Clemson University
TigerPrints

All Theses | Theses

8-2010

Detecting Piecewise Linear Networks Using Reversible Jump Markov Chain Monte Carlo

Akshay Apte
Clemson University, apte.aa@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Part of the Electrical and Computer Engineering Commons

Recommended Citation

Apte, Akshay, "Detecting Piecewise Linear Networks Using Reversible Jump Markov Chain Monte Carlo" (2010). *All Theses*. 927.
https://tigerprints.clemson.edu/all_theses/927

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

DETECTING PIECEWISE LINEAR NETWORKS USING REVERSIBLE JUMP MARKOV CHAIN MONTE CARLO

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

by
Akshay Apte
August 2010

Accepted by:
Dr. Stanley Birchfield, Committee Chair
Dr. John Gowdy
Dr. Damon Woodard

ABSTRACT

This work proposes a piecewise linear network model to approximate structures observed in an image. An energy function is used to capture the characteristics of the structure. The energy function consists of two parts: the prior energy term and the data energy term. The prior energy term is calculated using prior information about the structures of interest. The data energy term is calculated using observations made from the image. The energy function is minimized using Reversible Jump Markov Chain Monte Carlo (RJMCMC) to get the approximate centerline of the structure. The algorithm was tested on a database of 150 images containing underground roots taken by a minirhizotron camera. The results show the importance of a novel non-Gaussian term introduced to handle roots with low intensity near the centerline. It is possible to use the proposed model to detect other structures such as roads as shown by the preliminary results.

DEDICATION

I will like to dedicate this work to everyone who has been there for me.

ACKNOWLEDGMENTS

I would like to thank my Advisor Dr Stan Birchfield without whom it would not have been possible for me to complete this work. His patience foresight and valuable guidance was the main reason for the success that I have achieved. I would also like to thank my committee members Dr Gowdy and Dr Woodard.

I would also like to thank all the people in my lab and research group. All of them were of great help time and time again.

Thanks to all the people who helped develop *Blepo* — the computer vision library in our department. It made all the code possible.

Last but not least I would like to thank my parents and my sister. It is due to there support and encouragement that I am here today.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF FIGURES

Chapter 1

Introduction

This thesis proposes a method to detect structures in an image as piecewise linear networks. A linear network is a sequence of line segments that are connected to each other. A line segment is defined as the shortest distance between two pixels in a given image. In this work a line segment can be connected to at most two other line segments. The connections always take place at the end points of the line segment.

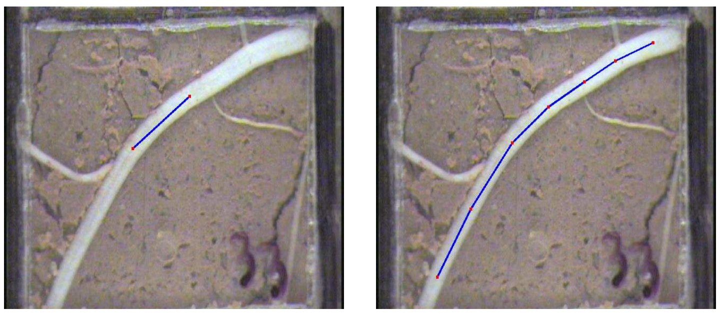

Figure 1.1: Left image: A line segment, Right image: A sequentially connected linear network with six segments.

Detecting structures in an image is an important first step for high level tasks in computer vision. A few examples of structures that can be approximated by piecewise linear networks are roads, roots and blood vessels. Detecting these structures allows researchers to draw further inferences such as

- Calculating the length and width of the roots can allow researchers to measure the growth of the plants.
- Detecting roads can help GIS researchers in mapping.
- Detecting blood vessels can help in determining if there are blockages in them.

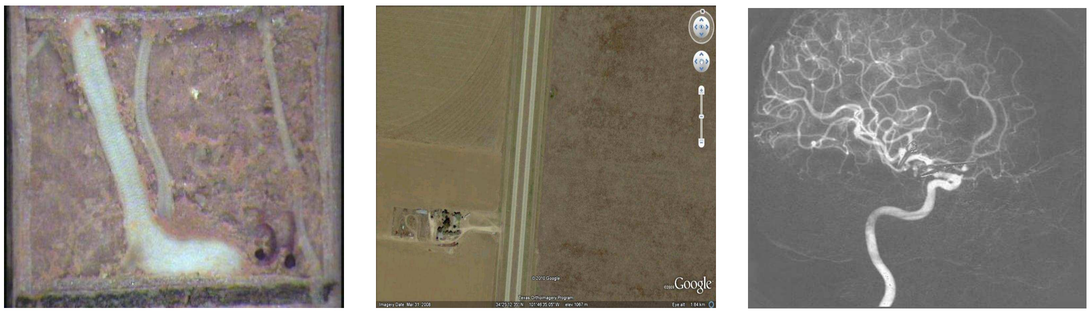

Figure 1.2: Examples of structures in images that can be approximated with piecewise linear networks.

An energy function is formulated using the appearance of these structures and then minimized to find the centerline of the structures. Due to their similar appearance it is possible to approximate all the above structures using a similar model. As seen from Figure 1 objects like roads and roots have long, thin ribbon like structures. These objects are usually brighter than the surrounding areas. They also do not have any sharp bends in their structure. All these properties of the object are used to formulate an energy function. This energy function is then minimized using Reversible Jump Markov Chain Monte Carlo (RJMCMC) to detect the objects.

1.1 Previous Work

The main work on detecting roots from minirhizotron images has been done by Zeng et al. in [19],[20],[21]. They use a greedy algorithm for minimizing the energy function that they have. The energy function is derived from seed points and the Candy model. The approach proposed in this thesis has both a different energy function and a different energy minimization technique. The assumption made for the Gaussian profile of the root is not correct in all the cases as discussed in the next chapters.

The RJMCMC algorithm used for energy minimization in this method was originally proposed by Green in [7]. This algorithm has been used in computer vision for a variety of applications. Khan et al. [8] use RJMCMC for multi object tracking, Smith et al. [14] also use RJMCMC for multiple people tracking. RJMCMC is also used for segmentation of muscle fibers in [6].

Stoica et al. [17] use a method similar to the proposed one to detect roads. The energy model that they used is called the candy model and is used by Zeng et al. as well. The candy model however is very different from the model used in this method. They use RJMCMC to minimize the energy function as well. However they consider the network as a set of connected segments with interactions instead of a sequence of connected segments. The connections between the connections are also weak connections as two segments are connected if there endpoints are within a distance of some predefined pixels. In the followup work Stoica et al. [10] extend the Candy model by using similar energy minimization techniques.

One application of the proposed method is road detection. The work previously done on road detection in[2], [3],[10] and [17] does a much better job of road detection than the proposed method. The number of roads in most of road detection methods is much larger than the number of roads detected using the proposed method. However the method in this thesis is primarily for modeling roots. Detecting roads is possible only due to the similarity in nature of these two structures.

1.2 Image Database

As mentioned in the abstract the main focus of the algorithm is to detect roots seen in an image taken by a minirhizotron camera. Minirhizotrons are transparent plastic tubes buried at an angle in the soil near the plants to be observed. A camera is then inserted in these tubes. This camera captures the images of the roots and transfers them to a computer hard drive. These images are then used by plant researchers to study the health of plants. The image database consists of 150 images taken by these minirhizotron camera. The database for detecting roads are images taken from Google earth. So far a set of 30 road images has been used.

1.3 Thesis Outline

In this work an attempt is made to develop an energy function that will capture the characteristics of the roots seen in images captured by a minirhizotron camera. Each root is defined by a sequence of connected segments. In order to detect the roots the distinguishing characteristics of the roots are captured in the form of an energy function. This energy function is evaluated for the pixels that are part of the series of the segments. The energy function is defined in such a way that the series of segments will have minimum energy at the center of the root.

The energy of the network is obtained by taking the negative logarithm of the probability distribution function. The details of the energy function are given in Chapter 2. In brief, Bayes' rule is used to evaluate the probability distribution function and get the energy value for a given state. The two parts of the Bayes' equation (prior and likelihood) are calculated using the state of the network used to capture the root and the actual data present at the pixels of the linear network. The energy of the network is then minimized using Reversible Jump Markov Chain Monte Carlo. The details of this are provided in Chapter 3. The algorithm proposes various moves which govern the next state of the network. The next state is accepted or rejected based on the acceptance ratio. The minimization algorithm is then run till the chain converges to a stationary distribution. This stationary distribution is the final state of the network that captures the shape of the root.

Chapter 4 provides the details of the actual working of the algorithm. This chapter describes the preprocessing and the way the root network is initialized in the image. Chapter 5 shows the results and experiments to show the effectiveness of the algorithm and compare it with the results of previous work. It is possible to use the model for roots to detect other similar structures like roads and blood vessels. Chapter 6 describes the modifications necessary for other objects and the results for other objects.

Chapter 2

Problem Formulation

This chapter provides the description of the problem formulation for detecting the roots seen in the images captured by a minirhizotron camera. Each root is described by a network of sequentially connected line segments $S = \langle s_1, s_2, \ldots, s_n \rangle$, where the number n of line segments is determined by the algorithm. Each line segment s_i of the network is described by a 3-tuple $(\bar{x_i}, \theta_i, \ell_i)$, where $\bar{x_i} = (x_i, y_i)$ contains the coordinates of the centroid of the ith line segment in the image, θ_i is the orientation of the segment (clockwise from the positive horizontal axis), and ℓ_i is its length (in pixels). Due to the fact that consecutive segments are connected, only $2n + 2$ unique parameters are necessary to fully specify such a network. Therefore, since the representation just given stores $4n$ values, it is clearly redundant as the end point of one segment is the same as the start point of the next segment: $x_{i+1} - \frac{1}{2}\ell_{i+1} \cos\theta_{i+1} = x_i + \frac{1}{2}\ell_i \cos\theta_i$, and similarly for the y coordinate. During the energy minimization the algorithm explores the state space by changing n and S.

2.1 Derivation of Energy Function

Let $Z = \langle z_1, z_2, ..z_m \rangle$ be the observations made about the network from the image. The energy $\psi\left(S|Z\right)$ of the network is obtained by taking the negative logarithm of the *a posteriori* probability.

$$\psi\left(S|Z\right) = -\log\Big(P\left(S|Z\right)\Big). \tag{2.1}$$

By Bayes' rule the *a posteriori* probability can be written as

$$P\left(S|Z\right) \propto P\left(Z|S\right)P\left(S\right). \tag{2.2}$$

Here $P(Z|S)$ is the likelihood term to represent the data from the image and $P(S)$ is the prior term that gives information about the state of the network. By using (2.2) we can write (2.1) as

$$\begin{aligned} \psi\left(S|Z\right) &\propto -\log\Big(P\left(Z|S\right)P\left(S\right)\Big) \\ &= -\log\Big(P\left(Z|S\right)\Big) - \log\Big(P\left(S\right)\Big) \\ &= \psi_d\left(Z|S\right) + \psi_p\left(S\right). \end{aligned} \tag{2.3}$$

Here $\psi_d\left(Z|S\right)$ is the energy term for the data obtained about the network from the image and $\psi_p\left(S\right)$ is the prior energy of the network.

The data energy term from the above equation is calculated using multiple observations made from the image. Assuming that all the observations are independent of each other yields

$$\psi_d\left(Z|S\right) = \sum_{i=1}^{m} \psi_d\left(z_i|S\right), \tag{2.4}$$

where m is the number of observations.

2.2 Prior Energy Terms

The state of a segment in the network is given by the 3-tuple $(\bar{x}_i, \theta_i, \ell_i)$. Out of the three the location vector $\bar{x}_i$ does not contribute anything to the state energy term. Therefore the prior energy is determined by the length and the orientation. However it is not that the orientation of each segment is important, but rather the difference in orientation of two connected segments. The prior energy term can be written as

$$\psi_p\left(S\right) = \psi_\ell\left(S\right) + \psi_\theta\left(S\right) \tag{2.5}$$

where $\psi_\ell\left(S\right)$ is the length energy term and $\psi_\theta\left(S\right)$ is the orientation differnce energy term.

To decide energy function for each of the prior we make use of manually generated ground truth. The ground truth is generated using a set of 50 images. The ground truth consists of linear networks at the center of each root in an image. This ground truth is used to analyze the properties of segments that form the network.

By using the ground truth a histogram for the orientation difference is calculated. The histogram for the ground truth is given in Figure 2.1 by the solid (red) line. As mentioned previously the orientation difference between two segments is of importance. The histogram plot is for the orientation difference term $\theta_{i,i+1}$ and not for orientation of each segment. The histogram shows that many segments have orientation difference of $0-10$ degrees. There are hardly any segments with orientation difference over 50 degrees. If the images of roots are observed it is seen that most of the roots have thin ribbon like long structures. These structures do not have any sharp bends in them. It is therefore not surprising that the orientation difference between two segments is very small. If the histogram is observed then the distribution resembles an exponential distribution. The energy function should reward a pair of segments with small orientation difference and penalize a pair of segments with large orientation dif-

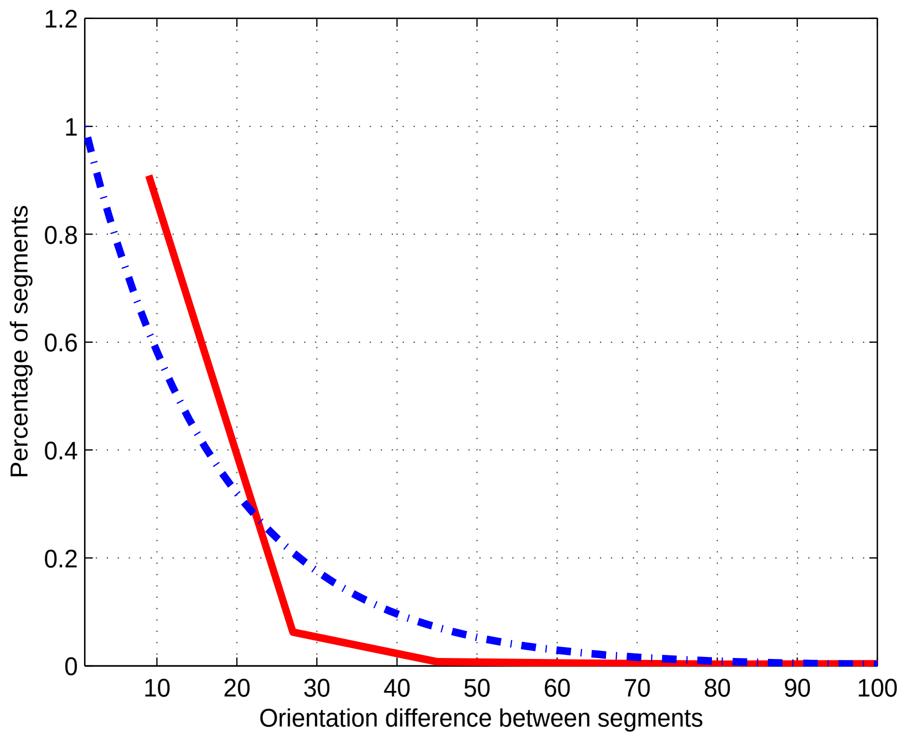

Figure 2.1: The solid (red) line shows the normalized histogram for the orientation difference between segments, the dashed (blue) line shows the shape of $e^{-\psi(\theta)}$ for that orientation difference value.

ference. The penalty should increase exponentially to mirror the histogram of the orientation difference. The dashed (blue) line represents the shape of the negative exponent of the energy function for the orientation difference. If the negative logarithm of the energy function is plotted, then orientation differences near 0 degrees have minimum value. This value increases exponentially afterwards

$$\psi_{\theta}\left(S\right)=\lambda_{1}\bar{\theta}_{i,i+1}, \tag{2.6}$$

where the term $\bar{\theta}_{i,i+1}$, the average angle difference of the network is defined as:

$$\bar{\theta}_{i,i+1} = \frac{1}{n-1}\sum_{i=1}^{n-1}\theta_{i,i+1} \tag{2.7}$$

The term λ_1 is a scaling constant that gives the distribution function which is of proper shape. To calculate this term the average orientation difference is calculated which is substituted in equation (2.6).

