Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

January 2014

Automatic Multi-Model Fitting for Blood Vessel Extraction

Xuefeng Chang
The University of Western Ontario

Supervisor
Olga Veksler
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

© Xuefeng Chang 2014

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

Chang, Xuefeng, "Automatic Multi-Model Fitting for Blood Vessel Extraction" (2014). *Electronic Thesis and Dissertation Repository*. 1869.
https://ir.lib.uwo.ca/etd/1869

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact tadam@uwo.ca.

AUTOMATIC MULTI-MODEL FITTING FOR BLOOD VESSEL EXTRACTION

(Thesis format: Monograph)

by

Xuefeng Chang

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Xuefeng Chang 2014

Abstract

Blood vessel extraction and visualization in 2D images or 3D volumes is an essential clinical task. A blood vessel system is an example of a tubular tree like structure, and fully automated reconstruction of tubular tree like structures remains an open computer vision problem. Most vessel extraction methods are based on the vesselness measure. A vesselness measure, usually based on the eigenvalues of the Hessian matrix, assigns a high value to a voxel that is likely to be a part of a blood vessel. After the vesselness measure is computed, most methods extract vessels based on the shortest paths connecting voxels with a high measure of vesselness. Our approach is quite different. We also start with the vesselness measure, but instead of computing shortest paths, we propose to fit a geometric of vessel system to the vesselness measure. Fitting a geometric model has the advantage that we can choose a model with desired properties and the appropriate goodness-of-fit function to control the fitting results. Changing the model and goodness-of-fit function allows us to change the properties of the reconstructed vessel system structure in a principled way. In contrast, with shortest paths, any undesirable reconstruction properties, such as short-cutting, is addressed by developing ad-hock procedures that are not easy to control.

Since the geometric model has to be fitted to a discrete set of points, we threshold the vesselness measure to extract voxels that are likely to be vessels, and fit our geometric model to these thresholded voxels.

Our geometric model is a piecewise-line segment model. That is we approximate the vessel structure as a collection of 3D straight line segments of various lengths and widths. This can be regarded as the problem of fitting multiple line segments, that is a multi-model fitting problem. We approach the multi-model fitting problem in the global energy optimization framework. That is we formulate a global energy function that reflects the goodness of fit of our piecewise line segment model to the thresholded vesselness voxels and we use the efficient and effective graph cut algorithm to optimize the energy.

Our global energy function consists of the data, smoothness and label cost. The data cost encourages a good geometric fit of each voxel to the line segment it is being assigned to. The smoothness cost encourages nearby line segments to have similar angles, thus encouraging smoother blood vessels. The label cost penalizes overly complex models, that is, it encourages to explain the data with fewer line segment models.

We apply our algorithm to the challenging 3D data that are micro-CT images of a mouse heart and obtain promising results.

Keywords: Blood vessel extraction, energy minimization, graph-cuts, geometric model fitting, piece-wise smooth model, Potts model

Acknowledgements

I want to give my deepest gratitude to Dr. Olga Veksler, my supervisor. I've learned a lot from her serious attitude toward academic research. I enjoy and appreciate the feeling of being treated as a equal friend during discussion. When I was writing this thesis, she gave me a lot of useful instructions and pointed out many mistakes. It is impossible to finish this thesis without her help.

I would like to thank Dr. Yuri Boykov. He gave me precious suggestions about model selection and optimization, which was proved to be very useful in experiment. Besides, I was deeply impressed by his way of problem tracing. He always could find out the potential problems by observing the details carefully. I believe this would help me a lot in my future work.

Special thanks are given to the members of my examining committee, Dr. Charles Ling, Dr. Maria Drangova and Dr. John Barron.

I would extend my sincere thanks to students in Vision Group, Yuchen Zhong worked together with me for almost one year. Also Dr. Lena Gorelick, Dr. Hossam Isack, Liqun Liu and Meng tang, for patiently answering my questions and I learnt quite a lot from the discussions with them.

My thanks would go to my beloved family for their loving considerations and great confidence in me. I especially appreciate the support of my fiancee Zhaoyang Liu. She is always there loving me, helping me, and encouraging me.

Contents

Abstract ii

Acknowledgements iii

List of Figures vii

List of Tables xii

List of Appendices xiii

1 Introduction 1

- 1.1 Overview 1
- 1.2 Motivation and Challenges 3
- 1.3 Our Approach 7
- 1.4 Outline of the Thesis 9

2 Related Work 10

- 2.1 Greedy Strategies and the Vesselness Measures 10
- 2.2 Graph-based Methods 12
  - 2.2.1 Automatic 3D neuron tracing using all-path pruning 12
    - Initial Reconstruction 13
    - Pruning a reconstruction 14
  - 2.2.2 Automated Reconstruction using Path Classifiers and Mixed Integer Programming 16
    - Graph Construction 16
    - Q-MIP Formulation 17
    - Path Classification 17

3 Energy Minimization Framework 20

- 3.1 From Multi Line Segment Fitting to Labeling Problem 21
- 3.2 The s-t min-cut problem 23

3.3 Multi-Label Energy Minimization 24
3.3.1 “α-expansion” 24
3.3.2 “$\alpha\beta$-swap” 25
3.4 Label Cost in Graph Cut 25

4 3D Visualization 28
4.1 3D Development Platform 28
4.2 Maximum Intensity Projection 29
4.3 Isolating 3D Objects of Interest 32
4.4 Window Center Adjustment 34

5 Piecewise Line Segment Model 37
5.1 Overview 38
5.2 Model Sampling 40
5.2.1 Random Sampling Strategy 40
5.2.2 Density Based Random Sampling 40
5.2.3 Random Sampling with Angle Agreement 41
5.2.4 Outlier Model and Re-Sampling 43
5.3 Global Energy Function 44
5.3.1 Data Cost 44
5.3.2 Smoothness Cost 46
5.3.3 Label Cost 49
5.4 Parameter Estimation 50
5.4.1 Exhaustive Grid Search 50
5.4.2 Least Squares Based Search 52
5.4.3 Fast Local Search Approach 53

6 Experimental Results 55
6.1 Data Term Experiments 55
6.2 Label Cost Experiments 57
6.3 Smoothness Term Experiments 59
6.4 Comparison of Different Parameter Re-Estimation Approaches 60
6.5 Estimation of σ 62
6.6 Experiment of the Energy 63
6.7 Remaining Challenges 64

7 Conclusion and Future Work 66

7.1 Summary 66
7.2 Limitations 67
7.3 Future Work 67

Bibliography **69**

Curriculum Vitae **74**

List of Figures

1.1 Three slices of an input medical volume, these slices are cross sections. 2

1.2 Left: fitting result on a less curvier blood vessel. Right: fitting results on a curvier vessel. 3

1.3 The first image is the original intensity image, the second image is the thresholded vesselness measures. 4

1.4 An example of partial voluming effect. 5

1.5 An example of bifurcation, The green circled area are shared by the three vessels, and its split among the three vessels is ambiguous. 6

1.6 Sample results generated by random sampling approach. 7

1.7 Sample results generated by density based random sampling approach. 8

1.8 The flow diagram of our approach. The input data is a small cube (30x20x20) cropped from the original volume. 9

2.1 A snapshot of vesselness orientations, most of which are along the vessels. . . . 11

2.2 Results of different steps in automatic 3D neuron tracing using all-path pruning, and this image is taken from [43]. 13

2.3 (a) The remaining complete reconstruction after DLP. (b) The remaining reconstruction after CLP. (c) The remaining reconstruction after an INP. 15

2.4 Left:Aerial image of a suburban neighborhood, Right:Graph obtained by linking the seed points [43]. 16

2.5 Left:The graph with probabilities assigned to paths using the path classification approach. Blue and transparent denote low probabilities, red and opaque high ones. Note that only the paths lying on roads appear in red. Right: Final reconstruction obtained by solving the Q-MIP problem [43]. 18

