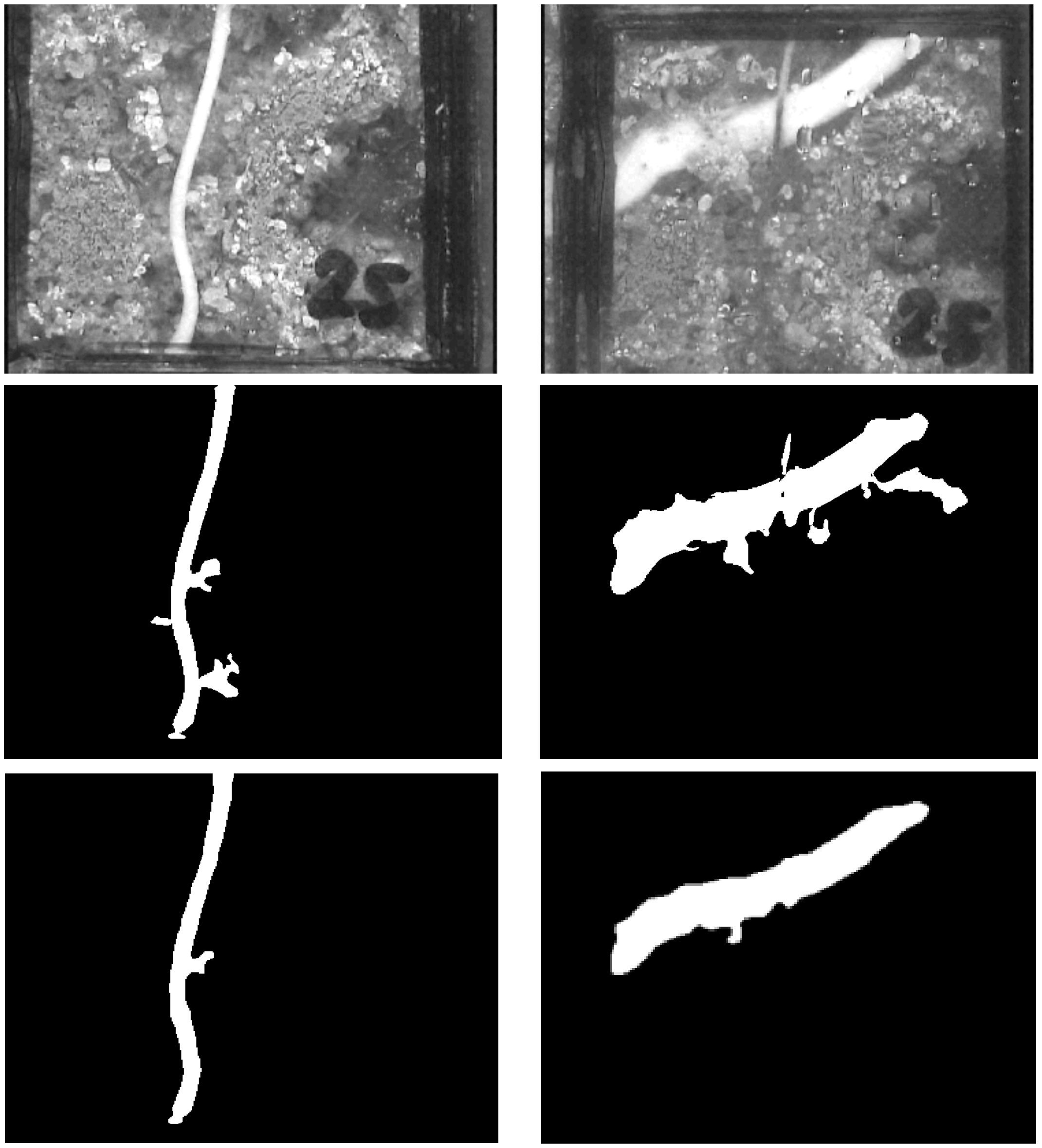
Figure A.8: A comparison of Otsu's method and local entropy thresholding (LET) on matched filter response (MFR) images. The MFR image is filtered at 45^o (cont).

is that it loses important information about the directionality of the responses. As shown in Figure A.9, thresholding the combined image results in shape distortion because bright background noise close to the main root segment is misclassified. We choose instead to threshold each of the MFR images separately, followed by combining the 24 outputs and extracting the root using the technique described in the next section.