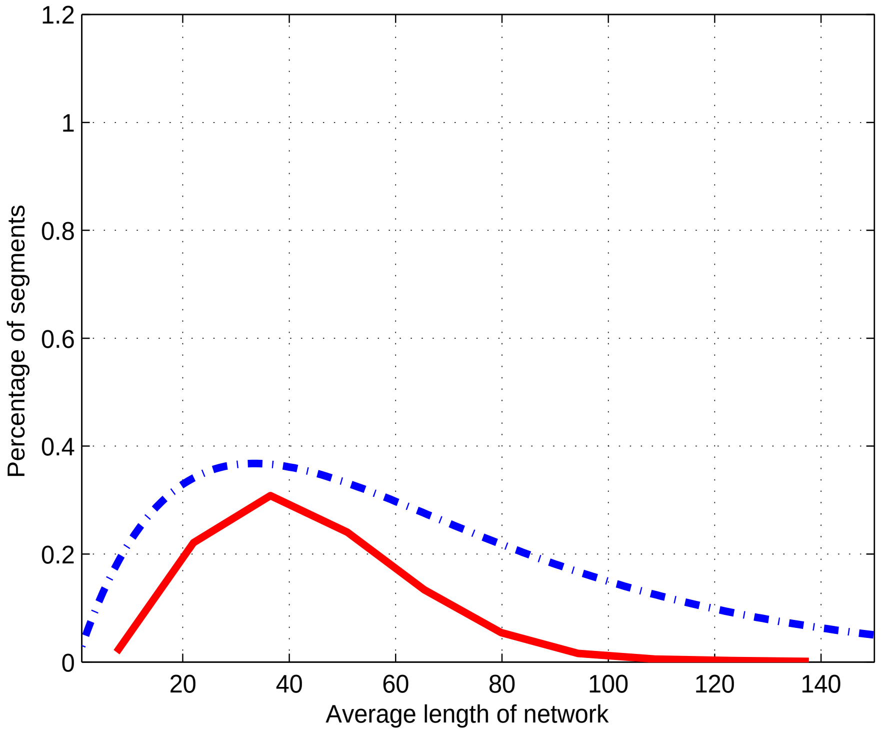

Figure 2.2: The solid line shows the normalized histogram for the length of segments and the dashed line shows the shape of $e^{-\psi(L)}$ for that length value.

The histogram for the length is also calculated using the ground truth for calculating histogram of orientation difference. The histogram for average segment length for each network is shown in Figure 2.2 by the solid (red) line. The histogram for the average segment length has an odd shape. It looks like a Gaussian curve with mean at 50. The curve, however is skewed towards the left side. Most of the network have segments with average lengths in the region of $30 - 50$ pixels. The assumption made about the roots is that they have a long ribbon like structure. This leads to a conclusion that the longer a segment the better it is. But from the histogram it is seen that long segments are not as likely. Because the image size is 640×480, this limits the maximum size that a segment can have. However most of the roots occupy at least one third of the image height or width. Most of the segments should have lengths close to $70 - 80$ pixels. On closer look it is observed that roots have subtle bends in them. In order to maintain the segments at the center of the root it becomes necessary to break the segment at the point of bend and start a new segment. This results in segment that are shorter than the expected value. The distribution of the lengths is approximated by a distribution given by $\lambda x e^{-\lambda x}$, where x shall be ℓ and λ shall be λ_2. Taking the negative logarithm yields the dashed line in Figure 2.2.

$$\psi_\ell\left(S\right) = -\log\left(\lambda_2\right) - \log \bar{\ell} + \lambda_2 \bar{\ell}, \tag{2.8}$$

where ℓ, the average length of the network is defined as:

$$\bar{\ell} = \frac{1}{n}\sum_{i=1}^{n} \ell_i. \tag{2.9}$$

The constant λ_2 is used to adjust the shape of the function. This function rewards segments that have length between $30 - 50$ pixels. The roll of this function is similar to the distribution observed from the ground truth. To calculate the length term the average length of the segments in the network is calculated and then substituted in equation (2.8).

By using equations $(2.6, 2.8)$ the equation for the state energy becomes

$$\psi_p\left(S\right) = \lambda_1 \bar{\theta}_{i,i+1} - \log\left(\lambda_2\right) - \log \bar{\ell} + \lambda_2 \bar{\ell} \tag{2.10}$$

For detecting roots the value of λ_1 is 0.00001 and λ_2 is 0.015.

2.3 Data Energy Terms

The second term in the equation for the network energy is the data or observation energy term. As mentioned previously this term is calculated using multiple observations based on the state of the network. For detecting there are five observations that are used to calculate the data energy term:

1. Average pixel intensity of the network.
2. The number of seed points close to each segment.
3. Aspect ratio of the root with respect to the segment.
4. The sum of absolute difference of angle between the segments.
5. Correction for non-Gaussian profile of the root.

Each of the above terms and their significance is discussed in this section of the chapter.

The first data energy term is the average intensity of the network. The intensity of the network provides important information about the brightness of the network. The roots are usually brighter than the background. The ground truth that was plotted previously is used to find the histogram of the average intensity of the network. Average intensity is the average of

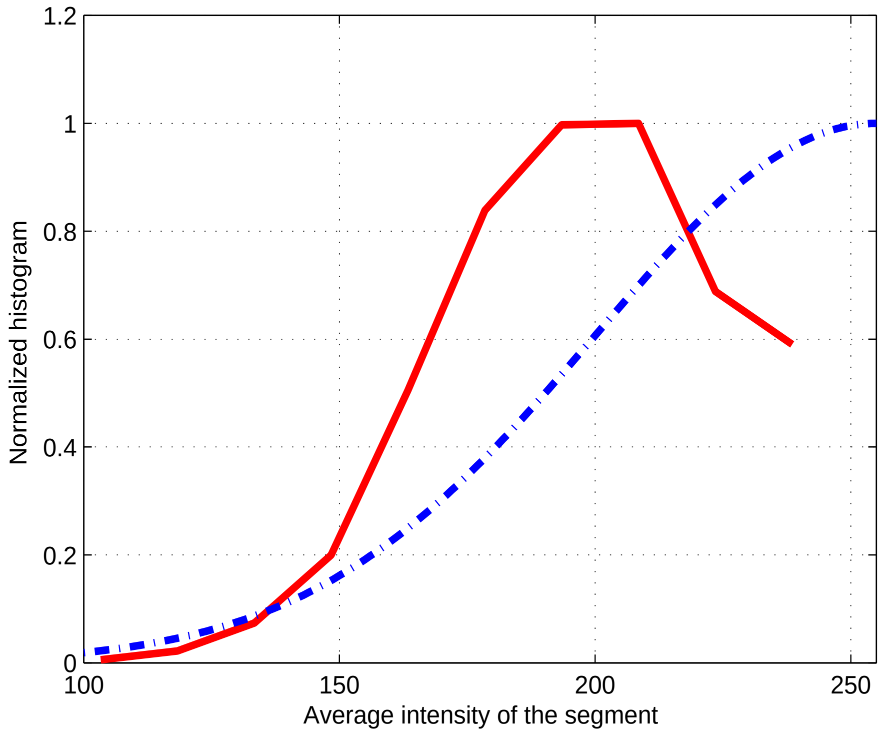

Figure 2.3: The solid (red) line shows the normalized histogram for the average intensity of the network and the dashed (blue) line shows the shape of $e^{\psi(z_1|S)}$ for that intensity value.

the intensities of the segment forming one network. Let I_i be the intensity of the i^{th} segment in the network. The average intensity of the network I_{avg} is given by

$$I_{avg} = \frac{\sum_{i=1}^{n} I_i}{n} \tag{2.11}$$

The graph showing the histogram of average intensity of the network is show in Figure 2.3 by the red curve. The histogram of the average intensity of the network follows a Gaussian distribution with mean of 200 and variance of 55. The initial assumption made about the intensity is the same as Zeng et al. [21]. The roots of the network have a Gaussian profile with the center of the root having the brightest pixels. The intensity of the pixels reduces with a Gaussian shape. By this assumption the number of pixels with intensity close to 255 should be the largest. However from the graph it is seen that this is not the case. The reason for this is discussed in the subsequent part of the chapter. If the distribution obtained from the ground truth is used then pixels that are brighter than the mean value which is 200 in this case will be considered as bad pixels instead of good ones. This is not desirable as this would give the center of the root more energy than the surrounding areas. In order to avoid this the energy or reward function is changed. Instead of giving the maximum reward for networks with average intensity of 200, the mean of the reward or energy function is shifted to 255. The shape of the average intensity energy function is given by the negative logarithm of the dashed curve. The energy function has the lowest value for networks with mean intensity closer to 255. The energy increase with the shape of a Gaussian curve with the variance of 105. The number 110 is the sum of the variance of the Gaussian curve from the histogram and the shift in the center of the histogram.

$$\psi_d\left(z_1|S\right) = \frac{1}{2}\log\left(2\pi\sigma_1^2\right) + \frac{\left(I_{avg}-\mu_i\right)^2}{2\sigma_1^2} \tag{2.12}$$

Here I_{avg} is the mean intensity of the network, μ_i is the mean of the Gaussian curve, which in this case is 255 and σ_1 is the variance of the curve with value of 110.

The second term is called the seedpoint term. In order to calculate this term the image is divided into boxes each of size 11×11. The maximum pixel value in each of the boxes is calculated. The maximum value is considered only if it is above 160. This approach is based on the approach used by [19] to compute the local maximum. An example of the detected seed points is shown in Figure 2.4. Based on the assumption that the center of the root is the brightest part, it is expected that most of the seed points will appear in the center of the root. The histogram showing the distribution of seed point density is shown in Figure 2.5 by the red curve. This term rewards segments based on the number of seed points that are close to the

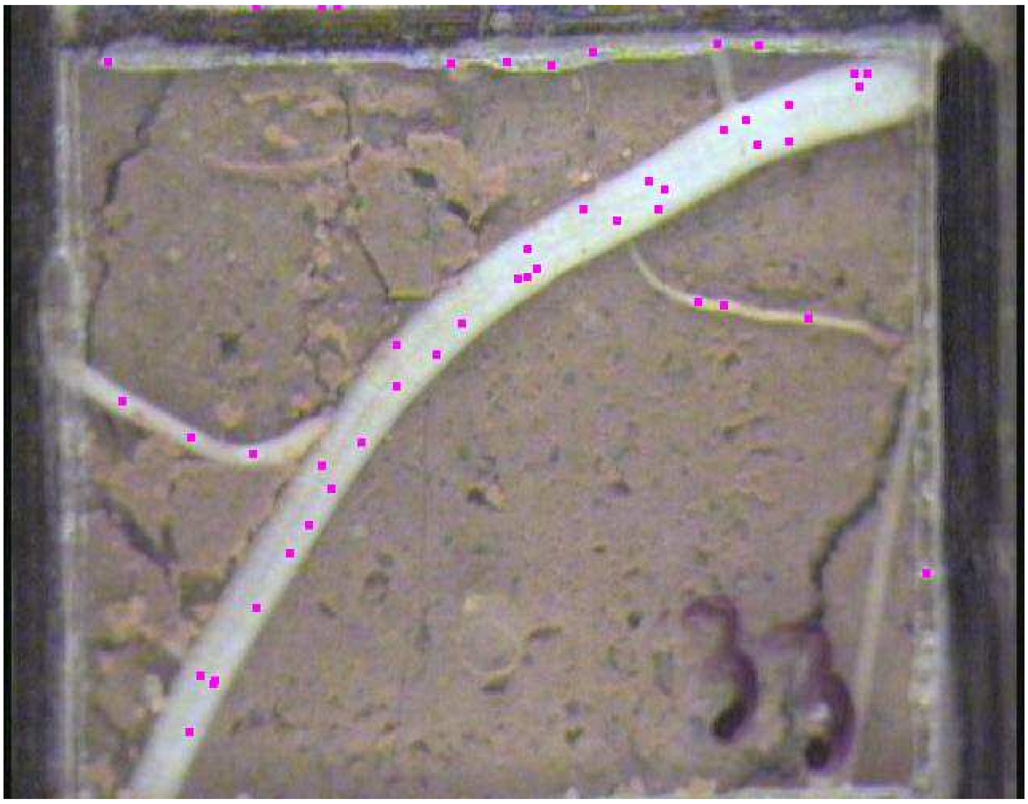

Figure 2.4: This image shows the seed points detected by dividing the image into blocks of 11×11.

root. The energy function used is the negative logarithm of the sigmoid function. It is given mathematically as

$$\psi_d\left(z_2|S\right) = -\log\left(2\right) + \log\left(1 + e^{-\bar{r}}\right), \tag{2.13}$$

where r is the average number of seed points near the segment. The right hand side of equation (2.13) is the negative logarithm of $\frac{1}{1+e^{-r}} - \frac{1}{2}$. The sigmoid function has a value of $\frac{1}{2}$ when r is 0. If the function is used directly, it would say that even if there is not a single point in the neighborhood of the segment there is a 50% chance that that segment lies near the centerline of the root. To avoid this $\frac{1}{2}$ is subtracted from the signum function. If there are no seed points near a segment it has the highest energy. The energy of the segment decreases as the number

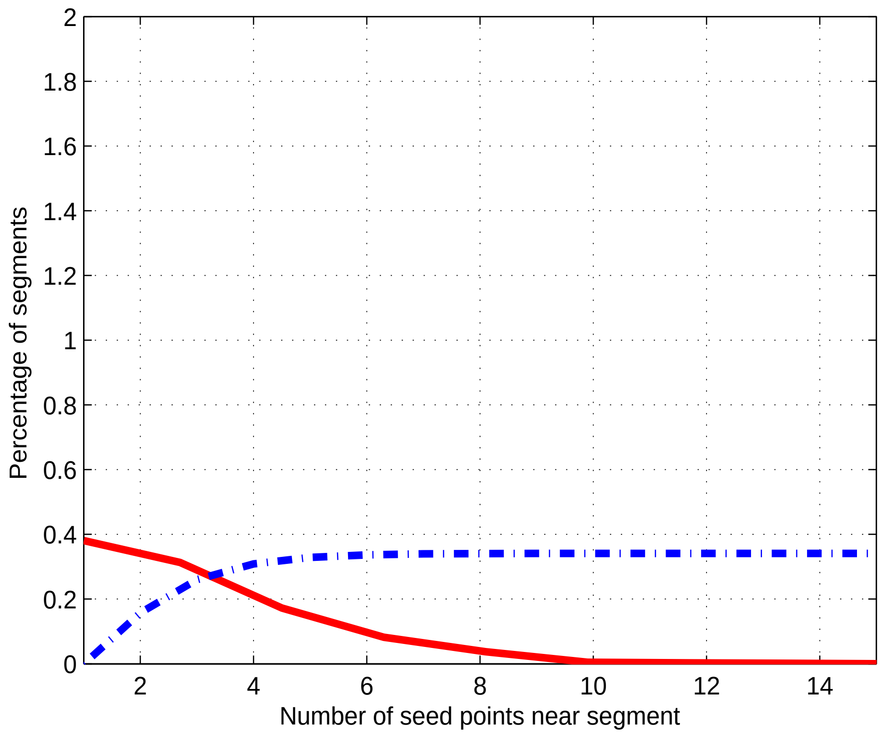

Figure 2.5: The solid (red)line shows the normalized histogram for the average number of seed points near a each segment of the network and the dashed (blue) line shows the shape of $e^{\psi(z_2|S)}$ for that intensity value.

of seed points near the segment increases. The energy saturates after the number of seed points becomes more than six. From Figure 2.5 it is seen that there are very few or no segments that have more than 10 seed points near them. This is due to the length of the segments which are comparatively short as compared to the root. If however a segment has 10 or more seed points near it then there is a high probability that the segment is near the centerline of the root. Due to this such segments have lower energy. The curve saturates because the histogram of the ground truth shows that it is highly likely that a segment will have up to 10 seed points near it. So the energy values after this remain same. The seed point term is calculated by first calculating the average number of seed points close to each segment of the network. This average is then used to calculate the term given in equation (2.13).