3.1 Solving line segment fitting problem in energy minimization framework. 21

3.2 Multi-layer graph for stereo problem, each label represents a disparity, which is also a layer. The data points are graph nodes, the vertical arcs are data cost(matching cost), horizon arcs are smoothness cost. 22

3.3 A simple min-cut problem instance. There are 9 possible s-t cuts: $C = \{sa, sb\}, \{sa, bd\}, \{sa, dt\}, \{ac, sb\}, \{$ Two of the cuts $\{sa, bd\}, \{sa, dt\}$ have minimal cut cost $\omega(C) = 3$, which means the min-cut solution is not unique. 23

3.4 An example of $\alpha-expansion$ with label cost in directed graph, from [11]. Left: s is source node and t is sink node. $X_1, X_2..X_k$ are variables. y is an auxiliary node. Right: several possible cuts on this graph. 26

3.5 An example of $\alpha - expansion$ with label cost in undirected graph,cited from [11]. Left: each variable node X_p is connected with sink t and auxiliary node † at cost h/2. Right: several possible cuts on this graph. 27

4.1 The architecture of OpenSceneGraph. 28

4.2 Six slices of a small cube of the original data, these slices are cross sections. The slice numbers are 3,13,17,23,27 and 31 from left to right and up to down. It is hard to understand the structure of the vessels from the cross sections.. . . . 30

4.3 The 3D visualization results with MIP on the same cube, the structure is quite clear. 31

4.4 Left: the 3D scene contains vesselness points and the line segments fitted to these points. The yellow circled area shows our segment of interest. Right: the detailed result after isolating the segment of interest, now the rotation center is reset to this segment, notice that the rotation center has moved to this segment, which enables us rotate and observe it more clearly. 32

4.5 The figure of the mechanism for object isolation. The truncated pyramid is the perspective projection view volume, objects outside this volume will not be rendered. The red rectangle is a the bounding box surrounding a line segment. We first build $l(P_{near}, P_{far})$, then visit all nodes within this volume to find the intersected objects, which should be the surrounding box of the segment of interest. In OSG the surrounding box shares a same parent node with its surrounded segment, so we can easily get the pointer of the segment. 33

4.6 Suppose the full intensity range of the data is between -4000 and 32767, and most of the voxels have intensity range between 2000 and 6000. When we use linear interpolation, pixels in the range between 2000 and 6000 get mapped to a very narrow range of output intensities. If we use Window Center adjustment (bottom image), pixels in the range from 2000 to 6000 get mapped into the full output range. 34

4.7 A comparison of visualization results with and without Window Center Adjustment. Here we map voxels whose intensity ranges between 2000 and 6000 to the whole output range. After adjustment, the thin vessels are brighter than before. 35

5.1 The flow diagram of our line segment fitting approach. 38

5.2 Random sampling result on a image data of size 102x96x34. There are many segments around thick vessels, but very few segments around thin vessels. . . . 40

5.3 Sample results based on the density of vessel points. This strategy samples less points around thick vessels and more points around thin vessels, making the overall distribution of line segments more equal throughout the dataset. This ensures that there is neither oversampling (leading to longer computational time) no undersampling leading to poor model fit). 41

5.4 Sample result without angle agreement. 42

5.5 Sample result with angle agreement. 42

5.6 Result comparison with and without outlier model. In the first image, a wrong segment L1 fits both inlier and one outlier point marked with ”p” in the image. . In the second image the outlier model was added and L2 only fits the inliers. The data cost between noise P and L2 is very high, if there is no outlier model, L1 is definitely a better option than L2. 43

5.7 Line segment model. Here t is a point on the segment, x is a vessel point. We assume that the probability of x to appear at some distance from t follows gaussian distribution, and t is evenly distribution on the segment. So the data cost is an infinite mixture of Gaussian distributions. 45

5.8 Line Model. 46

5.9 Result generated by line model based approach. 46

5.10 The Potts model, here L_p and L_q are two different labels. In our work, we set a constant penalty T_{potts}. If two points belongs to different labels the penalty is T_{potts}, otherwise the penalty is 0. 47

5.11 A snapshot of the Potts model result. The triangular shaped area is fitted by a horizontal line segment , however, we want a vertical segment as neighboring segments should have similar directions. The energy of the Potts model is not related to the difference/similarity of line segments (labels), so it can not regularize the relative orientations of the nearby segments. 48

5.12 A truncated linear piecewise smoothness cost. 49

5.13 A 2D example of Olsson and Boykov model. Points $\tilde{p}$ and $\tilde{q}$ are noisy measurements of points p and q on the underlying curve. The quotient $\frac{|q-q'|}{|p-q|^2}$ yields half the curvature at p under the assumption that p and q belong to a constant curvature segment. 49

5.14 Comparison results between Potts model and piece-wise smooth model. 50

5.15 2D example of our exhaustive grid search approach. The grid is our search space, the red line segment is the global minima, the green one is our exhaustive search result. There is some different between the two line segments, as we only search points on the grid corners. But the endpoints of the global minimum solution may be located not exactly on the grid corners. Finding a more precise solution would require a sub-pixel search, which is too computational expensive. 51

5.16 A snapshot of fitting results based on the least squares approach. These segments are roughly correct, but their precision is not good enough, as this approach only minimizes part of the energy function in equation 5.3. 52

5.17 The local search approach, the green segment is the least squares solution, the red segment is the globally optimal solution. We locally search the endpoints within the regions surrounded by the yellow boxes. In this example, a globally optimum solution will be found as both the endpoints of the globally optimum solution are inside the yellow boxes. 53

5.18 A snapshot of fitting result based on local search. The green circle shows much improved results compared with least squares solution in Figure 5.16. 54

6.1 Left: fitting result generated by line model. Right:fitting result generated by line segment model. 55

6.2 A snapshot of fitting result based on the line model. As we have to 'guess' the length, the line segments often extend beyond what they should be. 56

6.3 A snapshot of fitting result based on the line segment model. 56

6.4 Comparison of of results with different label cost. There is no smoothness cost, and the data cost is based on the line segment model. T_v is the value of the label cost. But there is no good value that can generates very reasonable results. Without smoothness cost, label cost alone does not do a good job. . . . 57

6.5 Comparison of results without label cost, left ($T_V = 0$) and with the label cost, right ($T_v = 320$). The smoothness cost is set to 10. The result is better with non-zero label cost. 58

6.6 Fitting result without label cost, the smoothness cost is set to 30. 58

6.7 A snapshot of fitting result generated by Potts model, with constant label cost and line segment model data cost. 59
6.8 A snapshot of fitting result generated by piece-wise smooth model. There is no label cost, the data cost is based on the line segment model. 60
6.9 A snapshot of fitting result generated by exhaustive grid search approach. . . . 61
6.10 A snapshot of fitting result generated by local search approach. The input of this approach is the results generated by least squares based approach. 61
6.11 A snapshot of fitting result generated by least squares based search approach. . 62
6.12 The left image is the input vesselness measure, the right image is the corresponding σ estimation. σ is directly proportional but not equal to the radius of the vessel. 63
6.13 The converged energy values of $\alpha - expansion$. 63
6.14 The converged energy values of $\alpha - \beta swap$. 64
6.15 A snapshot of fitting result with rings (yellow circle). 64
6.16 A bad case of fitting around a bifurcation. As the ambiguous area among the three vessels are not divided properly, an extra line segment is used to fit vessel voxels within the yellow circle. 65

List of Tables

6.1 Running times of these two approaches. 62

List of Appendices

Chapter 1

Introduction

1.1 Overview

Blood vessel extraction and visualization is an essential clinical task, it is essential in many areas such as diagnosis of vascular diseases and blood flow simulation. Therefore, the automated tracing of vessel structures in 3D images has received much attention over the years [44]. However, we have to admit that there is still a long way to go, as the medical image data usually contains a lot of noise and the structures of vessels often exhibit complex morphology. In this thesis, we focus on automatic vessel extraction of 3D medical images.