A.3 Labeling Roots

After the root classifier discriminates roots from the bright background objects in the binary images, the classified components are compared with each other. Any pair of components that overlap by at least $p\%$ and whose orientations differ by no more than a certain amount ($\theta_{\max}$) are combined into one component. Among the remaining components, the challenge is then to determine which components belong to the same root, and which components belong to separate roots.

The problem is illustrated in Figure A.10. In this figure are shown two images in which the matched filter responses occur at multiple orientations, yielding components that overlap in the image. To determine whether the components are part of the same root or whether they indicate separate roots, we compare the two individual components, which we call R_1 and R_2, along with the combined component obtained by logically ORing the two individual components, which we call R_{12}. For each of the components R_1, R_2, and R_{12}, we find its extreme points vertically, called endpoints. If both of the endpoints of R_1 are more than a distance $d_{\max}$ to both of the endpoints of R_2, then R_1 and R_2 are labeled as separate roots. On the other hand, if one endpoint of R_1 is separated by one endpoint of R_2 by less than $d_{\max}$ while the remaining endpoints are separated by less than $d_{\max}$ to the endpoints of R_{12}, then R_1


25
25

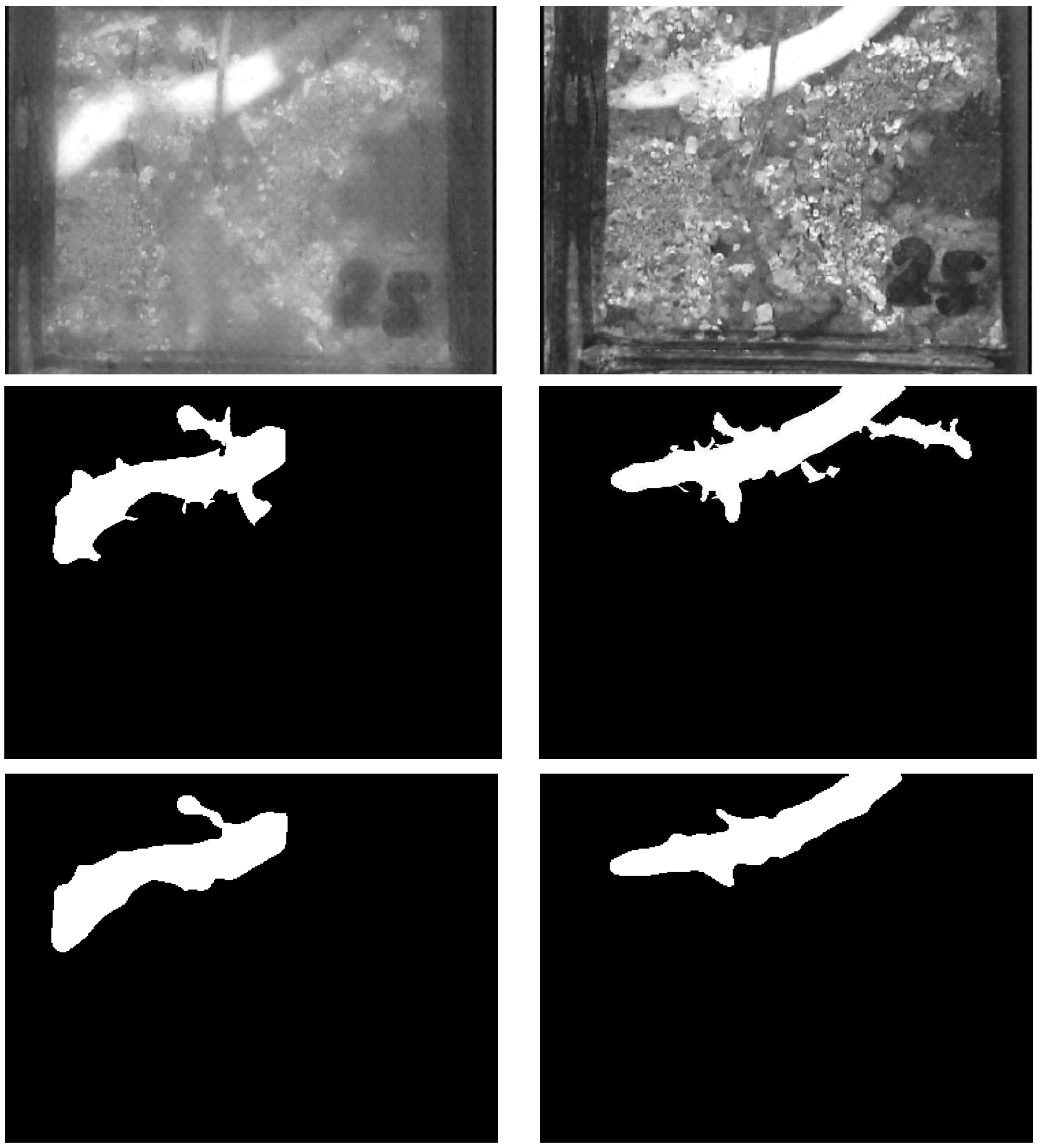


Figure A.9: Four different minirhizotron images (top), the result of thresholding the combined MFR image (middle), and the result of combining the 24 separately thresholded MFR images (bottom). Our approach reduces the shape distortion that results from combining all the orientations before thresholding.

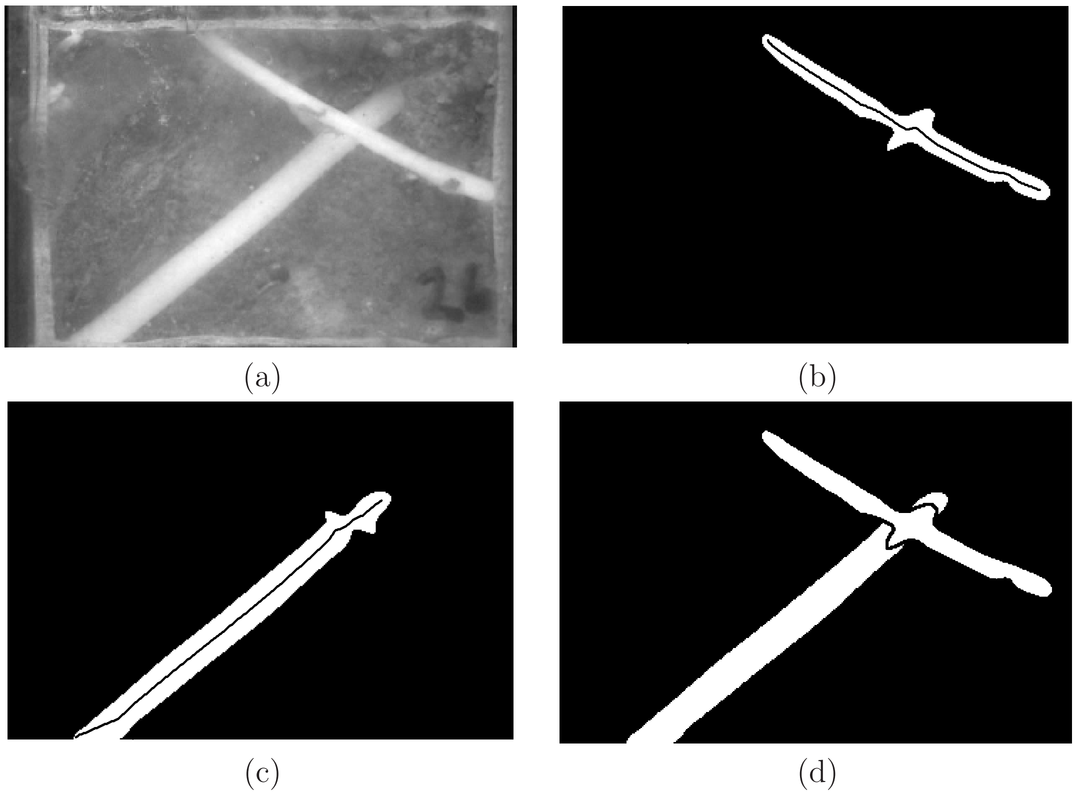


Figure A.10: (a) An image that yield overlapping matched filter responses in multiple directions. (b) and (c) The separate MFR components with the central axis overlaid. (d) The final result, in which the crossing roots are detected as separate roots.

and R_2 are combined into a single root. The results are shown in Figure A.11.

An additional challenge occurs when a root is partially covered by soil, in which case the algorithm detects two disjointed components for the same root. To overcome this problem, we compare the orientations of the components. If the orientations of the components differ by no more than $\theta_{\max}$, and if the orientation of the line connecting the centroids of the two components is less than $\theta_{\max}$ from the orientations of the components, then the separate components are considered to be portions of the same root. The results of all the processing steps described in this section are shown in Figure A.12 for five example images, using $p = 60$, $\theta_{\max} = 5$ degrees, and $d_{\max} = 30$ pixels.