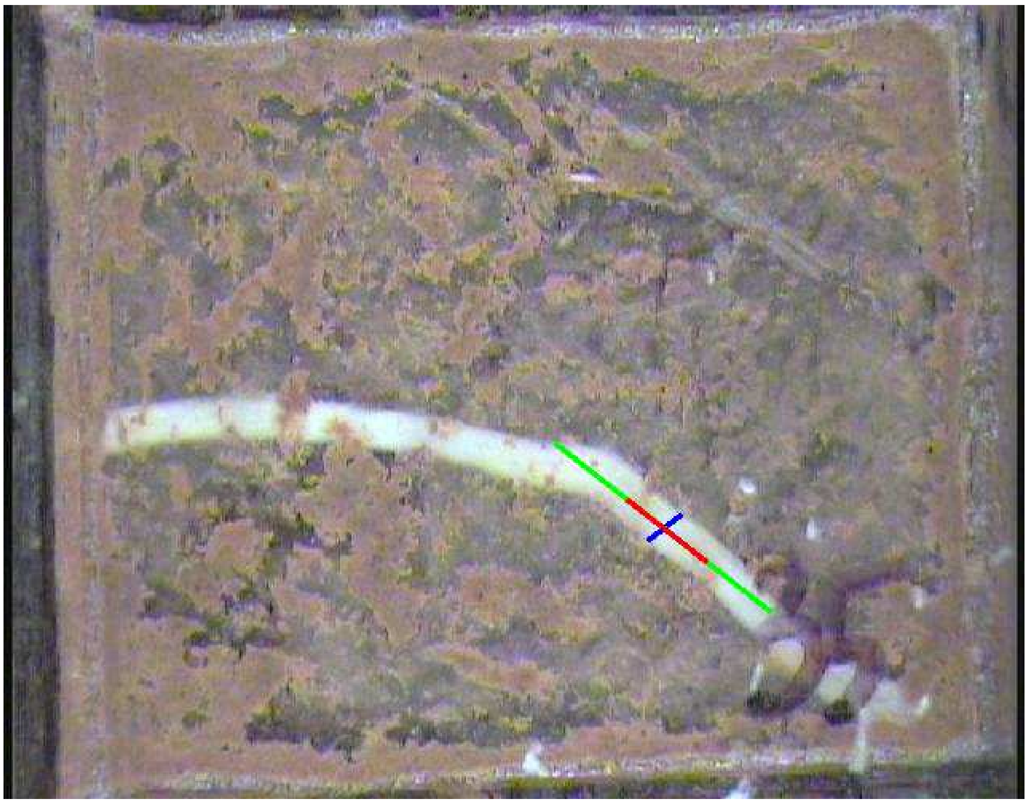

Figure 2.6: This image show how the aspect ratio is calculated. The aspect ratio is the length (in pixels) of green line to the length (in pixels) of blue line.

The third likelihood term is the aspect ratio term, which ensures that the segment has a proper aspect ratio. As shown in Figure 2.6 for a segment that is in the right place the number of root pixels in a perpendicular direction to the root should be less than the number of root pixels in the direction of the segment. The number of pixels on the green segment is the term in direction of the root and the number of pixels in the blue segment is the term for perpendicular direction of the root. In order to compute this term the number of root pixels in a perpendicular direction and the number of pixels in the direction of the root from the center pixel of the segment is calculated. To make a decision whether or not a pixel belongs to a root the intensity of the pixel is used. It is already known that the root pixels are the brightest

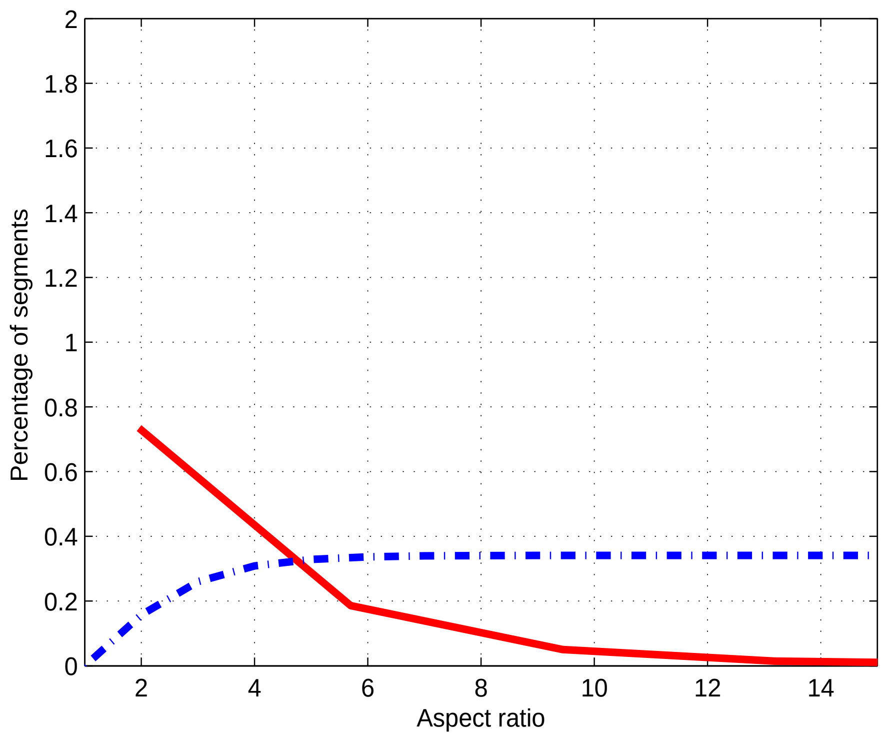

Figure 2.7: The solid (red) line shows the normalized histogram for the aspect ratio for each segment and the dashed (blue) line shows the value of $e^{\psi(z_3|S)}$ for that intensity value

ones in the image. The maximum intensity value that a pixel can have is 255. A pixel is considered as a root pixel if it has an intensity value that is greater than or equal to the 60% of the maximum value. The distance of the first pixel with intensity less than the threshold is calculated from the center of the image. In order to reduce the computation only 100 pixels are scanned at the interval of five pixels. After the computations are done two terms are obtained. One term gives the root length in the direction of the segment and the other term gives the root length in the direction perpendicular to the segment. The ratio of these two terms is taken as the aspect ratio. From the ground truth it was found that this ratio has to be at least one in the worst case. Ideally the ratio should be as high as possible. The function that is used for this term rewards a segment more if it has a higher aspect ratio. The energy function used is similar to the function used for the seed point term. If the histograms for the seed point from

Figure 2.5 and aspect ratio from Figure 2.7 are compared, it is seen that they are similar. This is the reason for using similar energy functions for both terms.

$$\psi_d\left(z_3|S\right) = -\log\left(2\right) + \log\left(1 + e^{-t}\right), \tag{2.14}$$

where t is the aspect ratio. The average aspect of the given network is calculated and used in equation (2.14) to get the aspect ratio term.

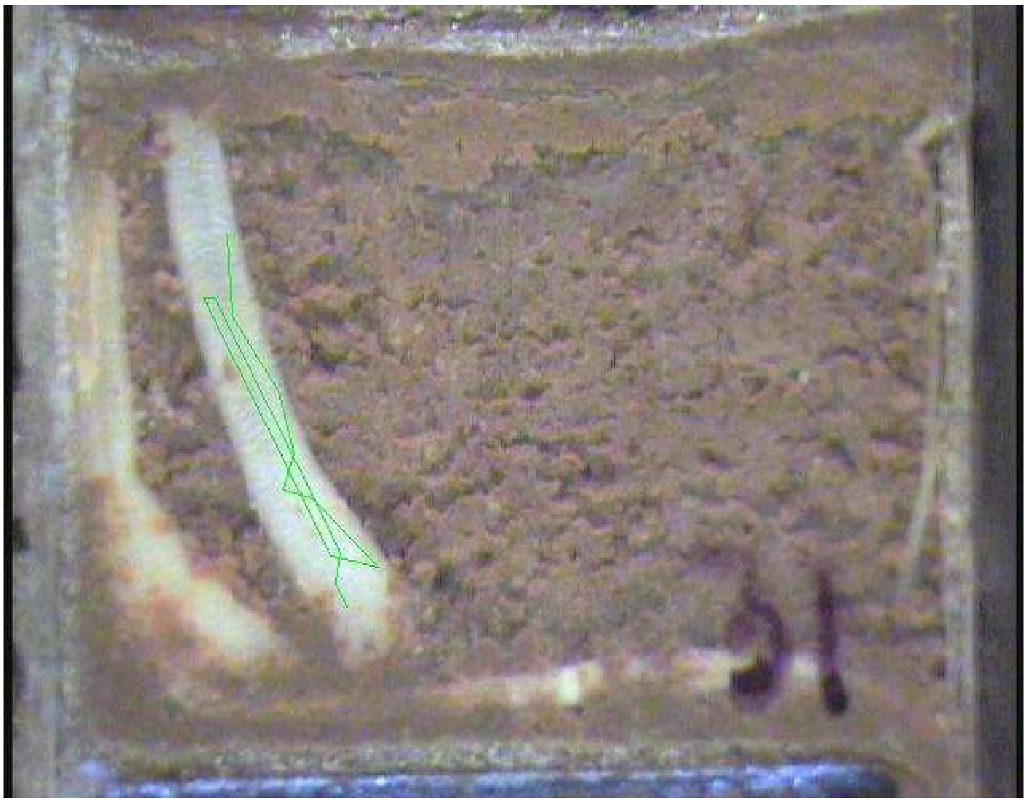

Figure 2.8: This image show the network doubling back on itself if the energy function does not have a term to prevent the line segments from doubling back on network.

The next term in the equation is the sum of the absolute difference of the angle between the segments. This term is considered in order to prevent the line segments to double back on itself. An example of this is shown in Figure 2.8 where the line segments double back on the network. The result shown in Figure 2.8 is obtained by running the complete algorithm without the term to prevent the root from doubling back on itself. Some of the angles between the segments are calculated as negative angles. This causes a problem with the angle energy term as the sum of positive and negative values will result is a smaller angle than the actual angle. To avoid this we calculate the sum of all the absolute values of angles difference of the segments. This sum for a valid linear network should be less than 90 degrees. If the sum goes more than 90 degrees a hard penalty of infinity is added to the energy function. There is no penalty when the sum of the angles is less than 90 degrees.

$$\psi_d\left(z_4|S\right) = \begin{cases} 0 & \text{if angle difference} \leq 90, \\ \infty & \text{otherwise} \end{cases} \tag{2.15}$$

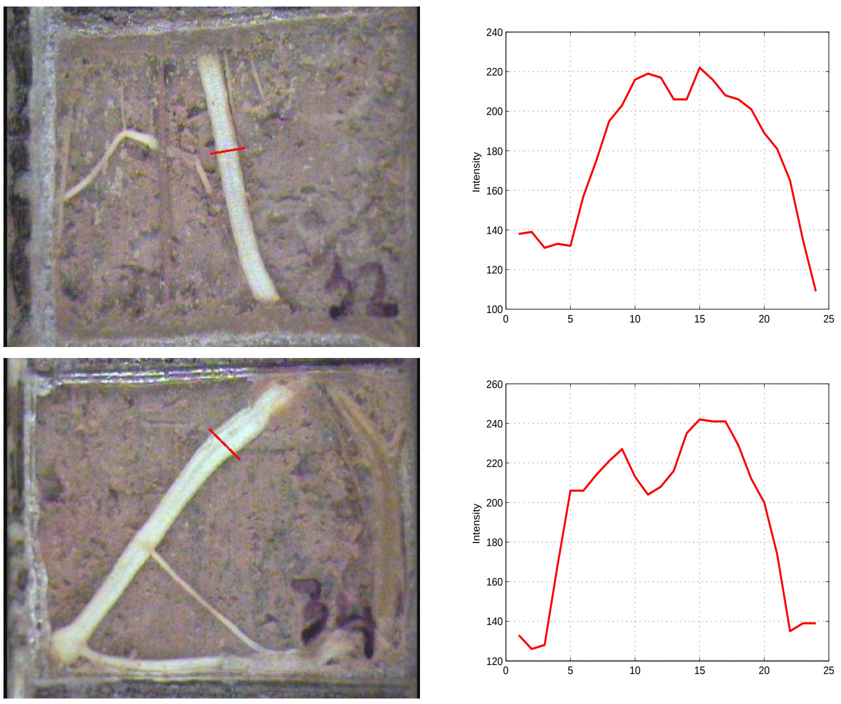

Figure 2.9: Left image: the place at which section of the root is taken, Right image: the graphs showing the intensity profile along the selected line.

In most of the work done on minirhizotron root images, it is assumed that the root profile is Gaussian in nature. The root center has the highest intensity value and the value of the intensity decreases as we move away from the root. While working on developing the energy function it was observed that not all the roots have the above Gaussian intensity profile. In the case of some roots, instead of having a peak at the center they have a local intensity minimum at the center of the root. This is the reason why the intensity histogram from Figure 2.3 has the maximum value at 200 and not near 255 as expected. Although some of the roots have this non Gaussian profile most of the roots still follow the Gaussian intensity assumption made by Zeng et al [19].