Given an input medical volumetric image, our task is to detect the topology structure of the blood vessels, then reconstruct and visualize those vessels. The medical volumetric images we used are three-dimensional images that contain vessel structures. They are obtained by a CT scanner, which is the widely used technique to create images for clinical purposes or medical science.

Our input data are micro-CT images of a mouse heart. The coronary vessels were perfused with a radio-opaque dye called Microfil MV-122 from a company called Flowtech [1]. It is injected in the vessels and then cures to become a solid. The mouse heart is approximately 4 mm across. Figure 1.1 shows several slices of an input medical volume, of size 585x525x892, and these slices are cross sections. There are 892 slices in this volume but we only show three slices, namely slices numbered 120,280 and 450. The bright regions are vessels, the dark regions are background (muscle). In the beginning of the sequence the vessels are mainly arteries and are pretty thick. Vessels are are getting thinner with the increase of the slice number.

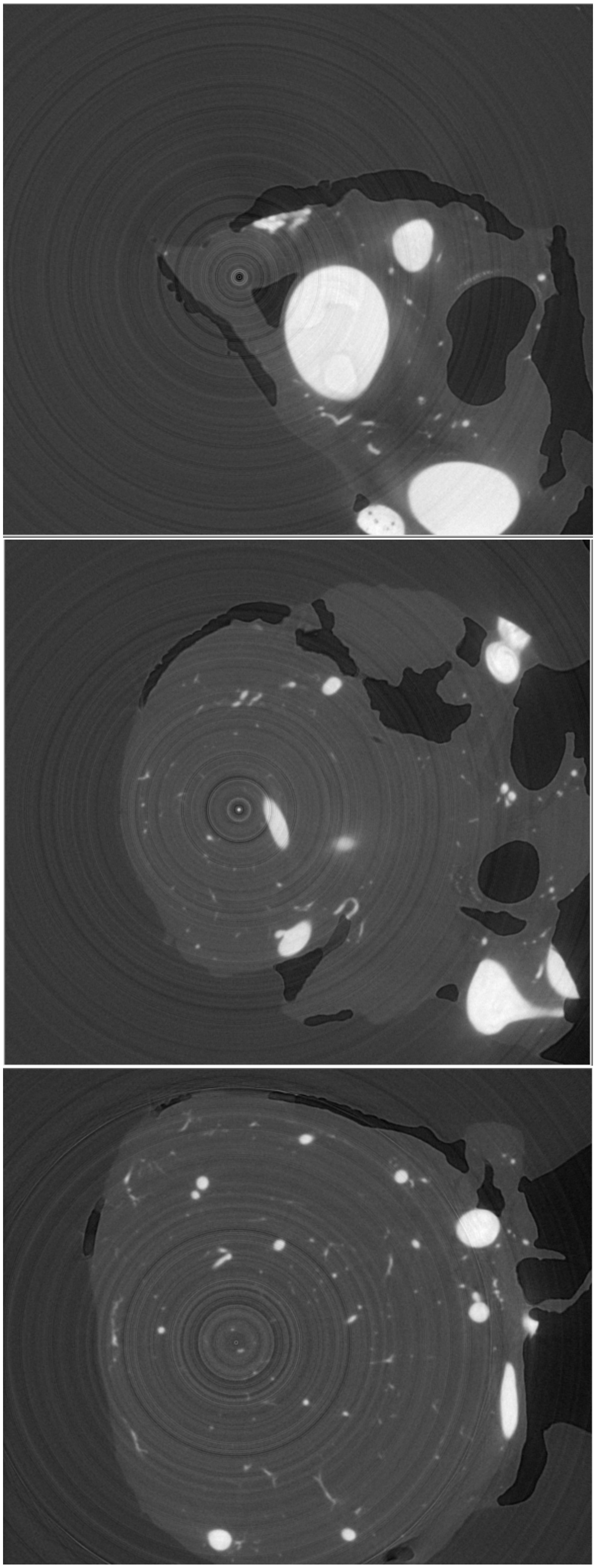

Figure 1.1: Three slices of an input medical volume, these slices are cross sections.

1.2 Motivation and Challenges

A blood vessel system is just one example of a tubular tree like structure. Other examples include neuronal arbors in optical microscopy image stacks, blood vessels in retinal scans, road networks in aerial images, etc. Fully automated reconstruction of tubular tree like structures remains an open computer vision problem [43].

It is very difficult to robustly deal with imaging artifacts such as noise, non-uniform illumination, inhomogeneous contrasts and scene clutter. Practical systems usually require extensive manual intervention. For example, in the DIADEM challenge [2], the algorithms with good performance at tracing dendritic trees usually provide some good tools for manual editing. In medical science research, such editing tasks are very tedious and need prior knowledge of the structure, which dramatically slows down the process. Recently, significant progress has been achieved by formulating the problem as one of optimizing a global objective function [9, 45] without any manually editing. Our work is based on a similar global energy minimization approach, and is directly motivated by the graph cuts algorithm for geometric multi-model fitting problems [10]. In particular, we model a blood vessel as a structure consisting of a sequence of straight line segments. This is a good approximation, provided line segments are of appropriate length. To have a good fit, for a very curved blood vessel, a shorter sequence of line segments is required. For a less curvy blood vessel, a sequence of longer line segments will suffice, see Figure 1.2. We formulate a global energy function that reflects the goodness of fit of the

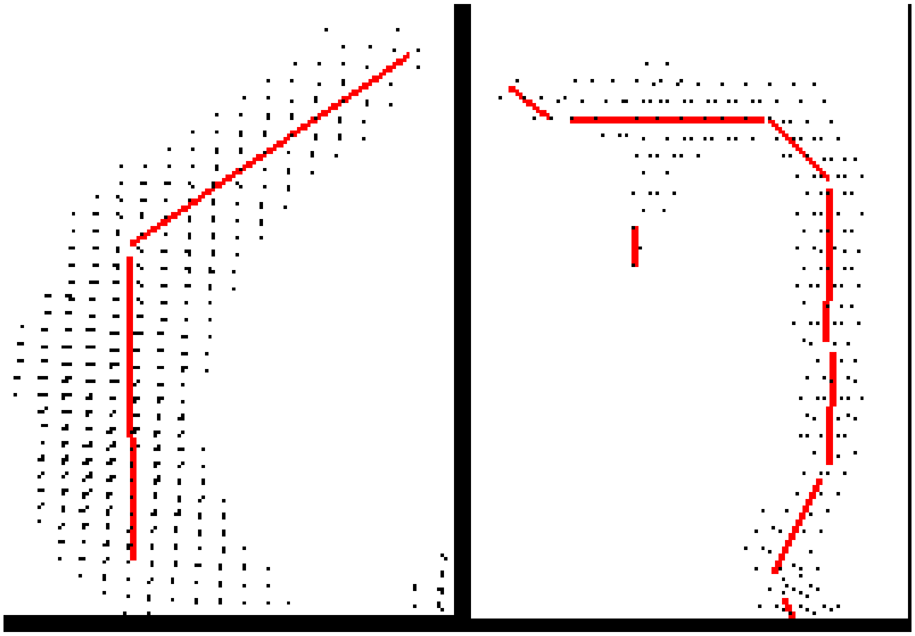

Figure 1.2: Left: fitting result on a less curvier blood vessel. Right: fitting results on a curvier vessel.

piecewise linear segment model and use the graph cut algorithm [7, 10] for optimization.