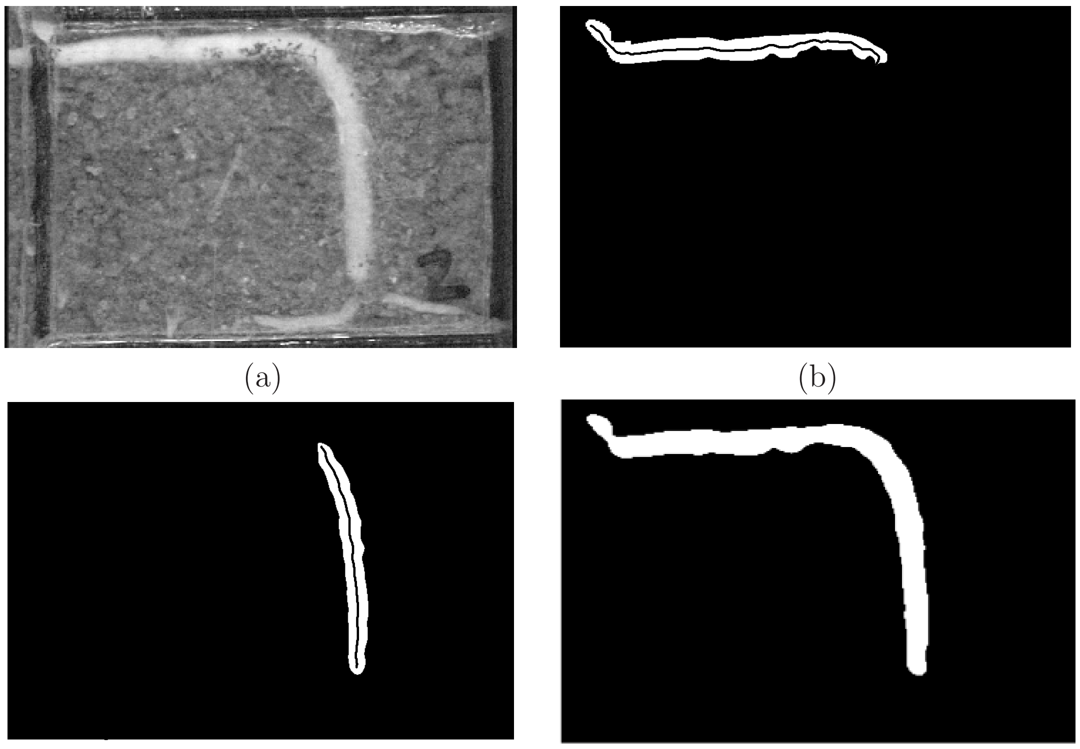

(a) (b) (c) (d)

Figure A.11: (a) An image that yield overlapping matched filter responses in multiple directions. (b) and (c) The separate MFR components with the central axis overlaid. (d) The final result in which the bending root is detected as one (cont).