In the energy model this term is called Non Gaussian Compensation (NGC). Since the network needs to stay at the center of the root this term is added. Note that it is not possible to completely discard the Gaussian profile assumption as most of the roots still have Gaussian intensity profile. For a root with Gaussian profile this term does not play a major part and can be ignored. For roots with Non Gaussian profile as shown in Figure 2.9, this term makes sure that the network remains at the center of the root. This term is calculated at the two endpoints of the segment. At each of the end points the width of the root on each side of the point in direction perpendicular to the segment is calculated. For the segment to be in the center of the root the width on each side of the point will be equal. The segment receives a penalty if both the widths are not equal. This penalty increases as the difference between the two widths increases. A Gaussian function is used as the shape for the negative exponent of the energy function. As stated this term is calculated at both the ends of the segment. The final output is the mean of both the terms. Since absolute value is used in calculation of the width difference the two differences do not cancel each other out. This term is mathematically given as

$$\psi_d\left(z_5|S\right) = \frac{1}{2}\log\left(2\pi\sigma_2^2\right) + \frac{\left(W_d\right)^2}{2\sigma_2^2}, \tag{2.16}$$

where W_d is the width difference on the two sides of segment. The variance σ_2 is the hypothesized width for the current segment. The hypothesized width on each side is calculated by taking the mean of the widths on both sides. A segment at the center of the root will have minimum value for the energy function. The energy value will increase as the difference in the widths increases. This term is important to keep the linear network at the center of the roots with non-Gaussian profile. This term is not very important for segments with Gaussian intensity profile.

The equation for the data energy term is given as:

$$\begin{aligned} \psi_d\left(Z|S\right) &= \textstyle\sum_{i=1}^{m} \psi_d\left(z_i|S\right) \\ &= \left[\tfrac{1}{2}\log\left(2\pi\sigma_1^2\right) + \tfrac{\left(I_{avg}-\mu_i\right)^2}{2\sigma_1^2}\right] + \left[-\log\left(2\right) + \log\left(1+e^{-r}\right)\right] \\ &\quad + \left[-\log\left(2\right) + \log\left(1+e^{-t}\right)\right] + \psi_d\left(z_4|S\right) \\ &\quad + \left[\tfrac{1}{2}\log\left(2\pi\sigma_2^2\right) + \tfrac{\left(W_d\right)^2}{2\sigma_2^2}\right] \end{aligned} \tag{2.17}$$

Using the equations for prior energy and data energy the equation for the energy of the network is given as:

$$\begin{aligned} \psi(S|Z) &\propto \psi_d(Z|S) + \psi_p(S) \\ &= \left[\tfrac{1}{2}\log\left(2\pi\sigma_1^2\right) + \tfrac{\left(I_{avg}-\mu_i\right)^2}{2\sigma_1^2}\right] + \left[-\log\left(2\right) + \log\left(1+e^{-r}\right)\right] \\ &\quad + \left[-\log\left(2\right) + \log\left(1+e^{-t}\right)\right] + \psi_d\left(z_4|S\right) \\ &\quad + \left[\tfrac{1}{2}\log\left(2\pi\sigma_2^2\right) + \tfrac{\left(W_d\right)^2}{2\sigma_2^2}\right] + \left[\lambda_1\bar{\theta}_{i,i+1}\right] \\ &\quad + \left[-\log\left(\lambda_2\right) - \log\bar{\ell} + \lambda_2\bar{\ell}\right] \end{aligned} \tag{2.18}$$

Chapter 3

Minimizing The Energy Function

The energy function obtained in the previous chapter is minimized using Reversible Jump Markov Chain Monte Carlo (RJMCMC). This approach has been used to simulate a finite point process and reduce the energy in [17]. RJMCMC has also been used for tracking [8] multiple objects. RJMCMC is a generalized form of Markov chain Monte Carlo (MCMC). Although this algorithm takes longer than greedy algorithms used by Zeng et al.[21] it has a higher probability of finding the global energy minimum without getting stuck in local minimum.

3.1 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a general purpose technique used to generate samples from a high dimensional probability distribution function. MCMC consists of two terms Markov chains and Monte Carlo approximation. Markov chains are a way of modeling statistical processes with discrete or continuous state governed by transitional probability. Markov chains follow the Markovian property of localization in time. In every Markov chain the next state of the system depends only on the current state of the system. A Markov chain is usually denoted as (ω, η, κ) for state space, initial state, and transition probability respectively.

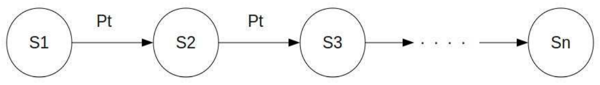

Figure 3.1: A Markov Chain with State $\{S_1, S_2, \ldots, S_n\}$ with state transition probabilities of $\{P_1, P_2, \ldots, P_n\}$.

Monte Carlo approximations is a statistical technique used for sampling. It is often necessary to approximate integrals in high dimensional space. Consider the following integral.

$$C = \int_{\omega} \pi\left(x\right) f\left(x\right)\ \mathrm{dx} \tag{3.1}$$

It is very difficult to solve integral equations like this in high dimensional state space. However the integral can be approximated by using Monte Carlo approximations. This approximation draws N samples from the distribution $\pi\left(x\right)$ and use those samples to approximate the value of C by calculating the sample mean.

$$C = \frac{1}{N} \sum_{i=1}^{N} f\left(x_i\right) \tag{3.2}$$

In Markov Chain Monte Carlo a Markov chain is designed to simulate $\Pi\left(x\right)$ as its stationary distribution. Stationary distribution for a Markov Chain is the distribution that the chain is try-

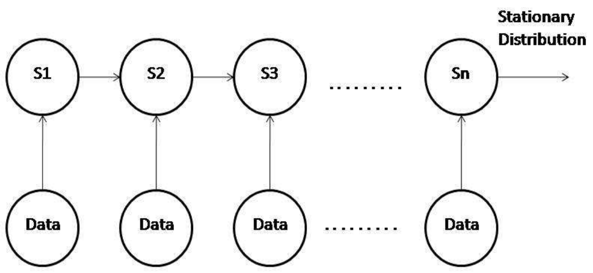

Figure 3.2: A Markov Chain Monte Carlo algorithm used to simulate a distribution. The stationary distribution is the distribution the chain is trying to approximate. $\{S_1, S_2, \ldots, S_n\}$ are the states of the network. The data is collected from the image.

ing to approximate. The distribution $\Pi(x)$ usually exists in a high dimensional space and it is complicated to simulate it. The distribution is approximated as a Markov Chain by extracting N samples from the posterior distribution, hence the name Markov Chain Monte Carlo. One of the popular techniques for sampling is Random Walk. In Random Walk a random point is selected in the solution space and the chain is evaluated at that point. An acceptance ratio is then calculated to decide whether to keep the current state or not. The probability of accepting the next state s' from the current state s is given by

$$\alpha_i(s \to s') = \min\left[1, \frac{f(s)\gamma_i\left(s \to ds'\right)}{f(s')\gamma_i(s' \to ds)}\right] \tag{3.3}$$

Here $\gamma_i\left(s \to ds'\right)$ is the probability with which s transforms to s' and $\gamma_i(s' \to ds)$ is the probability of the reverse move, $f(s)$ and $f(s')$ are the values of the energy of the system in state s and s' respectively. The algorithm accepts some transitions that will increase the energy of the system. This allows the algorithm to get out of local minima and find the global energy minimum. Before the actual MCMC algorithm is implemented there are a few parameters that need to be arbitrarily decided.

1. The Chain Length : The chain length is the length of chain that is needed before it converges to the desired distribution. There is no way to actually decide the length of the chain and most of the time it is empirically decided.

2. The Burning In Period : The second parameter is the burn in period. Some of the initial samples in the chain tend to bias the stationary distribution towards them. In order to avoid this the chain is allowed to run for some time and these samples are discarded. Like in the previous case there is really no way to decide the burn in period and it is arbitrarily decided.

Some of the well known MCMC methods include the Metropolis algorithm and the Metropolis Hastings algorithm. It is possible to simulate distribution in very high dimensional spaces using these algorithms. These methods, however, have one drawback that they are able to simulate distributions having fix dimensions. The classical Metropolis-Hastings algorithm goes as follows

Initialize the Metropolis-Hastings sampler by choosing a random starting state for Markov chain and set the network according to the variables.

- Begin with the starting state X^* of the Markov Chain.

- Propose a new configuration for X^* by sampling a new configuration for the network m^* from the proposal distribution (state evolution).

- Compute the acceptance ratio α_i.

- Add the nth sample to the Markov chain. If $\alpha_i \geq 1$, add the proposed configuration $X^* \to X^n$. If not, add the proposed configuration with probability α_i. If the proposed configuration is rejected, add the previous configuration $X^{n-1} \to X_n$.

3.2 Reversible Jump Markov Chain Monte Carlo

The Reversible Jump Markov chain Monte Carlo algorithm introduced by Green [7] provides a way to sample a state space with varying dimensions. Both RJMCMC and other MCMC methods perform random walk through the state space. The main difference between RJMCMC and other MCMC methods is that RJMCMC allows for moves that change the dimension of the space. There is however one restriction in RJMCMC: for every move that is proposed which changes the dimension of the problem there must a move that restores the original dimensions. The dimension changing moves always occur in complementary pairs. If one move increases the dimension then its compliment decreases the dimensions. The acceptance ratio for RJMCMC is given by

$$\alpha_i \left(s \to s^{'}\right) = \min\left[1, \frac{f\left(s\right)\gamma_i\left(s \to ds^{'}\right)}{f\left(s^{'}\right)\gamma_i\left(s^{'} \to ds\right)}\left|\frac{\delta\varphi}{\delta\left(s,u\right)}\right|\right] \tag{3.4}$$

The acceptance ratio for RJMCMC has an additional Jacobian term absent in normal MCMC. This term is due to change in dimensions of the state space. The Jacobian term evaluates to unity as dimension changing moves are always invertible. Metropolis-Hasting algorithm is a special case of RJMCMC, where all the moves have same dimensions. The algorithm for RJMCMC goes as follows.

Initialize the RJMCMC sampler by choosing a random state for the Markov chain.

- Begin with the starting state X^* of the Markov Chain.

- Choose a move type from the set of possible move types.
- Apply the chosen move. This involves proposing a new configuration X_n.
- Compute the acceptance ratio α_i, keeping in mind it is defined differently for the various move types.
- Add the nth sample to the Markov chain. If α_i = 1, add the proposed configuration $X^* \rightarrow X_n$. If not, add the proposed configuration with probability α_i. If the proposed configuration is rejected, add the previous configuration $X_{n-1} \rightarrow X_n$.

The algorithm starts with a single point as the first state of the linear network. The way that the single point is determined is explained in Chapter 4. The algorithm then selects a move randomly for state evolution. The new state for the network is determined with the help of the proposal density $Q(X_n, X^*)$. This proposal density is defined differently for each move. As every move needs to have a reversible move there is also a reversible proposal density given by $Q^{'}(X^*, X_n)$. Any move that changes the dimensions of the problem is called a 'jump'. The proposal densities for each move are discussed in the next section. After getting the next state of the network, the acceptance ratio α_i is calculated as

$$\alpha_i = \min\Big[1, \frac{f\left(s^{'}\right) p_m Q\left(X^*, X_n\right)}{f\left(s\right) p_m^{'} Q\left(X_n, X^*\right)}\Big], \tag{3.5}$$

where $p_m^{'}$ is the probability that a given move is selected and p_m is the probability of the corresponding reverse move. In this algorithm each of the moves is selected with equal probability. The terms $p_m^{'}$ and p_m are constants throughout and can be ignored.

The above explanation of both MCMC and RJMCMC is written with help from [13].

3.3 Moves for RJMCMC

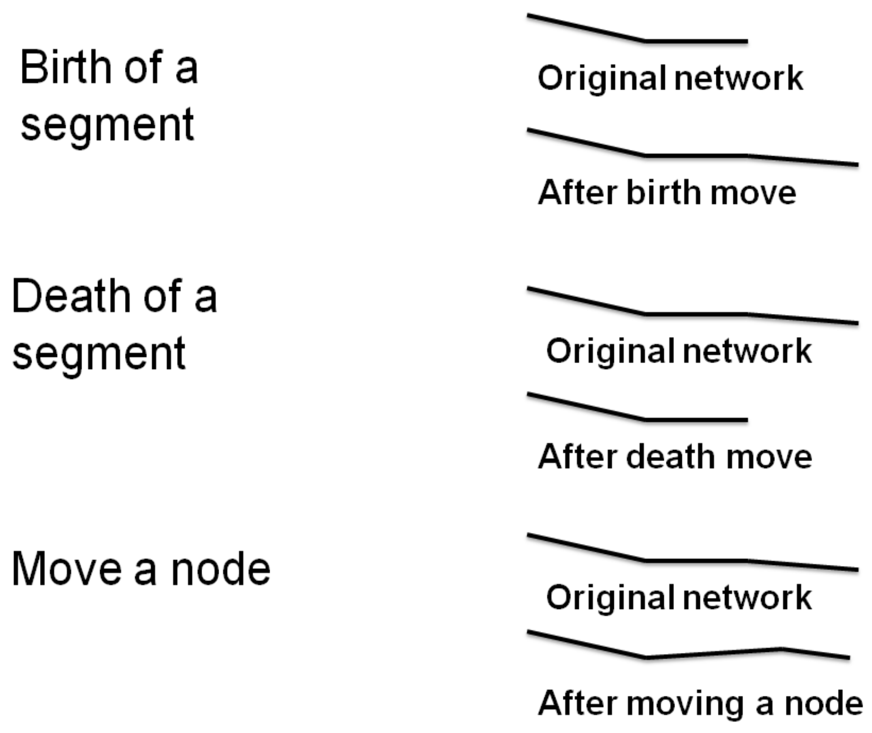

Figure 3.3: This figure shows the three moves that are used in RJMCMC and the effect of these moves on the linear network.

As stated in the RJMCMC algorithm it is necessary to have a set of moves that will change the state of the network. For every move that changes the dimension of the problem there must be a move that changes it back. In this work three moves are used as shown in Figure (3.3). For proposing moves the network is considered as a collection of connected points instead of collection of lines. The size of the network is defined as the total number of end points of the segments. As defined previously the number of line segments is n, so the number of end points is $n + 1$. This allows the moves proposed to be simpler than the moves for a collection of lines. The moves that change the dimension of the state space are birth of a node and death of a node. Move a node does not change the dimensions of the state space. These moves allow the network to grow, shrink and change its shape. These moves allow the network to start as a single point and detect the entire root. Let k be the next iteration. The details of the moves are as follows

1. Birth of a node : In this move we add a node to the network. This is equivalent to adding a segment with single connection. Right now our aim is to trace the main part of the root and not the branches so the nodes are added only at the ends of the network. In this move a new point is added to either end of the segment. The proposal is used to select the node to which the newly added point has to be connected. As there are only two nodes the probability that any one node is selected is $\frac{1}{2}$. If only one node is present then that is the only node that can be selected.

$$Q\left(X_k, X^*\right) = \begin{cases} \frac{1}{2} & \text{node 1 is selected} \\ \frac{1}{2} & \text{node n+1 is selected} \\ 1 & \text{if only 1 node present} \end{cases} \tag{3.6}$$

   In order to speed up the process it is necessary to propose the new state in such a way that majority of the proposed moves will be accepted. To propose such points the prior information about the appearance of the roots is used. It is known from the shape of the root that there will not be a sharp bend in the root and also the distribution for the length of the segment. The orientation and the length of the new segment are randomly sampled from distributions that are similar to but not exactly the same as the histograms obtained in chapter 2. The value of the energy function is evaluated for this new state and the acceptance ratio is calculated.