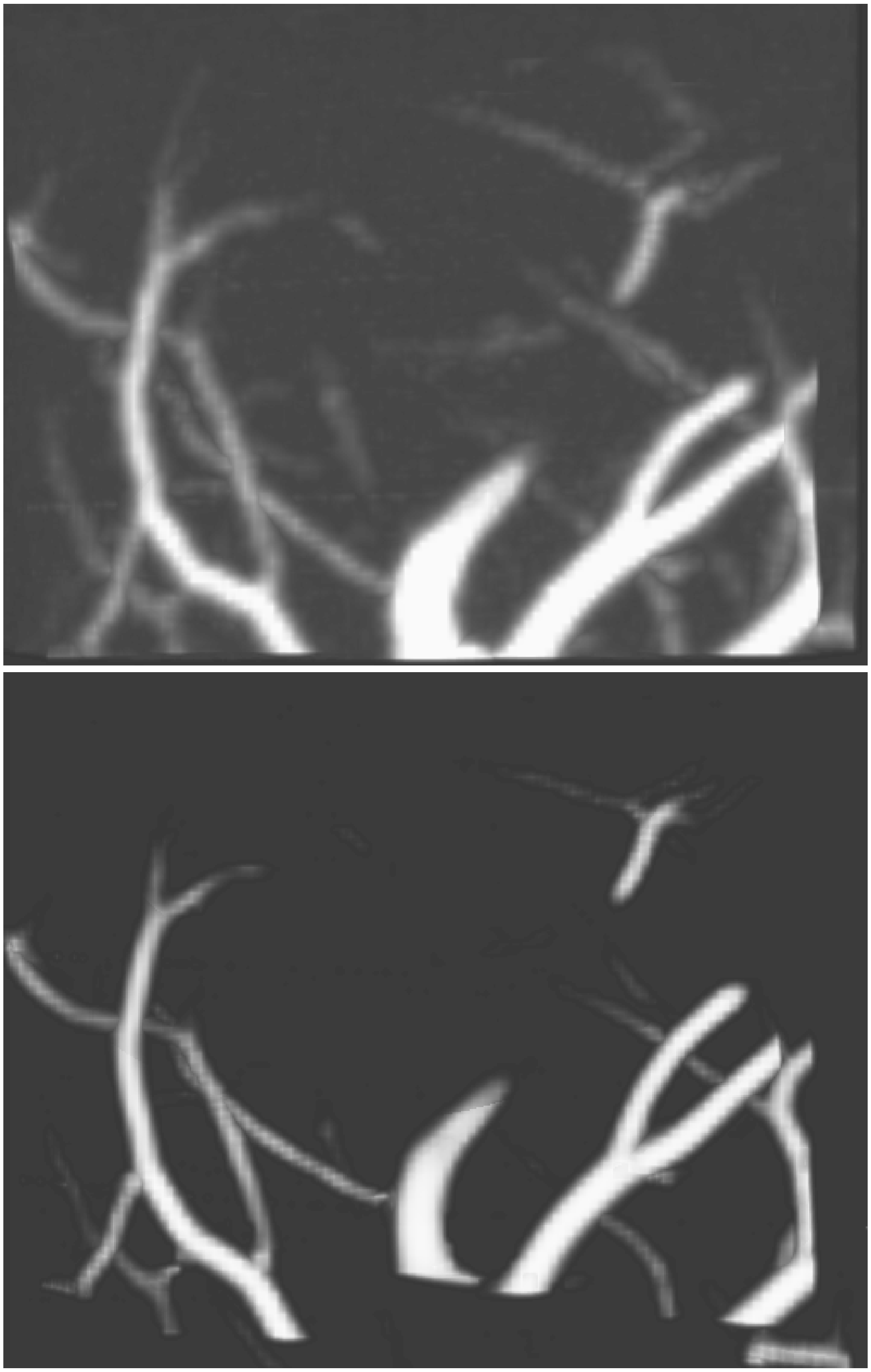

Figure 1.3: The first image is the original intensity image, the second image is the thresholded vesselness measures.

Before we start to fit a piecewise line-segment model to the data, we need to extract the points that are likely to be blood vessels. This can be accomplished by the so-called ”vesselness

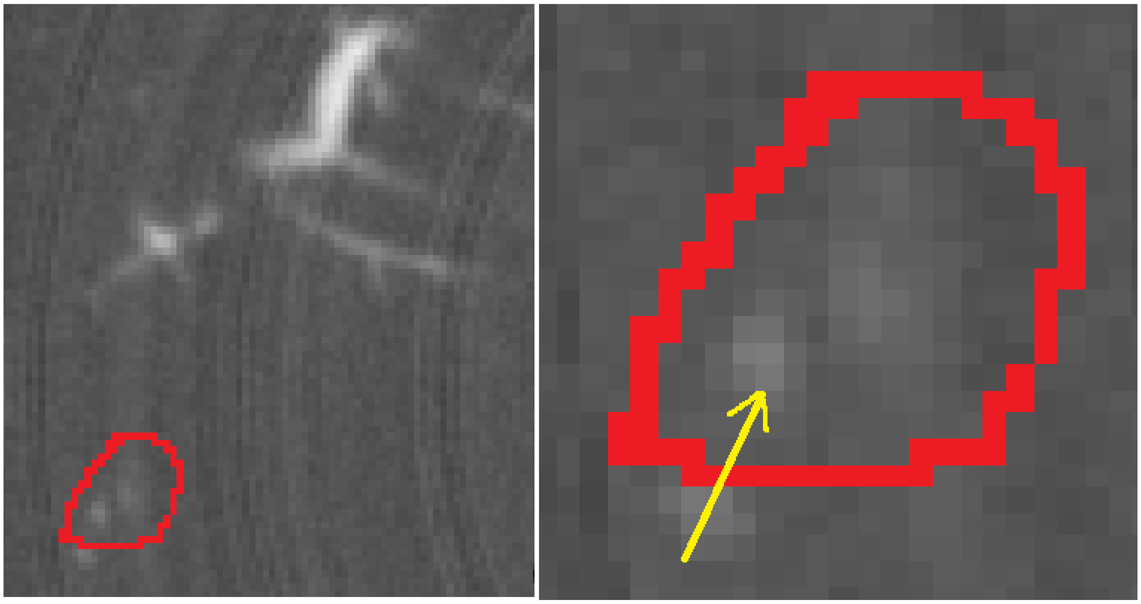

Figure 1.4: An example of partial voluming effect.

measure" [15]. A vesselness measure assigns a high value to a voxel that is likely to be a part of a blood vessel, and a lower value to a voxel that is unlikely to be a blood vessel. Usually vesselness measure is based on the eigenvalues of a Heissian matrix, centered at the given voxel [15]. We threshold the vesselness measures to obtain the voxels that are likely to be part of some vessels, and fit our piecewise line-segment model to the thresholded data. More specifically, our labels correspond to line segments, and the task becomes to fit the thresholded vesselness measure voxels to some line segments.

Figure 1.3 shows an original intensity image and its corresponding thresholded vesselness measures. This data is a small cube (88x67x70) cropped from the original volume (585x525x892). To project 3D data in 2D visualization plane, we use Maximum Intensity Projection (MIP) technology, and we will introduce it in Chapter 4.

The Multi-model fitting approach of [19] has several advantages for our problem. First, the energy minimization framework allows formulation of a global energy function that incorporates any desired fitness criterion for different models, such as distance from point to a line segment, angle difference, etc. Second, by formulating the multi-model fitting problem as a optimal labeling problem, we can use existing optimization technologies, with guaranteed optimality bounds efficiently, such as $\alpha - expansion$ [7] and $\alpha - \beta$ swap [7] in graph-cut.