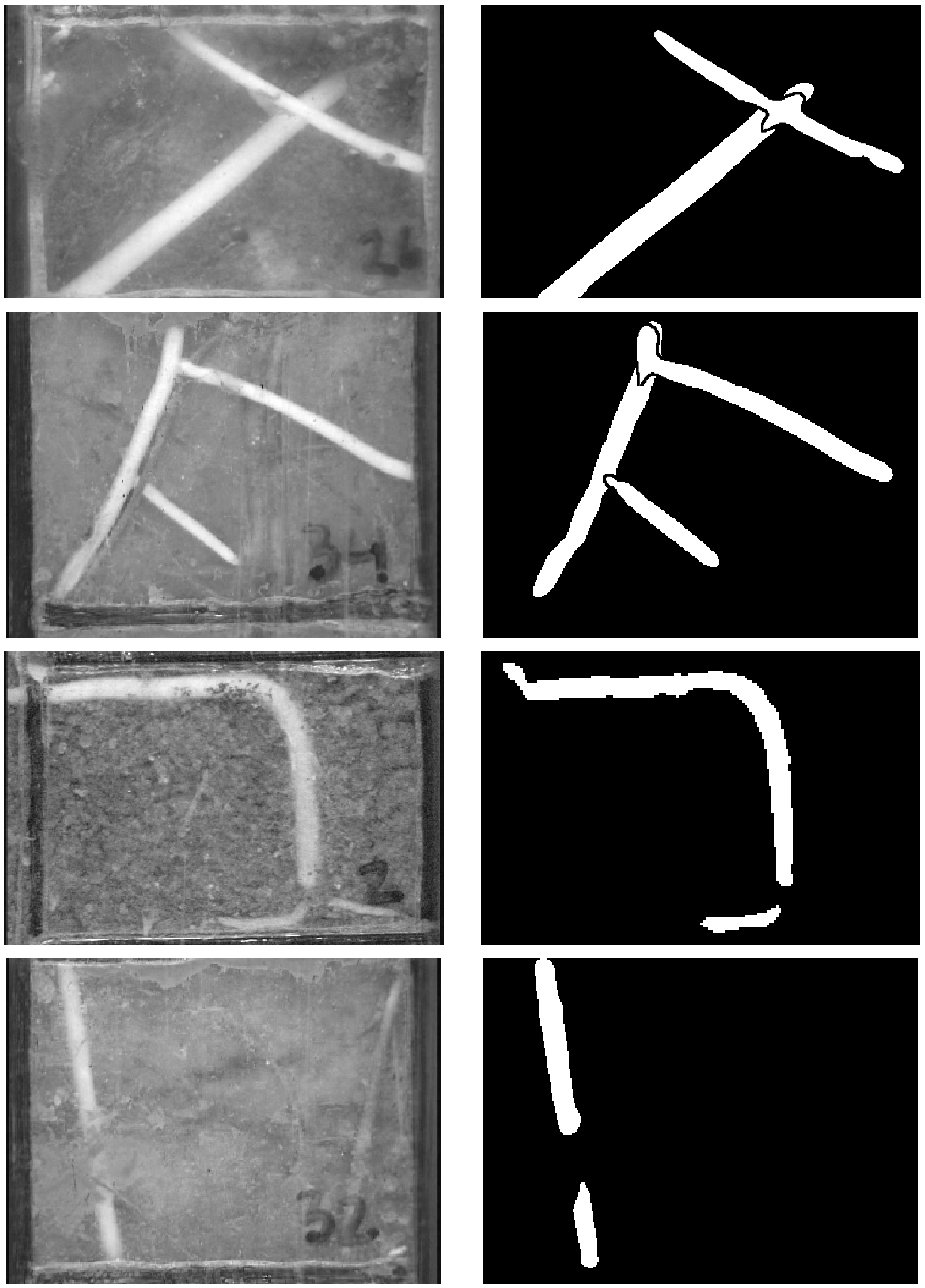


Figure A.12: Four images with the detected roots. The thin black lines indicate separate roots found by the algorithm. In the last row the two regions are detected as belonging to the same root

Bibliography

[1] A. Baddeley and M. N. M. V. Lieshout. Stochastic geometry models in high-level vision. *Statistics and Images*, 1:223–258, 1993.

[2] A. Barsia and C. Heipkeb. Artificial neural networks for the detection of road junctions in aerial images. In *Proceedings of the International Society for Photogrammetry and Remote Sensing (ISPRS) Workshop on Photogrammetric Image Analysis*, volume XXXIV, Sept. 2003.

[3] M. Barzohar and D. B. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):707–721, July 1996.

[4] A. Can, H. Shen, J. N. Turner, H. L. Tanenbaum, and B. Roysam. Rapid automated tracing and feature extraction from retinal fundus images using direct exploratory algorithms. *IEEE Transactions on Information Technology in Biomedicine*, 3(2):125–138, 1999.

[5] J. Cha, R. Cofer, and S. Kozaitis. Extended hough transform for linear feature detection. *Pattern Recognition*, 39:1034–1043, 2006.

[6] T. Chanwimaluang and G. Fan. An efficient blood vessel detection algorithm for retinal images using local entropy thresholding. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, volume 5, pages 21–24, 2003.

[7] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, and M. Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imaging*, 8(3):263–269, 1989.

[8] D. Chen, B. Li, Z. Liang, M. Wan, A. Kaufman, and M. Wax. A tree-branch searching, multiresolution approach to skeletonization for virtual endoscopy. In *Proceedings of the International Society for Optical Engineering*, volume 3979, pages 726–734, 2000.

[9] A. C. F. Colchester, R. Ritchings, and N. D. Kodikara. Image segmentation using maximum gradient profiles orthogonal to edges. *Image and Vision Computing*, 8(3):211–217, 1990.