   Let $n_b(S)$ be the number of points in the network after the birth move and $n_d(S)$ be the number of points after death move. P_b and P_d are the probabilities of birth and death respectively. $\psi(S)$ and $\psi(S')$ are the energy functions for the current move and proposed move respectively. The acceptance ratio is then given by:

$$\alpha_i = \min\Big[1, \frac{\psi(S)P_b Q\left(X^*, X_k\right)}{\psi(S')P_d Q\left(X_k, X^*\right)}\Big]. \tag{3.7}$$

2. Death of a node : The second move that is proposed is the death of a node. This is the reverse move for the birth of a node. As birth increases the dimension of the space this decreases the dimension of the state space. A node is randomly selected from the existing network and deleted. The proposal for this move can be written as

$$Q\left(X_k, X^*\right) = \begin{cases} \frac{1}{n+1} & \text{if size of the network} \geq 2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

The energy of the new network is then evaluated for the new state and the acceptance ratio is calculated. By using the notation used above the acceptance ratio for this move is given as

$$\alpha_i = \min\Big[1, \frac{\psi(S)P_d Q\left(X^*, X_k\right)}{\psi(S')P_b Q\left(X_k, X^*\right)}\Big]. \tag{3.9}$$

3. Move a node : The third move is move a node. In this move a node from the network is randomly selected and moved randomly. The proposal for moving this move is

$$Q\left(X_k, X^*\right) = \begin{cases} \frac{1}{n+1} & \text{if size of the network} \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The restriction on moving the node is that it can only move the node in a circle of 25 pixels around it. This move does not change the dimensions of the network hence no reverse move is needed for it. The acceptance ratio for this move is

$$\alpha_i = \min\left[1, \frac{\psi(S)Q\left(X^*, X_k\right)}{\psi(S')Q\left(X_k, X^*\right)}\right] \tag{3.11}$$

The calculation of the acceptance ratio for this move is the same as normal MCMC. This move does not change the dimension of the state space this reduces RJMCMC to MCMC.

At every iteration of the algorithm one of the above move is used to determine the next state of the network. This algorithm requires just three moves. The reason that the moves are simple because the network is considered as a collection of points and not as lines. If the network is considered as a collection of lines as done in [17] then it will add extra moves like add a singly connected segment and add a doubly connected segment. However since only the main stem of the roots is detected these moves are unnecessary.

Chapter 4

Algorithm

Previous chapters describe how to calculate the energy of the network and how to minimize. In order to calculate the energy it is necessary to initialize the network in the image. The energy function and the minimization algorithm can detect a single root. Most of the images contain multiple roots. An initialization algorithm is needed to initialize the network on the root and detect if there are multiple roots in the image. If there are multiple roots in the image then the network need to be initialized multiple times . Each initialization of the network will detect one root in the image.

To propose possible locations for the roots the idea of local maxima is used again. Since the roots are brighter than the background most of the maxima will be on or near the roots in the image. To find the local maxima the image is divided into 21×21 blocks. The local maximum is calculated for each of the blocks. The local maximum is not considered if its intensity is less than 60% of 255. This process is similar to the one for calculating the seed points. The major difference is the size of block in which the image is divided. This step is performed only once while the calculations for the seed point terms are made in every iteration. A list of only these maximums is created. Doing this throws more than 85% of the data and makes furthur calculations faster. However just finding the local maxima and initializing them on one of them does not work. In some of the images there are bright blobs on the background that are not part of any root. If the network is initialized on these blobs then it will give false positives. In order to make sure that the network is not initialized at a background maximum an additional test is run. The pixels on the root are surrounded by bright pixels. The neighborhood of these pixels will have brighter intensity than background seed points. A point is randomly selected from the list of maxima and the neighborhood of 100 pixels around that pixel is scanned in $0, 90, 180$ and 270 degrees. The reference that is 0 degrees is the positive x axis of the image coordinate system. The average intensity of in each direction is calculated. If the minimum average intensity is greater than 127 (half of 255), the pixel is selected as the seed point. If the first selected pixel does not meet the criteria then it is deleted from the list of maxima and another point is selected. This process is repeated till all the maxima are eliminated.

The network is initialized at the point that meets the criteria. The energy of the network is initialized to ∞. During the initial stages of the network there are constraints on which moves can or cannot be used. Till the network has at least two points it is not possible to use death and move a node mode. So the only move allowed is birth of a node. It is also possible to initialize the network as a line instead of a point. In this case it is necessary for the initialization algorithm to determine a line instead of a point. This will add more computations unnecessarily. The energy minimization algorithm takes care of finding the optimal second point. The RJMCMC algorithm is then used to minimized the energy of the network. The network changes its state as described by the moves for the RJMCMC. The number of states in the Markov Chain in which the stationary distribution is reached is empirically determined. It was observed that a single root is detected within 3000 iterations. After the minimization algorithm completes, the network is stored. The energy of the network at each step is also stored. After a network is completed, any local maximums that are within a 30 pixel range are removed from the list of maxima. This ensures that the network is not initialized at the same place.

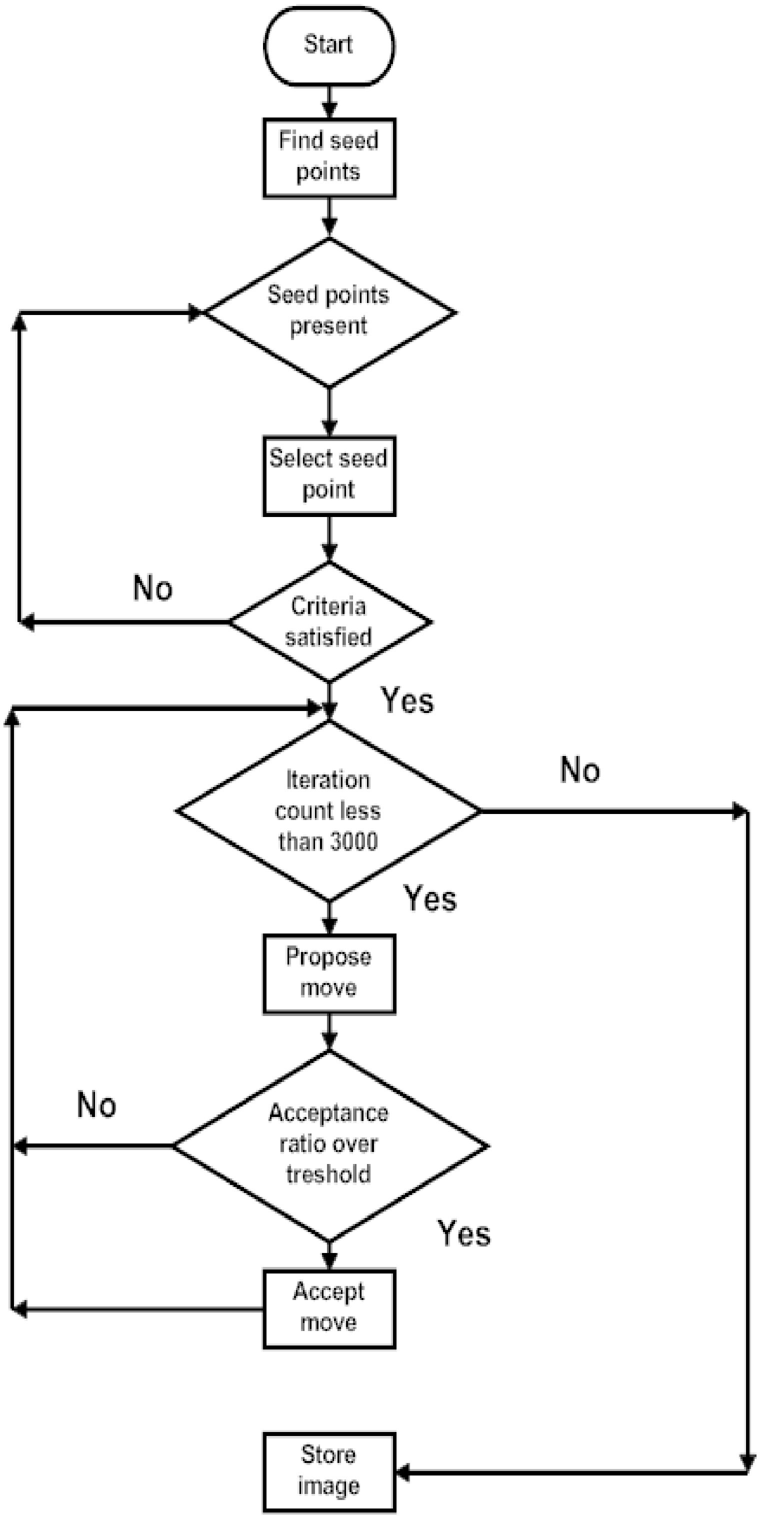

Figure 4.1: Flowchart showing the entire process of detection of roots.

Chapter 5

Experiments And Results

In order to show the effectiveness of the algorithm it was run on a data set of 150 images of Peach tree roots. Some of the results of the algorithm are shown in Figure 5.1, 5.2, 5.3 and 5.4. Most of the roots shown in the results have a Gaussian intensity profile. Some images in Figure 5.3 have a non-Gaussian profile. The results show that the algorithm is effective for roots having a non-Gaussian intensity profile. The results show that the algorithm is able to detect multiple roots, roots that cross each other, and roots that are close to each other. The algorithm is also able to detect roots that are comparatively darker than other roots and roots that have mud blobs on them.

Some of the images have structures near the image boundaries that have the characteristics of roots, but are not roots. The initialization algorithm initializes the network at these structures and the energy minimization algorithm fits a piecewise linear network at the centerline of these structures. Most of the false positives detected by the algorithm are due to these structures at the end of the images. Results shown in Figure 5.2 and 5.4 have structures which produce false positives. Some of the results shown in Figure 5.4 also have false negatives. In these cases the contrast between the roots and the background is low. The initialization algorithm is not able to initialize the networks for such an image. If the network is manually initialized on the root then the algorithm is able to detect the centerline of these roots. A possible solution for this problem will be to do some preprocessing using contrast stretching. The last result in Figure 5.4 shows an image which has both false positive and false negative. The graphs showing the minimization of the energy function as the algorithm progress are shown in Figure 5.5.

In the Chapter 3, we state that the chain length or the number of iterations for which the algorithm runs is empirically decided. Usually of the times the algorithm is able to detect the roots in 3000 iterations. For most of the roots the energy function reaches its minimum value well before 3000 iterations as seen from the graphs in Figure 5.5. There are some cases as shown in Figure 5.6 when the number of iterations selected is not enough to detect the entire root.

Figures 5.7,5.8 and 5.9 show the comparison of the proposed algorithm with the algorithm implemented by Zeng et al. Figure 5.7 shows cases in which the proposed algorithm detects roots that the method proposed by Zeng et al. fails. Figure 5.8 shows images in which the proposed algorithm detects some of the roots missed by Zeng et al. Figure 5.9 shows that the proposed algorithm does a better job at keeping the linear network near the center of the root.

The last output in Figure 5.8 shows an interesting case. This image has 5 roots in it. The Zeng et al. method is able to detect two roots and the proposed algorithm is able to detect three roots. There is one root detected by Zeng et al. that is missed by the proposed algorithm. This image has roots that cross each other. The horizontal root at the center of the image has the value of energy function that is lower than the value of the third vertical root. The energy values of each root are given in Figure 5.10. The values of the numbers is not important but rather the difference between the two value is important. The difference between the energy values of both the roots shown in Figure 5.10 is such that, even if the network is initialized on the vertical root it still detects the horizontal root. This is an interesting case in which the characteristics of the image are a cause for the failure of both the energy function and the minimization algorithm.

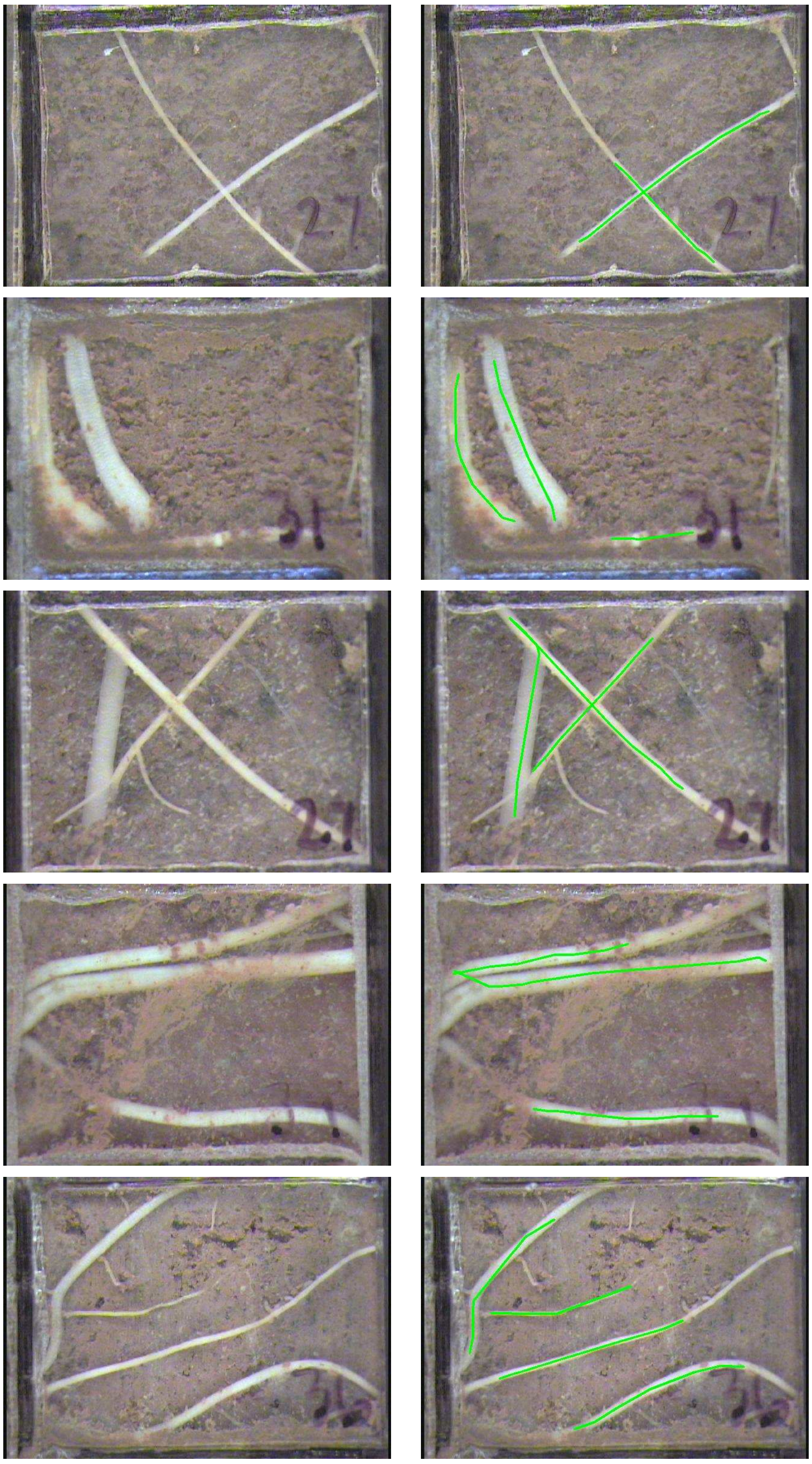

Figure 5.1: Left image: the original images. Right image: the detected roots.