The challenges of our work mainly come from two aspects. First, there are many small vessels that undergo partial voluming effects. A partial voluming means that part of a vessel occupies only a portion of a voxel, therefore the intensity of the partially occupied voxel is a mixture of the background (muscle) and the vessel tissue. An extreme example is a capillary,

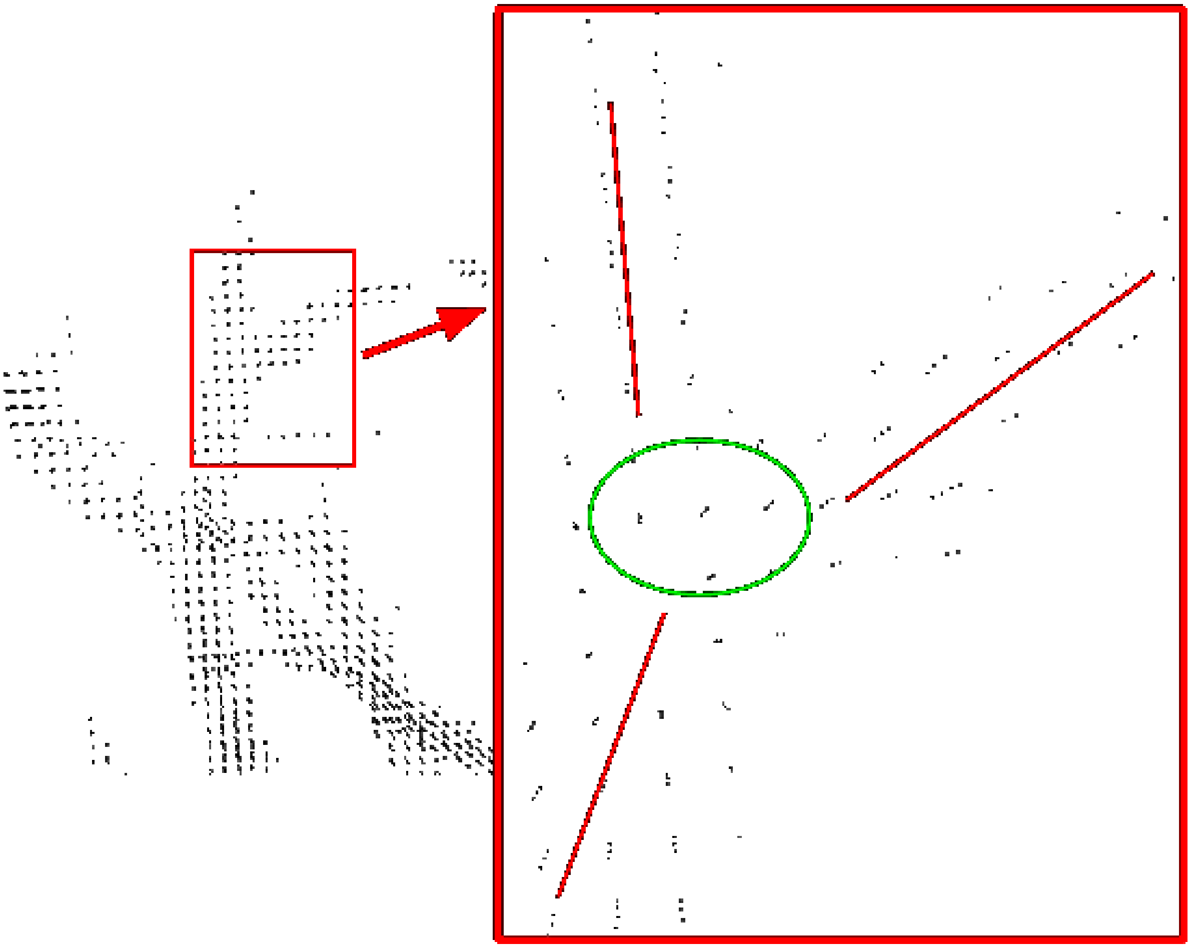

Figure 1.5: An example of bifurcation, The green circled area are shared by the three vessels, and its split among the three vessels is ambiguous.

which is so thin that the cross section occupies only a small portion of a voxel. Figure 1.4 is an example of partial voluming effects. The intensity value of a partial voxel is very low due to intensity mixture, and is difficulty to be distinguished from background. However, most of the vessels are not capillaries, and are easier to segment. We want to be able to detect vessels even if they are so thin as to undergo partial voluming artifacts.

The second challenge of our approach is how to fit well around bifurcations. A bifurcation is a place where two vessels meet and merge into one bigger vessel. It is more challenging to accurately fit line segments at the bifurcation area, as the ambiguous area has to be divided among the three vessels, which is illustrated in Figure 1.5.

1.3 Our Approach

We now outline our approach in more detail. First of all, in this thesis, we assume that we start with the thresholded vesselness measure voxels. This gives us a set of voxels that are very likely to be vessels and we fit the piecewise line segment model to these voxels.

Next we need to fit the piecewise line segment model to the thresholded vesselness data. A possible line segment corresponds to a distinct label in our framework. However, the space of all possible labels is too large to handle directly. The majority of these all possible line segments are a very poor fit for the data at hand. Following [10] , we start by sampling a set of models, i.e. line segments that have a good chance of being a good fit. In our work we use a density based random sampling strategy. The random sampling strategy is to propose line segments on the vessel voxels directly, this strategy provides evenly distributed segments on vessels, and thick vessels usually get more segments than thin vessels, as shown in Figure 1.6.

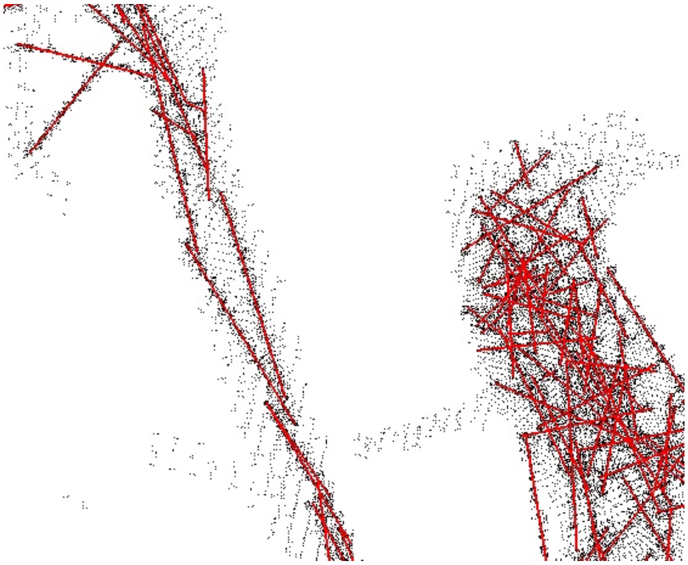

Figure 1.6: Sample results generated by random sampling approach.

However, if most segments are proposed on thick vessels, there will be not enough segments on thin vessels. So we use a density based approach to sample more segments on thin vessels and less on thick vessels, as shown in Figure 1.7. In addition to the ”regular” line segment models, we also need an outlier model to model outlier points, such as noise.

After sampling, we build a global energy function based on line segment model, which consists of data cost, smoothness cost and label cost.