[10] C. Coppini, M. Demi, R. Poli, and G. Valli. An artificial vision system for x-ray images of human coronary trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):156–162, 1993.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw–Hill, New York, 1990.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, New York, 1991.

[13] D. R. Cox and V. Isham. *Point Processes.* Chapman & Hall/CRC, 1st edition, 1980.

[14] X. Descombes and J. Zerubia. Marked point process in image analysis. *IEEE Signal Processing Magazine*, 19:77–84, 2002.

[15] X. Descombes and E. Zhizhina. The Gibbs fields approach and related dynamics in image processing. *Condensed Matter Physics*, 11(2):293–312, 2008.

[16] L. Dorst and A. W. M. Smeulders. Length estimators for digitized contours. *Computer Vision, Graphics, and Image Processing*, 40:311–333, 1987.

[17] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[18] G. Erz and S. Posch. Root detection by hierarchical seed expansion. In *Proceedings of the International Conference on Computer as a Tool (EROCON 2005)*, pages 963–966, Nov. 2005.

[19] M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. *Computer Graphics and Image Processing*, 15:201–223, 1981.

[20] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[21] H. Freeman. Boundary encoding and processing. In *Picture Processing and Psychopictorics*, pages 241–266. Academic Press, New York, 1970.

[22] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[23] C. A. Glasbey and G. W. Horgan. *Image Analysis for the Biological Sciences.* Wiley, Chichester, 1995.

[24] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice–Hall, New Jersey, 2nd edition, 2002.

[25] J. W. Han and L. Guo. An algorithm for automatic detection of runways in aerial images. *Machine Graphics and Vision International Journal*, 10(4):503–518, Sept. 2001.

[26] A. Hoover, V. Kouznetsova, and M. H. Goldbaum. Locating blood vessels in retinal images by piece-wise threshold probing of a matched filter response. *IEEE Transactions on Medical Imaging*, 19:203–210, 2000.

[27] M. Jacob and M. Unser. Design of steerable filters for feature detection using canny-like criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1007–1019, 2004.

[28] T. J. Jech, editor. *Set Theory (Perspectives in Mathematical Logic)*. Berlin: Springer-Verlag, second edition, 1997.

[29] R. Karcha, M. Neumann, F. Neumann, R. Ullrich, J. Neumuller, and W. Schreiner. A Gibbs point field model for the spatial pattern of coronary capillaries. *Physica A*, 369:599–611, 2006.

[30] K. Kimura, S. Kikuchi, and S. Yamasaki. Accurate root length measurement by image analysis. *Plant and Soil*, 216(1):117–127, 1999.

[31] I. S. Kweon and T. Kanad. Extracting topographic trrain features from elevation map. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 59(2):171–182, 1994.

[32] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, and A. Baumgartner. Automatic extraction of roads from aerial images based on scale space and snakes. *Machine Vision and Applications*, 12:23–31, 2000.

[33] R. J. Lebowitz. Digital image analysis measurement of root length and diameter. *Environmental and Experimental Botany*, 28:267–273, 1988.

[34] H. Liu, J. Li, and M. A. Chapman. Automated road extraction from satellite imagery using hybrid genetic algorithms and cluster analysis. *Journal of Environmental Informatics*, 1:40–47, 2003.

[35] J. B. A. Maintz, P. van den Elsen, and M. A. Viergever. Evaluation of ridge seeking operators for multimodality medical image matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):353–365, 1996.

[36] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[37] J. Moller and R. P. Waagepetersen. Modern statistics for spatial point processes. *Scandinavian Journal of Statistics*, 34:643–684, 2007.

[38] O. Monga, N. Armande, and P. Montesinos. Thin nets and ccrest lines: application to satellite data and medical images. *International Conference on Image Processing*, 3:2468–2468, 1995.

[39] O. Monga, N. Armande, and P. Montesinos. A common framework for the extraction of lines and edges. *International Archives of Photogrammetry and Remote Sensing*, 31:88–93, 1996.

[40] W. M. Neuenschwander, P. Fua, L. Iverson, G. Szekely, and O. Kubler. Ziplock snakes. *International Journal of Computer Vision*, 25(3):191–201, 1997.

[41] M. Ortner, X. Descombes, and J. Zerubia. Building outline extraction from Digital Elevation Models using marked point processes. *International Journal of Computer Vision*, 72(2):107–132, 2007.