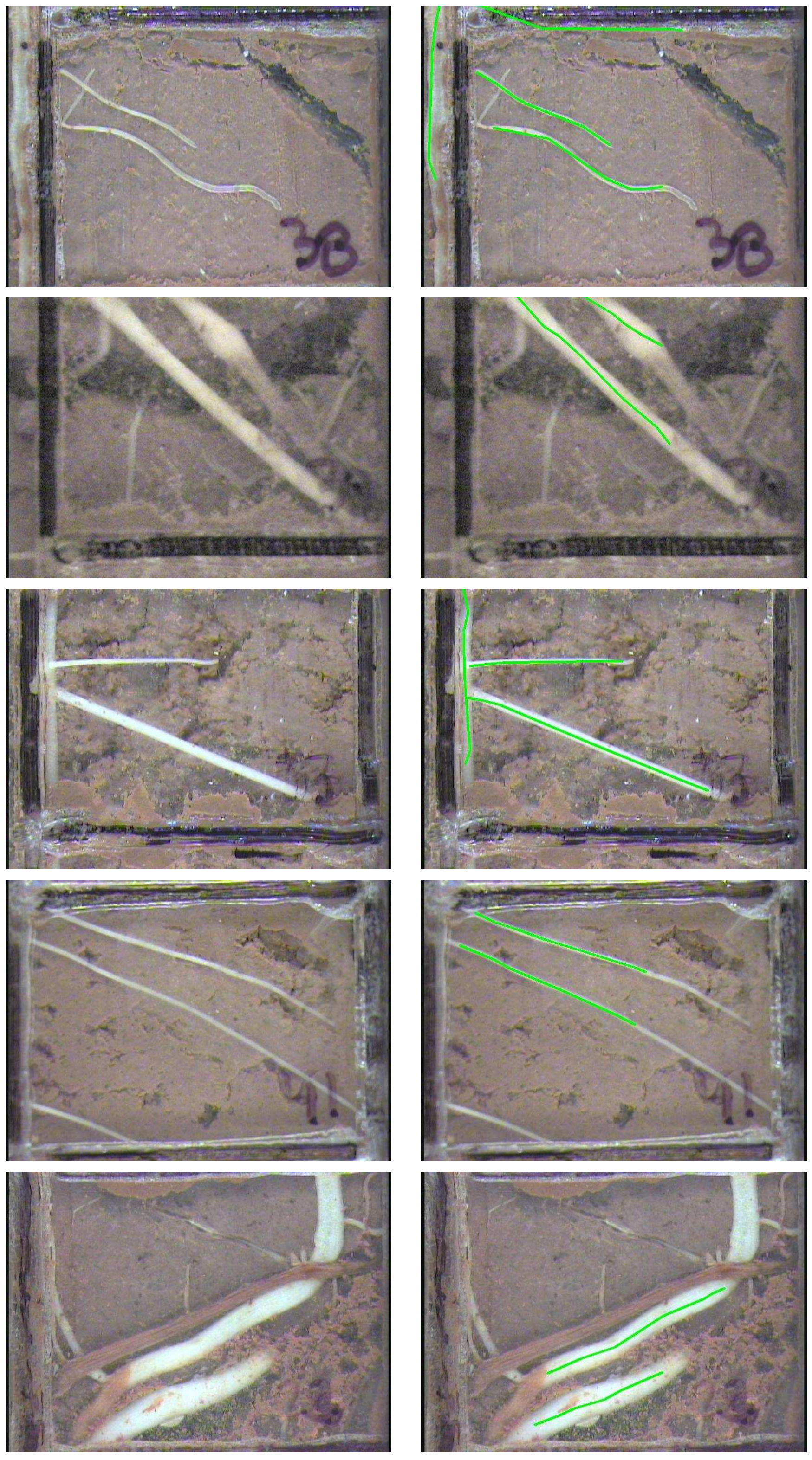

Figure 5.2: Left image: the original images. Right image: the detected roots.

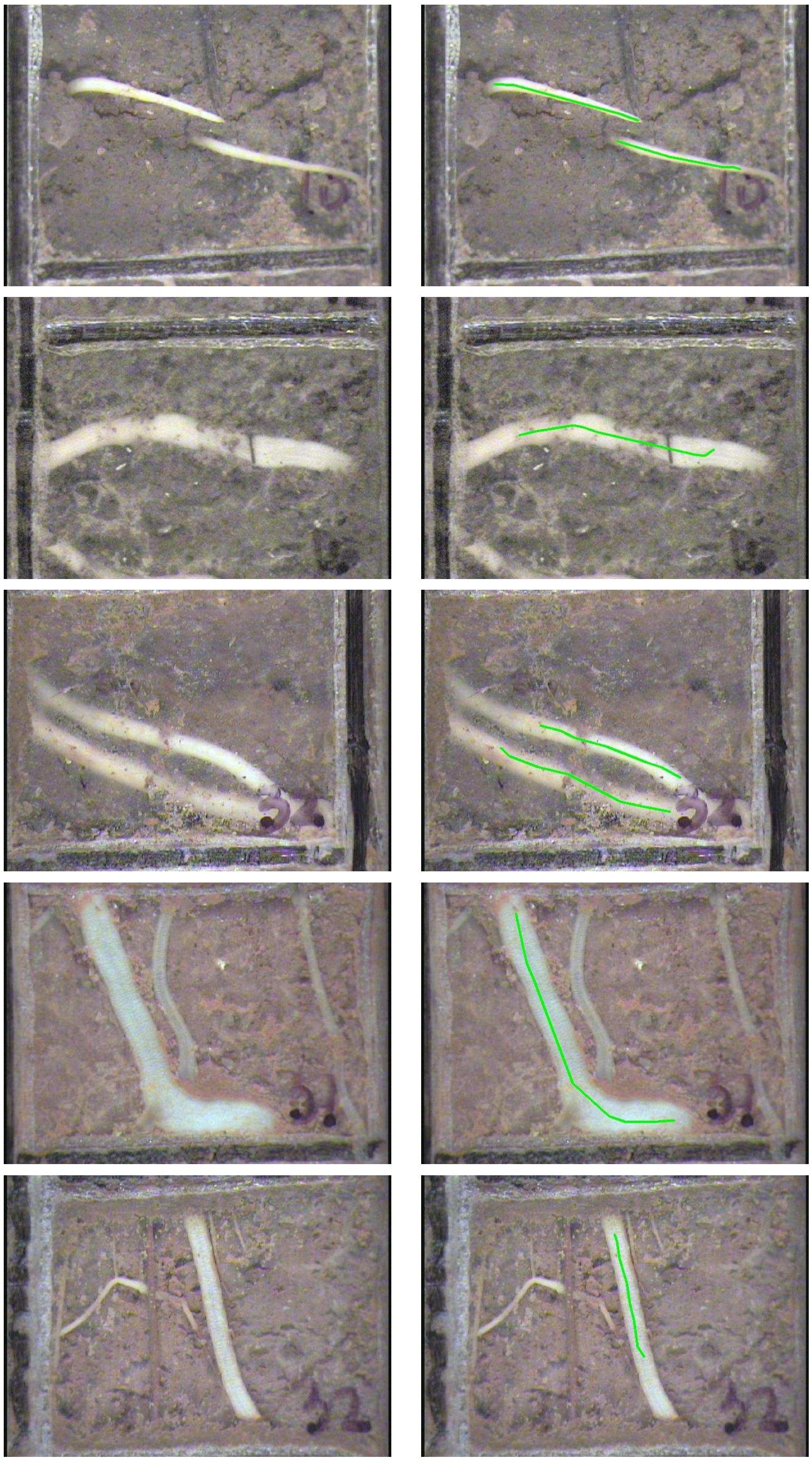

Figure 5.3: Left image: the original images. Right image: the detected roots.

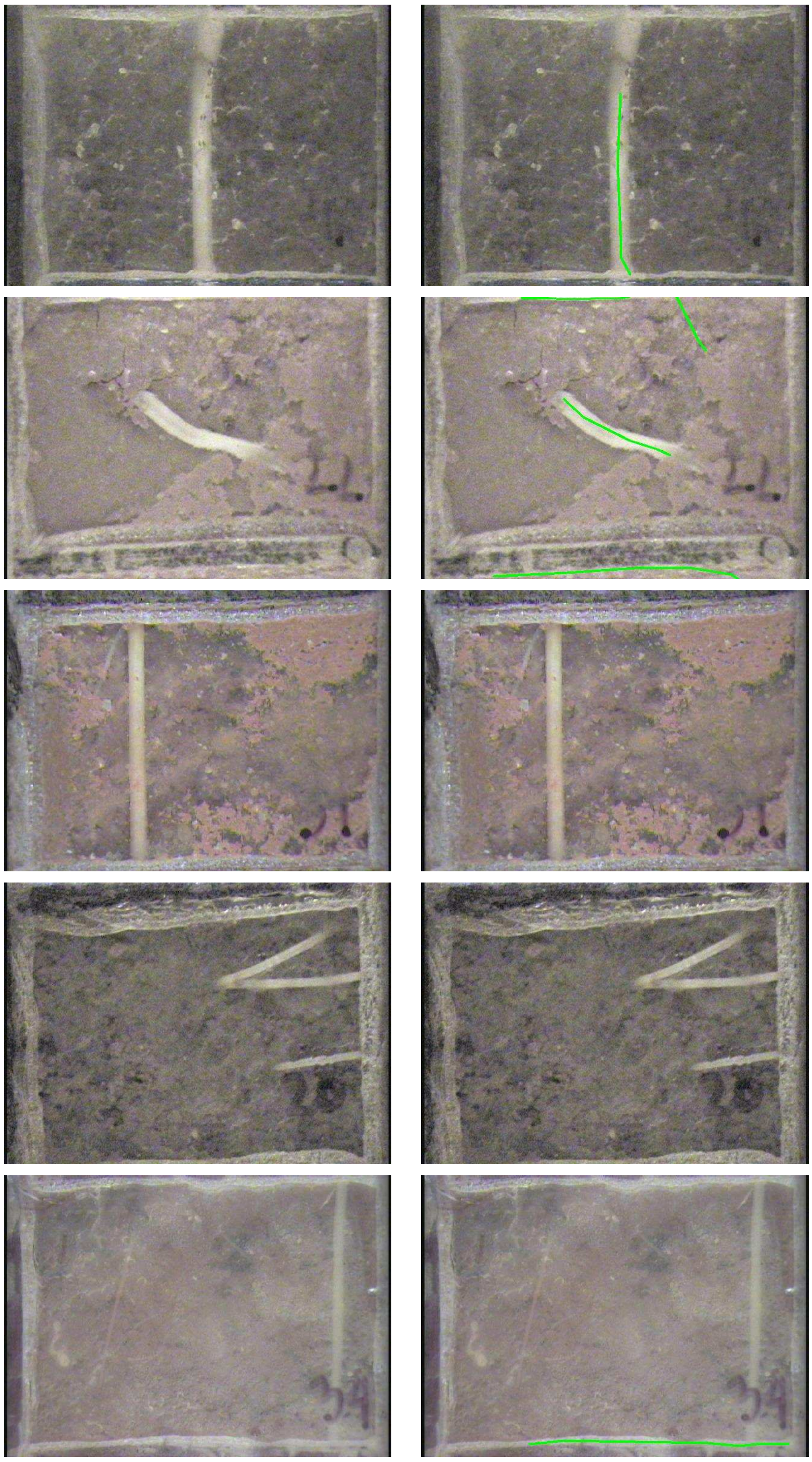

Figure 5.4: Left image: the original images. Right image: the detected roots.

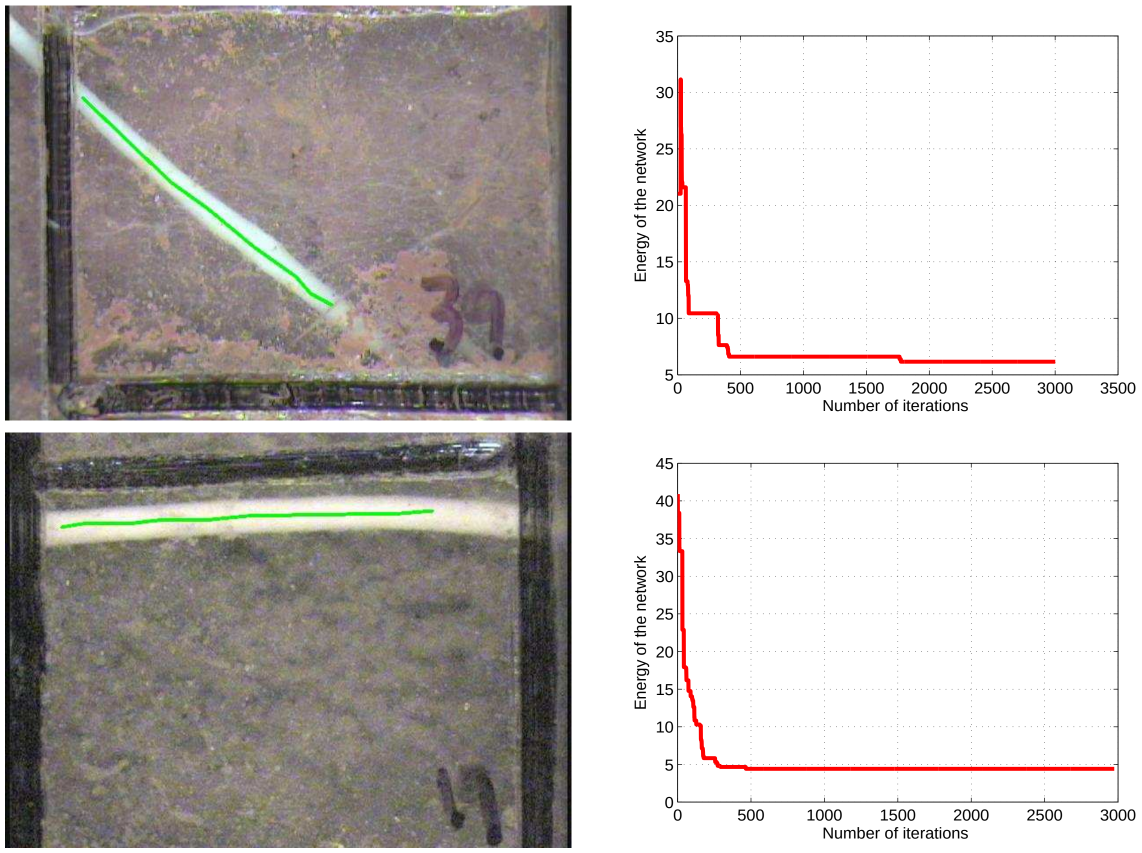

Figure 5.5: Left image: the roots detected by the algorithm, Right image: the graph of energy value over 3000 iterations .