- The data cost measures the probability that a vessel voxel belong to a segment. It gives a high penalty if the voxel is far away form the segment, and vice versa. In our work

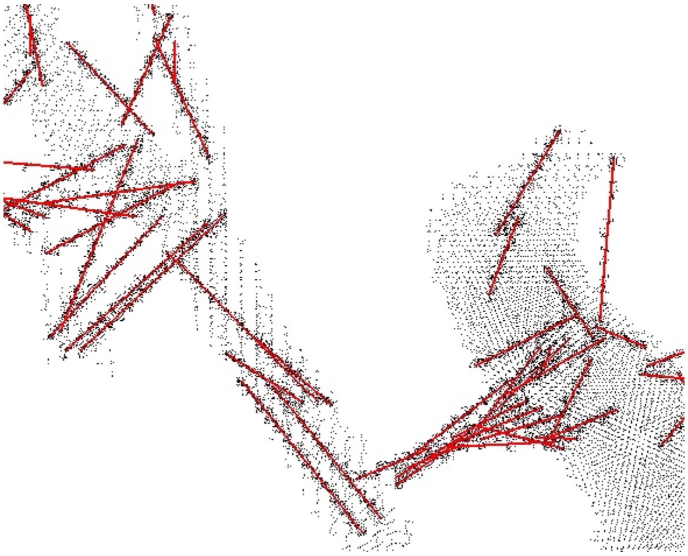

Figure 1.7: Sample results generated by density based random sampling approach.

the data cost is computed as an infinite mixture of isotropic gaussian distribution on line segment.

- The smoothness cost encourages the labeling to be consistent so that fluctuations caused by noisy data cost are smoothed out. We also use the smoothness cost to regularize the relative positions between neighboring segment pairs. Most of the time, two neighboring segments on the same vessel should have similar directions, and one of their endpoints should be close enough.
- The label cost penalizes overly-complex models, that is label cost prefers to explain the data with fewer, cheaper models. We do not want too many line segments fitting one vessel, so we add label cost to the energy function. For each existing model, we set a constant penalty to control the line segment numbers.

We minimize the global energy function via a graph cut algorithm. If the smoothness term is a metric [7] then we use $\alpha - expansion$ to optimize the global function, and if it is a semi-metric [7] we use $\alpha - \beta$ swap. The output of the graph cut is the optimal labeling result, which consists of several clusters. Each cluster consists of all voxels that are assigned to the same label, that is to the same line segment. Since the set of the initial line segments may have been far from optimal, we use these initial label results to find a possibly better set of line segment models. Specifically, we re-estimate the line segments from the initial labeling results by re-fitting the initial labeling with line segments of best fit. In our work, we use a local greedy search based approach to re-estimate the model parameters, this approach gives a locally optimal solution in a relatively short time. Experimental results show that these solutions give a reasonable fit in practice.

After parameter re-estimation, we update the set of labels, that is the set of line segments with the re-fitted models, and remove any line segment models that failed to get any support, i.e. no voxel was assigned to them. Then, we minimize the global energy function again, using the updated set of labels. This process is iterated until the convergence of the energy, which is guaranteed. Figure 1.8 illustrates how our approach works.

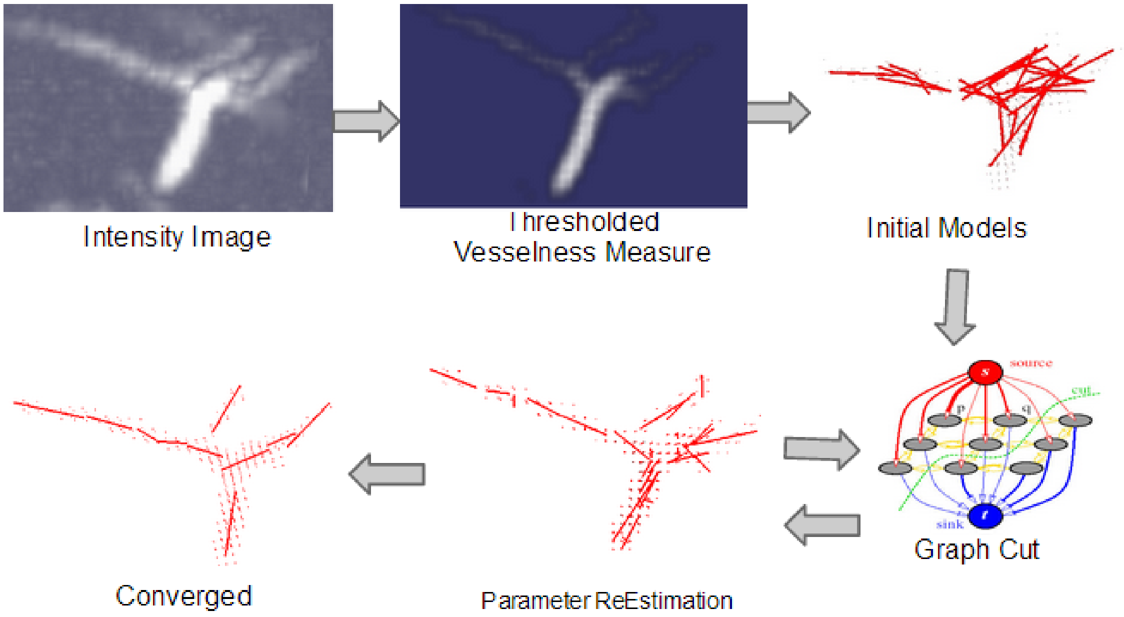

Figure 1.8: The flow diagram of our approach. The input data is a small cube (30x20x20) cropped from the original volume.

1.4 Outline of the Thesis

This thesis is organized as follows. Chapter 2 reviews and analyses the previous works on blood vessel extraction. Chapter 3 introduces optimizing multi-labeling problem with graph cuts in energy minimization framework. Chapter 4 introduces the 3D technology used in our work. Chapter 5 focuses on the construction and optimization of piecewise line segment models. Some experimental results are provided in Chapter 6. The conclusion and future work is in Chapter 7.

Chapter 2

Related Work

Extracting curvilinear structures automatically and robustly is of fundamental relevance to many scientific disciplines, especially in medical research, such as fine modeling of complex blood vessel structures, automated delineation of linear structures in aerial imagery databases. Recently, there has been a resurgence of interest in automated delineation techniques [13, 24, 32, 34], which can be categorized into greedy strategies methods [26, 40] and graph-based methods [14, 37, 43, 45, 50]. In this chapter, we will briefly review several related methods for automatical delineation of linear structures that form complex and potentially loopy networks.

2.1 Greedy Strategies and the Vesselness Measures

Greedy strategies start from a set of seed points, incrementally grow branches by evaluating a local tubularity measure usually based on the Hessian and Oriented Flux matrices [26, 40]. High tubularity paths are then iteratively added to the solution and their end points are treated as the new seeds from which the process can be restarted. Since the search typically involves processing only a part of the image data, these methods are computationally fast. However, they lack robustness to large gaps in the image data as they are sensitive to imaging artifacts and noise.