[42] M. Ortner, X. Descombes, and J. Zerubia. A marked point process of rectangles and segments for automatic analysis of Digital Elevation Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):105–119, 2008.

[43] N. Otsu. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[44] P. P. Fua and Y. G. Leclerc. Model driven edge detection. *Machine Vision and Applications*, 3:45–56, 1990.

[45] N. R. Pal and S. K. Pal. Entropic thresholding. *Signal Processing*, 16:97–108, 1989.

[46] S. J. Perantonis and V. Virvilis. Efficient linear discriminant analysis using a fast quadratic programming algorithm. In *Proceeding of International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling*, volume 2781, pages 164–169, 1998.

[47] M. Petrou. Optimal convolution filters and an algorithm for the detection of wide linear features. *IEE Proceedings I, Vision, Signal and Image Processing*, 140(5):331–339, Oct. 1993.

[48] R. M. Rangayyan. *Biomedical Image Analysis*. New York: CRC, 1st edition, 2004.

[49] E. Renshawa and A. Sarkka. Gibbs point processes for studying the development of spatial-temporal stochastic processes. *Computational Statistics and Data Analysis*, 36:85–105, 2001.

[50] B. D. Ripley and F. P. Kelly. Markov point processes. *Journal of the London Mathematical Society*, 15:188–192, 1997.

[51] R. Rojas. *Neural Networks: A Systematic Introduction*. Berlin: Springer-Verlag, 1996.

[52] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[53] H. Rue and A. R. Syverseen. Bayesian object recognition with baddeley's delta loss. *Advances in Applied Probability*, 30:64–84, 1998.

[54] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York: John Wiley & Sons, 1992.

[55] L. I. Smith. A tutorial on principal components analysis. `http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`.

[56] R. F. Smith, B. K. Rutt, A. J. Fox, R. N. Rankin, and D. W. Holdsworth. Geometric characterization of stenosed human carotid arteries. *Academic Radiology*, 3(11):898–911, 1996.

[57] P. Soille and H. Talbot. Directional morphological filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1313–1329, 2001.

[58] C. Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):113–125, 1998.

[59] R. Stoica, X. Descombes, and J. Zerubia. A Gibbs point process for road extraction from remotely sensed images. *International Journal of Computer Vision*, 57(2):121–136, 2004.

[60] R. Stoica, V. J. Martinez, J. Mateu, and E. Saar. Detection of cosmic filaments using the candy models. *Astronomy and Astrophysics*, 434(2):423–432, 2005.

[61] D. Stoyan and A. Penttinen. Recent applications of point process methods in forestry statistics. *Statistical Science*, 15(1):61–78, 2000.

[62] K. Q. Sun, N. Sang, and T. X. Zhang. Marked point process for vascular tree extraction on angiogram. In *Proceedings of the 6th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 467–678, Aug. 2007.

[63] F. Tupin, H. Maitre, J. F. Mangin, J. M. Nicolas, and E. Pechersky. Detection of linear features in SAR images: application to road network extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 36(2):434–453, 1998.

[64] D. R. Upchurch and J. T. Ritchie. Root observations using a video recording system in mini-rhizotrons. *Agronomy Journal*, 75(6):1009–1015, 1983.

[65] L. Wang and T. Pavlidis. Detection of curved and straight segments from gray scale topography. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 58:352–365, 1993.

[66] G. Zeng. Autonatic minirhizotron root image analysis using two-dimenional matched filtering and local entropy thresholding. Master's thesis, Dept. of Electrical and Computer Engineering, Clemson University, May 2005. `http://people.clemson.edu/~guangz/thesis.pdf`.

[67] G. Zeng, S. T. Birchfield, and C. E. Wells. Detecting and measuring fine roots in minirhizotron images using matched filtering and local entropy thresholding. *Machine Vision and Applications*, 17(4):265–278, 2006.

[68] G. Zeng, S. T. Birchfield, and C. E. Wells. Automatic discrimination of fine roots in minirhizotron images. *New Phytologist*, 177:549–557, 2008.

[69] G. Zeng, S. T. Birchfield, and C. E. Wells. Rapid automated detection of roots in minirhizotron images. *Machine Vision and Applications*, 2008. (in press).

[70] Y. T. Zhou, V. Venkateswar, and R. Chellappa. Edge detection and linear feature extraction using a 2-D random field model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):84–95, 1989.