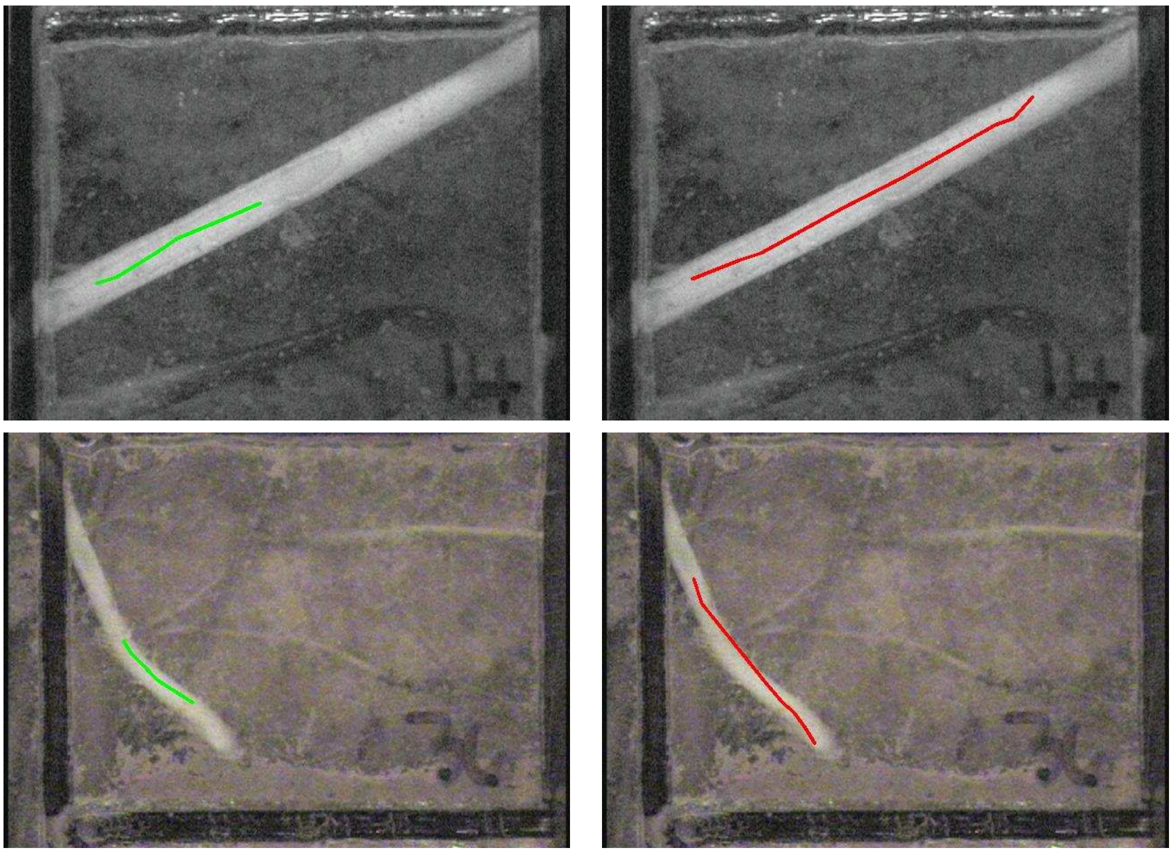
Figure 5.6: Left image: root detected in 3000 iterations, Right image: root detected in 5000 iterations .

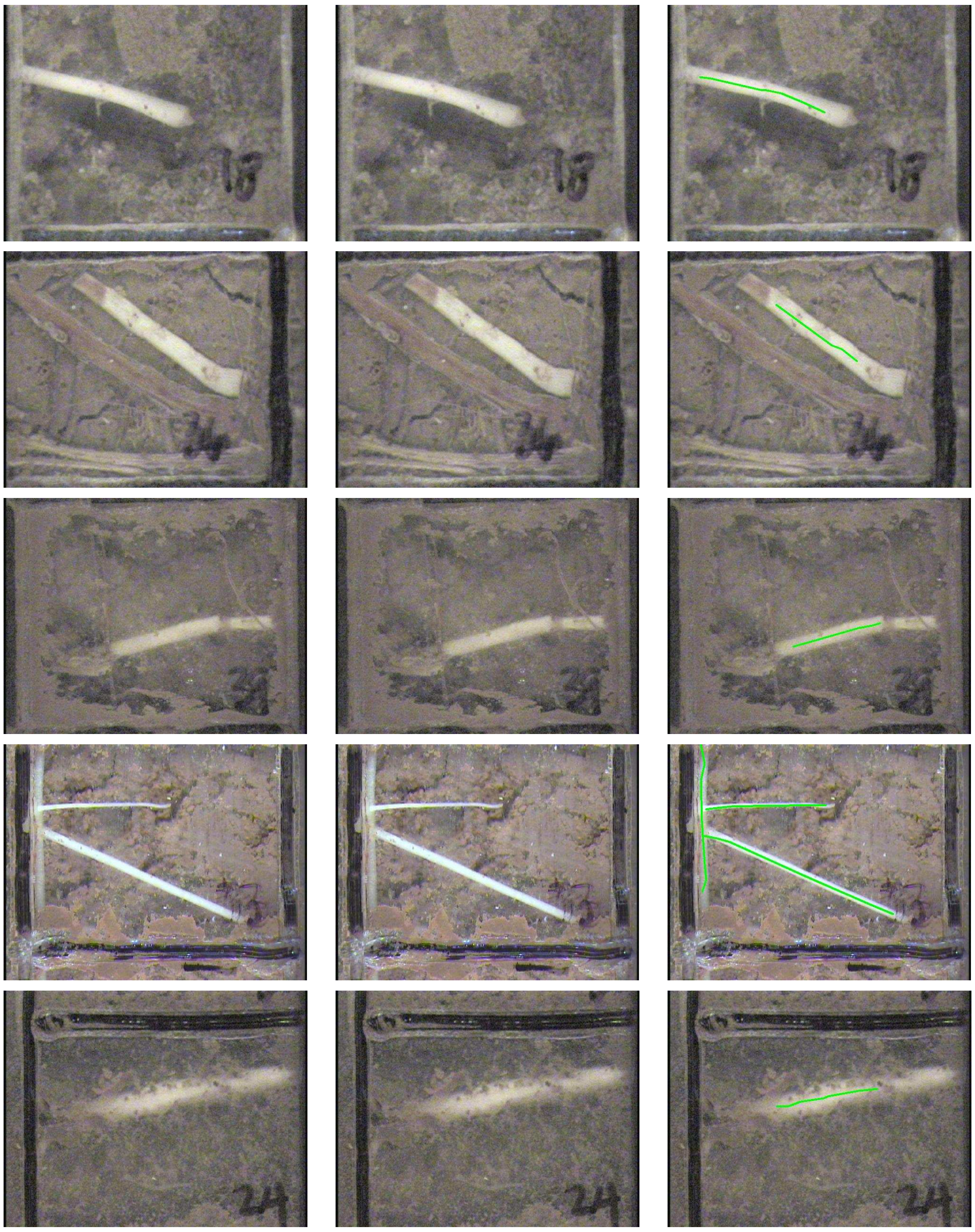

Figure 5.7: Right image: the original image. Center image: results of Zeng et al. Right image: results of proposed algorithm.

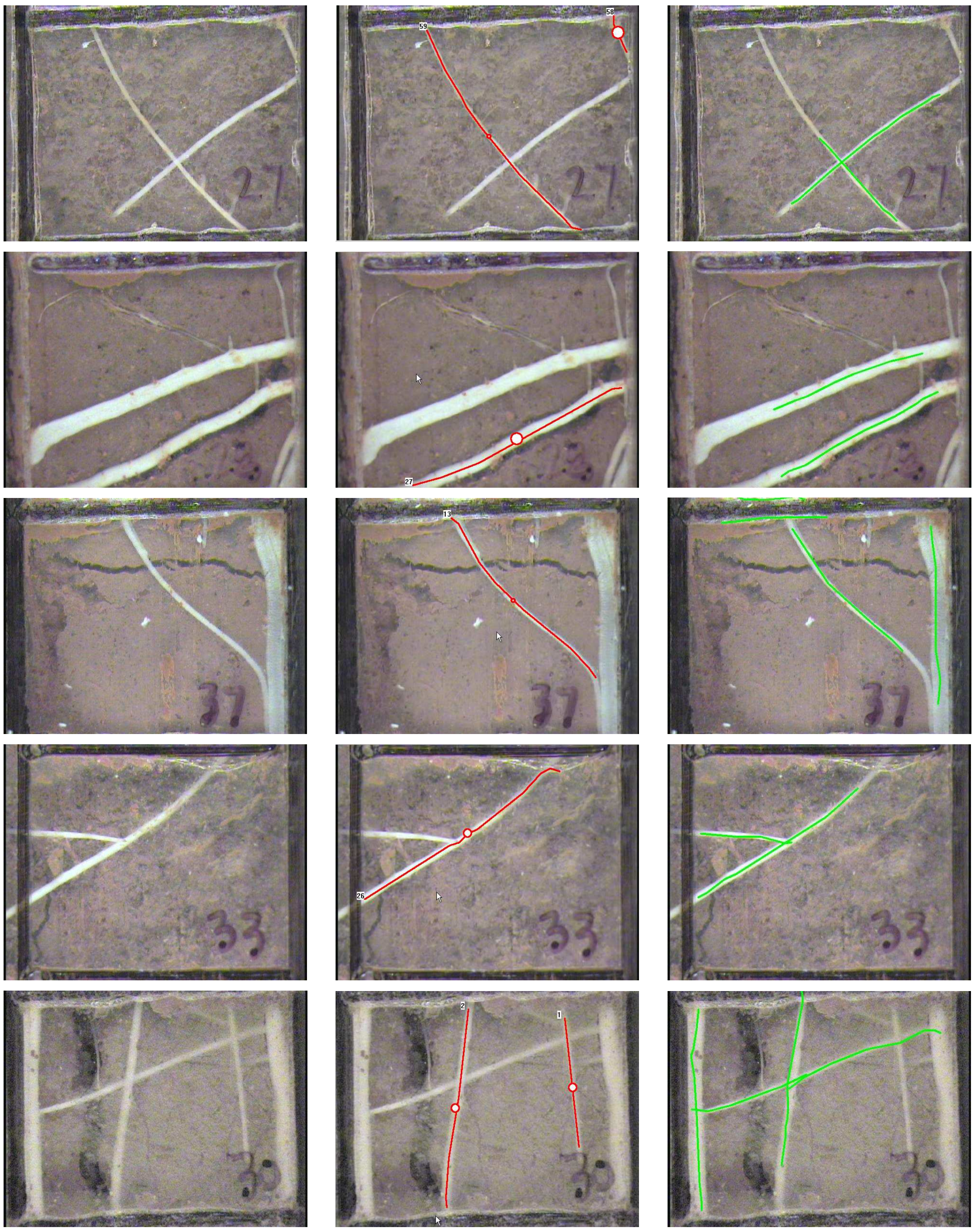

Figure 5.8: Right image: the original image. Center image: results of Zeng et al. Right image: results of proposed algorithm.

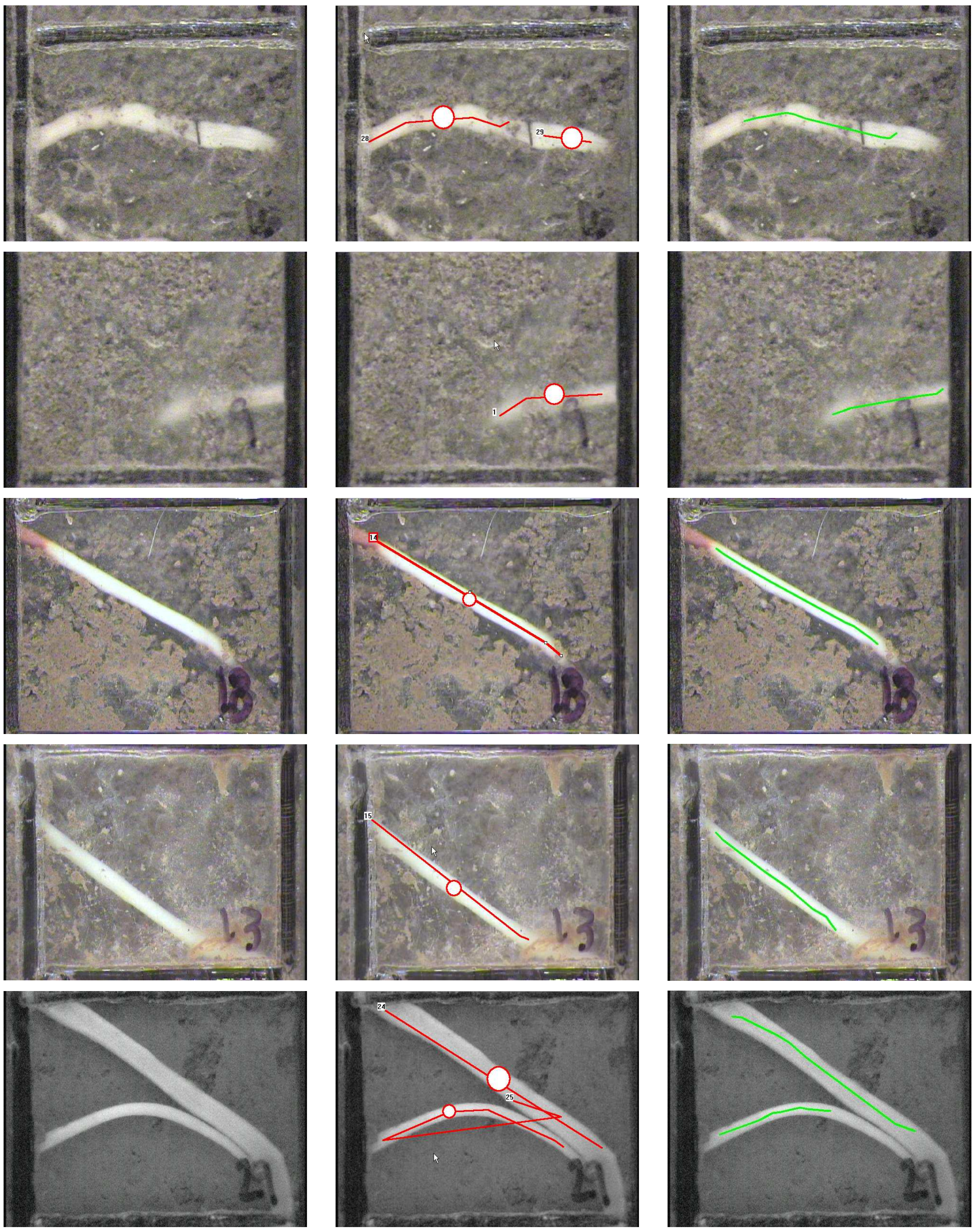

Figure 5.9: Right image: the original image. Center image: results of Zeng et al. Right image: results of proposed algorithm.