In our work, we use tubularity vesselness measure as the input data. A vesselness measure is obtained on the basis of all eigenvalues of the Hessian. Let λ_1, λ_2 and λ_3 be the eigenvalue of the Hessian ($|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$). For a point belongs to an ideal tubular structure in a 3D image, λ_1 should be pretty small (ideally zero), and λ_2 and λ_3 should be of a large magnitude and equal sign. The sign is an indicator of brightness.

$$
\begin{aligned}
|\lambda_1| &\approx 0| \\
|\lambda_1| &\ll |\lambda_2| \\
\lambda_2 &\approx \lambda_3
\end{aligned}
\tag{2.1}
$$

The respective eigenvectors u_1,u_2,u_3 point out singular directions: u_1 indicates the direction along the vessel, which is also the minimum intensity variation. We use u_1 as the vesselness orientation. u_2 and u_3 form a base for plane orthogonal to u_1. Figure 2.1 is a snapshot of a vesselness orientation result. In medical images, vessels emerge as bright tubular structures in a darker environment. This prior information can be used as a consistency check to discard structures present in the image with a polarity different than the one sought. The vesselness

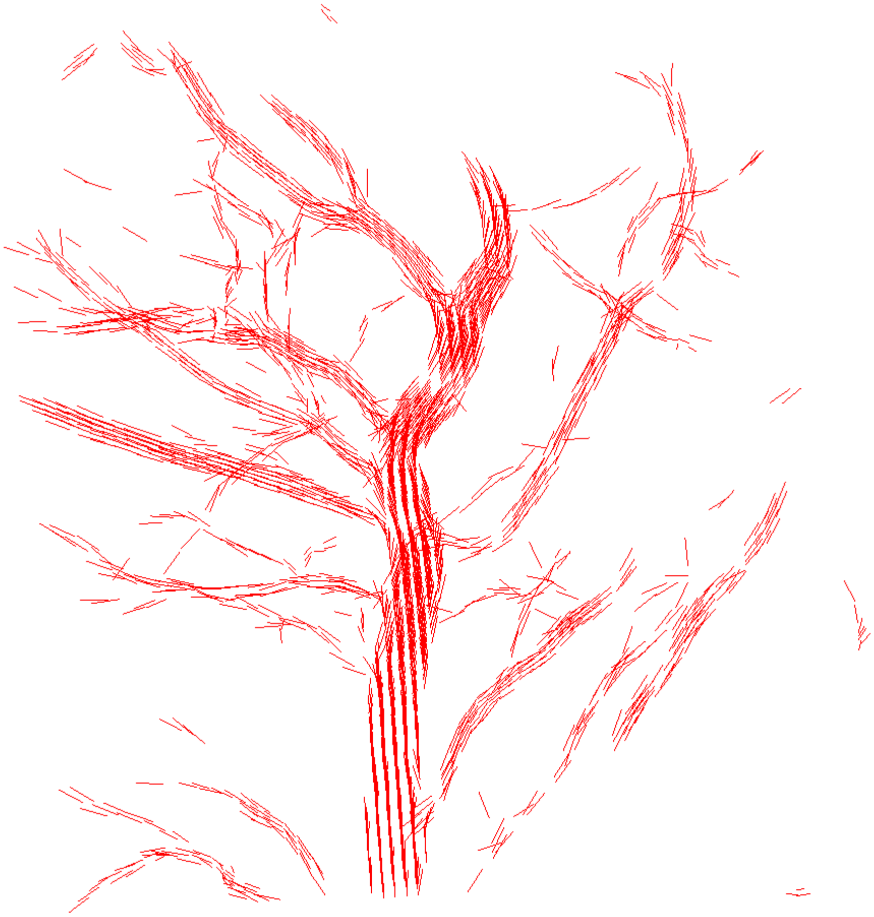

Figure 2.1: A snapshot of vesselness orientations, most of which are along the vessels.

measure function is defined as:

$$f(x) = \begin{cases} 0 & \lambda_2 > 0 \text{ or } \lambda_3 > 0 \\ \left(\left(1 - exp\left(-\frac{R_A^2}{2\alpha^2}\right)\right) exp\left(-\frac{R_B^2}{2\beta^2}\right)\left(1 - exp\left(-\frac{S^2}{2c^2}\right)\right)\right) & \text{otherwise} \end{cases} \tag{2.2}$$

where R_A and R_B are two geometric ratios based on the second order ellipsoid, S is the second order structureness measure. α, β and c are thresholds which control the sensitivity of the line filter to the measures RA, RB and S. $R_A = \frac{|\lambda_2|}{|\lambda_3|}$ refers to the largest area cross section of the ellipsoid and accounts for the aspect ratio of the two largest second order derivatives. It is essential for distinguishing between plate-like and line-like structures, since only in the latter case it is zero. $R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}$ accounts for the deviation from a blob-like structure but cannot distinguish between a line-like and a plate-like pattern; it attains its maximum for a blob-like structure and is zero whenever $\lambda_1 \approx 0$, or λ_1 and λ_2 tend to vanish. $S = \sqrt{\sum_{j \leq 3} \lambda_j^2}$ will be a low value in the background where no structure is present and the eigenvalues are small for the lack of contrast. In regions with high contrast compared to the background, S should be a larger value since at least one of the eigenvalues will be large.

2.2 Graph-based Methods

Graph-based methods find seed points in the whole image or volume by evaluating the tubularity measure [14, 37] densely and finding its local maxima [14, 37, 43, 45, 50]. Although this is more computationally demanding, it can still be done efficiently in Fourier space or using GPUs [12, 26, 27]. The seed points are then connected by paths that follow local maxima of the tubularity measure. This results in a graph that forms an overcomplete representation of the underlying tree structure and the final step is to build a tree by selecting an optimal subset of the edges. This can be done by finding the Shortest Path Tree (SPT) [37], the Minimum Spanning Tree (MST) [14, 53], the k-Minimum Spanning Tree (k-MST) [45], or a solution to the Minimum Arborescence Problem (MAP) [43]. Some graph-based methods are introduced in this section.

2.2.1 Automatic 3D neuron tracing using all-path pruning

The reconstruction of a neuron is defined as a set of topologically connected structure components that describe the 3D spatial morphology of this neuron in a 3D image. This neuron-tracing method consists of two major steps, in the first step an initial over-complete reconstruction (ICR) of a neuron is produced, in the second step the redundant structural components(SC) are pruned. SC is a loosely defined concept that could mean neuron branches, individual reconstruction node,voxels or other sub-structures contained in the neuron reconstruction. Figure 2.2 shows the results of different steps in this paper.