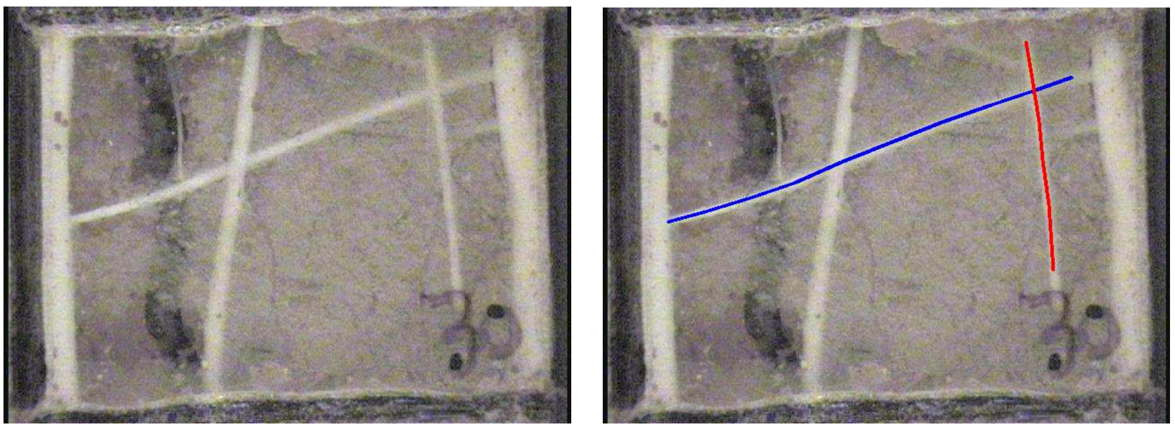

Figure 5.10: Left image: Original image. Right image: Two crossing roots. The algorithm detects the horizontal (blue) root but fails to detect the vertical (red) root. Horizontal root has energy of 11.732 and vertical root has energy value of 23.589.

Chapter 6

Results For Road Detection

The algorithm proposed in this work has been developed using the characteristics of roots from a minirhizotron camera image. Road images obtained from a satellite have similar characteristics. It is possible to modify the energy function by a small amount to detect roads. The energy function that was used to detect roots has a weight of 1 for all the terms. To detect roads it is necessary to tweak the energy function by changing the values of the weights. These weights were empirically determined. The energy function to detect roads is

$$\begin{aligned}\psi(S|Z) \quad \propto \quad & w_1\lambda_1\theta - w_2\log(\lambda_2) - w_2\log l + w_2\lambda_2 l + \tfrac{w_3}{2}\log(2\pi\sigma_1^2) \\ & + \tfrac{w_3(I-\mu_i)^2}{2\sigma_1^2} - (w_4)\log(2) + (w_4)\log(1+e^{-r}) \\ & - (w_5)\log(2) + (w_5)\log(1+e^{-t}) + \tfrac{w_6}{2}\log(2\pi\sigma_2^2) + \tfrac{w_6(W_d)^2}{2\sigma_2^2}\end{aligned} \tag{6.1}$$

In the above equation $w_1 = 1, w_2 = 1, w_3 = 3, w_4 = 1, w_5 = 1, w_6 = 1$. This energy function is minimized using the same algorithm used for roots. The initialization algorithm that was used for roots fails in most of the cases for roads. The background for the road images is different than roots. In case of roads there are spots in the background that satisfy the conditions that are used to determine the points of initialization. The results shown in this section are obtained by manually initializing the network on the roads.

The images used for this part were taken using Google Earth.

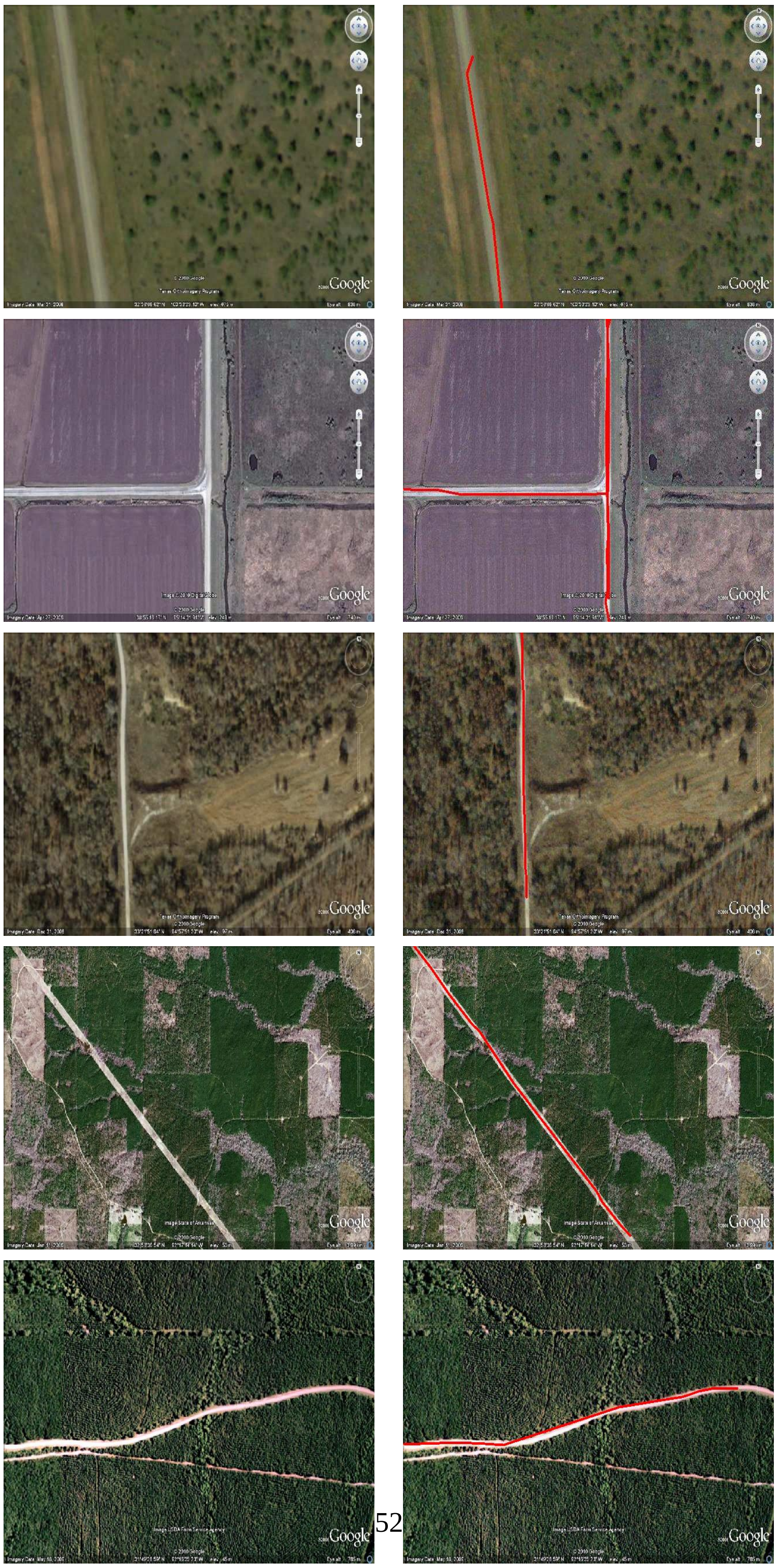

Figure 6.1: Left image: the original image. Right image: the detected roads.

Chapter 7

Conclusion

The algorithm proposed in this work is able to detect the roots present in an image taken by a minirhizotron camera. Previous work done on this topic assumed that the intensity profile of the roots is Gaussian in nature with the brightest pixels at the center of the root. Moving outwards from the center of the root the intensity decreases with a Gaussian profile. This work shows that the above assumption is not valid in all cases. There are roots that have a Non-Gaussian profile. This work takes the non-Gaussian profile of the root into consideration while formulating the energy function.

The difficult part of the proposed method is coming up with the energy function. The energy minimization framework used is a standard one used that has been used for a large number of applications. The energy function needs to be able to capture the exact characteristics of the centerline of the root. This proves to be a challenge as there are roots that are different than the standard roots which are used to determine with the energy function. The energy function has to be robust so that even though the root is very different it should still be able to detect the centerline. An example of this is roots with non-Gaussian intensity profile. If the energy function does not take into account these roots then it fails to detect the centerline of the roots. The non-Gaussian compensation term used in the energy function is used for roots with non-Gaussian profile. It does not play a significant part for roots with a Gaussian profile.

The greedy algorithm used by Zeng et al. is much faster than RJMCMC used in this work. It is possible to speed up the minimization by using methods like Data Driven MCMC. It is also possible to get a rough approximation of the root with the help of the initialization algorithm and then use RJMCMC to make fine adjustments to the network. The RJMCMC algorithm has an advantage over greedy algorithm in spite of its slow speed. RJMCMC is will find the global energy minimum without getting stuck in local minimum with higher probability than greedy algorithm. The results obtained using RJMCMC are better in terms are quality of detected roots as well as the number of roots detected.

Due to the similarity in the characteristics of roads with roots, it is possible to use the same method to detect both of them. Roads and roots do not have the exact same characteristics. In order to detect roads it is necessary to change the energy function by weighting the terms in the energy function. The initialization algorithm used for roots fails for some cases in case of roads due to different background. An initialization algorithm that will work for both can be developed as part of future work.

There are other structures like blood veins, rivers and cracks in cement blocks that can be detected using the proposed method due to their similarity with roots. In each case the energy function will have to be modified to capture the exact characteristics of the structure. The same energy minimization frame work can be used to detect these structures if the energy function is good enough to capture the distinguishing characteristics of the structures.

APPENDIX

Bayes' Rule

Bayes rule or Bayes theorem is an expression of conditional probabilities. Conditional probabilities represent the probability of an event occurring given some evidence. Bayes theorem provides a mathematical rule for changing existing beliefs in light of new evidence. Mathematical Bayes rule is written as

$$posterior = \frac{likelihood * prior}{marginal\ likelihood} \tag{A-1}$$

In terms of mathematical symbols it can be stated as

$$P\left(R = r|e\right) = \frac{P\left(e|R = r\right) P\left(R = r\right)}{P\left(e\right)} \tag{A-2}$$

where $P\left(R = r|e\right)$ denotes the probability that random variable R has value r given evidence e. The denominator is just a normalizing constant that ensures the posterior adds up to 1. It is computed by summing up the numerator over all possible values of R, i.e.

$$\begin{aligned} P\left(e\right) &= P\left(R = 0, e\right) + P\left(R = 1, e\right) + \ldots \\ &= \sum_{r} P\left(e|R = r\right) P\left(R = r\right) \end{aligned} \tag{A-3}$$

This is called the marginal likelihood and gives the prior probability of the evidence. In most cases the denominator term is ignored as it remains constant for all the observations.

Here a simple example of Bayes rule. Suppose a person X has tested positive for a disease, what is the probability of the person actually has the disease given that the test has some false positive rate?

Let $P\,(\text{Test} = \text{+ve} \mid \text{Disease = true}) = 0.95$, so the false negative rate, $P\,(\text{Test} = \text{-ve} \mid \text{Disease = true}) = 5\%$. Let $P\,(\text{Test} = \text{+ve}| \text{ Disease = false}) = 0.05$, so the false positive rate is also 5%. Suppose the disease is rare: $P\,(\text{Disease = true}) = 0.01$. Let D denote Disease (R in the above equation) and ”T = 1” denote the positive Test (e in the above equation). In case of disease D 1 represents that the person has disease and 0 represents that the preson does not have the disease.

$$\begin{aligned} P\,(D = 1|T = 1) &= \frac{P\,(T = 1|D = 1) * P\,(D = 1)}{P\,(T = 1|D = 1) * P\,(D = 1) + P\,(T = 1|D = 0) * P\,(D = 0)} \\ &= \frac{0.95 * 0.01}{0.95 * 0.01 + 0.05 * 0.99} \\ &= \frac{0.0095}{0.0590} \\ &= 0.161 \end{aligned} \tag{A-4}$$

So the probability of person X having the disease given that he tested positive is just 16%.

Bibliography

[1] C. Andrieu, N. de Freitas, and A. Doucet. Reversible Jump MCMC Simulated Annealing for Neural Networks. In *Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 11–18, San Francisco, CA, 2000.

[2] S. Bandyopadhyay. Simulated Annealing Using a Reversible Jump Markov Chain Monte Carlo Algorithm for Fuzzy Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):479–490, 2005.

[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, 1996.

[4] S. Brooks, P. Giudici, and G. Roberts. Efficient Construction of Reversible Jump MCMC Proposal Distributions. *Journal of the Royal Statistical Society*, 65:3 – 39, 2003.

[5] K. Choo and D. J. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. *IEEE International Conference on Computer Vision*, 2:321, 2001.

[6] I. L. Dryden, R. Farnoosh, and C. C. Taylor. Image Segmentation using Voronoi Polygons and MCMC, with application to muscle fiber images. *Journal of Applied Statistics*, 33(6):609, 2006.

[7] P. J. Green. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4):711 – 732, 1995.

[8] Z. Khan, T. Balch, and F. Dellaert. MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(11):1805–1918, 2005.

[9] W. Kim and K. M. Lee. Markov Chain Monte Carlo Combined with Deterministic Methods for Markov Random Field Optimization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1406–1413, 2009.

[10] C. Lacoste, X. Descombes, and J. Zerubia. Road Network Extraction in Remote Sensing. In *IEEE ICIP*, pages 1017–1020, 2003.

[11] K. N. McBride. *Vehicle Tracking in Occlusion and Clutter*. PhD thesis, University of Waterloo, 2007.

[12] D. P. M. Scollnik. An Introduction to Markov Chain Monte Carlo Methods and their Actuarial Applications. *Proceedings of the Casualty Actuarial Society, LXXXIII*, pages 114–165, 1996.

[13] K. Smith. Reversible-Jump Markov Chain Monte Carlo Multi-Object Tracking Tutorial, 2006.

[14] K. Smith, D. Gatica-Perez, and J.-M. Odobez. Using Particles to Track Varying Nnumbers of Interacting People. In *CVPR*, pages 962–969, 2005.

[15] L. Spezia. Reversible Jump and the Label Switching Problem in Hidden Markov Models. *Journal of Statistical Planning and Inference*, 139(7):2305 – 2315, 2009.

[16] M. Stephens. Bayesian Analysis of Mixture Models with an Unknown Number of Components – an Alternative to Reversible Jump Methods. *Annals of Statistics*, 28(1):40 – 74, 1998.

[17] R. Stoica, X. Descomes, and J. Zerubia. A Gibbs Point Process for Road Extraction from Remotely Sensed Images. *International Journal Of Computer Vision*, 57(2):121–126, July 2004.

[18] 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, March 1998.

[19] G. Zeng. *Real-Time Automatic Linear Feature Detection in Images*. PhD thesis, Clemson University, August 2008.

[20] 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.

[21] G. Zeng, S. T. Birchfield, and C. E. Wells. Automatic Discrimination of Fine Roots in Minirhizotron Images. *New Phytologist*, 177(2):549–557, Jan. 2008.