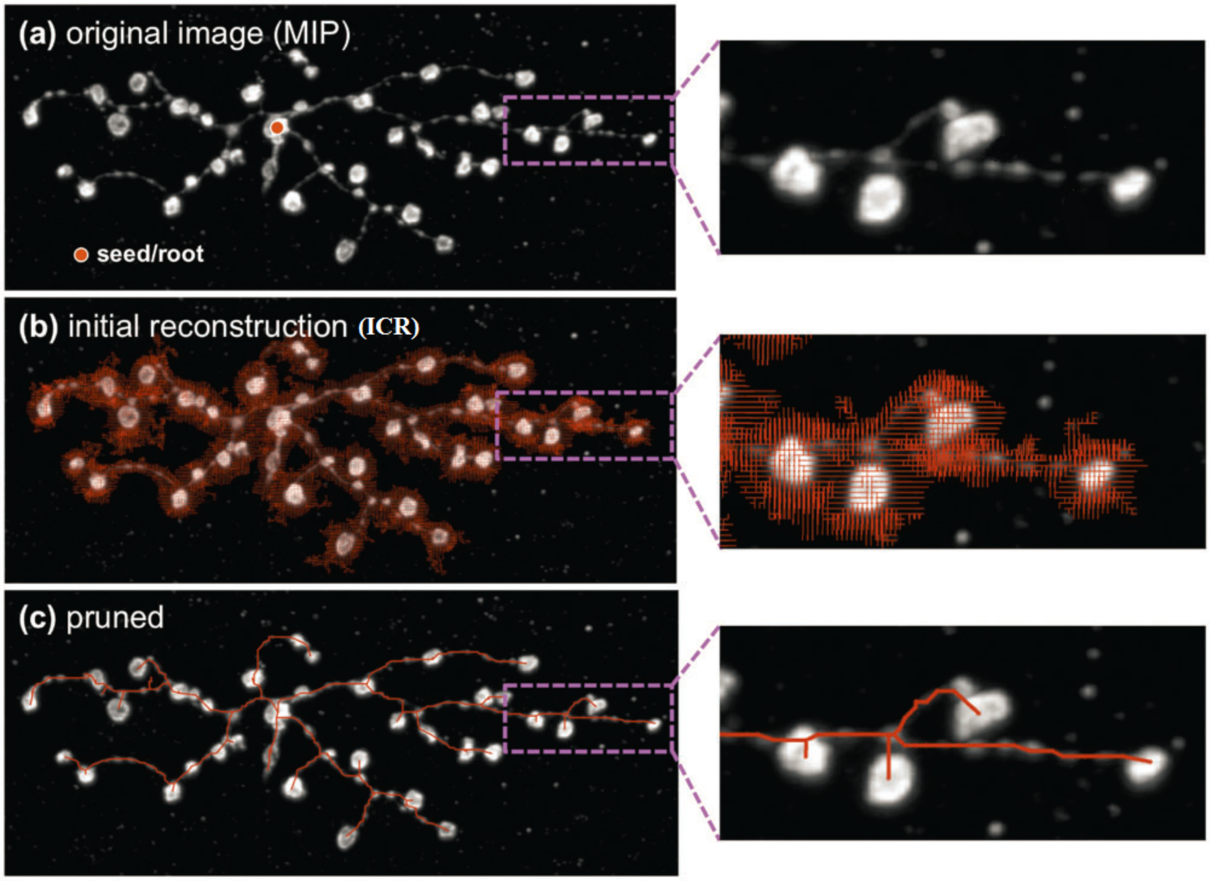

Figure 2.2: Results of different steps in automatic 3D neuron tracing using all-path pruning, and this image is taken from [43].

Initial Reconstruction

The input of this method is a 3D neuron image I and a seed $P_s(x_s, y_s, z_s)$ that is within the neuron region. P_s is often a bright spot of the neuron and can be automatically detected. A global threshold t_a is used to define the image foreground, that is, any voxel that has smaller value than t_a is assumed to be background, otherwise it belongs to foreground. Typically t_a is set to be the average intensity value of the entire image. After the global thresholding, a median filter is applied to remove noise.

Then an undirected graph $G = (V, E)$ is created for the image foreground, where the set of graph vertices V stands for image voxels and the set of undirected graph edges E encodes a geodesic metric function which is define as:

$$
\begin{aligned}
e(v0, v1) &= \|vo - v1\| \left(\frac{g_I(vo) + g_I(v1)}{2} \right) \\
g_I(p) &= exp\left(\lambda_I \left(1 - I(p) / I_max\right)^2 \right),
\end{aligned}
\tag{2.3}
$$

$v0 = (x0, y0, z0)$ and $v1 = (x1, y1, z1)$ are two adjacent foreground voxels, and their spatial coordinates must satisfy $|x0 - x1| \leq 1, |y0 - y1| \leq 1, |z0 - z1| \leq 1$. As G only contains edges of adjacent voxel-vertices, it is highly sparse. $I(p)$ is the intensity value for voxel p, I_{max} is the maximum intensity of the entire image I. λ_I is a positive coefficient.

Then Dijkstra algorithm is applied to G to find the shortest paths from the seed P_s to all other vertexes in G. In the resultant shortest path map, the vertices that have no child are leaves. Obviously, a path can be traced back from every leaf vertex to the seed P_s, which is the root vertex. All these paths share many common sub-paths. The entire solution can be organized as a tree graph G_T. Since it contains all ordered paths from the root to all foreground voxels, the building of this tree graph is called the all-path reconstruction, and it is an ICR.

Pruning a reconstruction

Since the ICR contains all the possible paths and thus could contain redundant SC, the redundant structural elements need to be pruned by a maximal covering minimal-redundant (MCMR) subgraph algorithm. MCMR consists three pruning steps: Dark-leaf pruning (DLP),Covered-leaf pruning (CLP) and Inter-node pruning (INP).

DLP removes dark leaf nodes from G_T. In the beginning, the input image is thresholded by t_a. t_a is a very low value so that all possible paths in the neuron can be captured and any potential connectivity of any possible neuron regions connecting to the seed can be maximized. However, t_a is so low that the foreground often contains many dark voxels. In an ICR, these dark voxels correspond to many redundant branches. The DLP iteratively removes the leaf nodes whose intensity value is lower than t_v. t_v is a new threshold that defines the lowest bright voxel intensity. After DLP, the structure complexity is reduced, while the connectivity of different regions is still maximized. In CLP, a radius-adjustable sphere centered at a reconstruction node is defined, the radius gradually increases until 0.1% of the image voxels within this sphere that are darker than t_a. Each of the reconstruction nodes, along with its estimated radius are treated together as an SC. For two SCs, a and b, a is significantly covered by b if the following condition is satisfied:

$$
\Omega(a \cap b) / \Omega(a) \geq 0.9, \tag{2.4}
$$

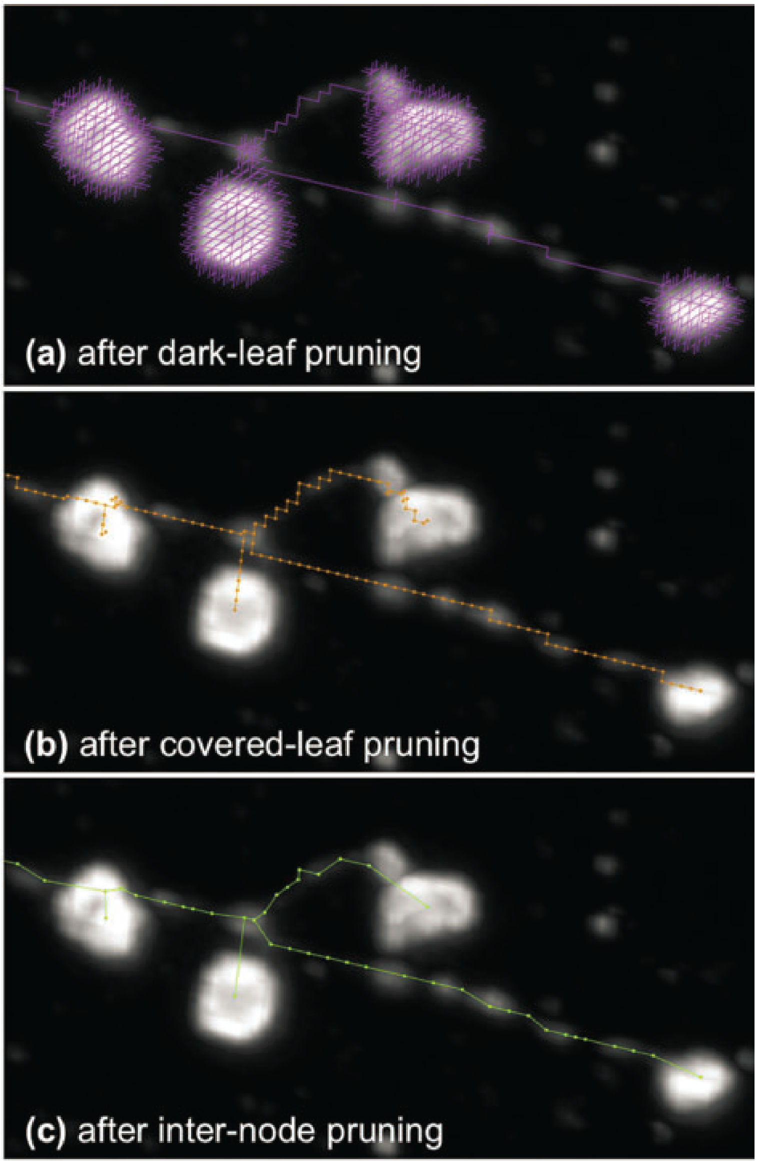
(a) after dark-leaf pruning
(b) after covered-leaf pruning
(c) after inter-node pruning

Figure 2.3: (a) The remaining complete reconstruction after DLP. (b) The remaining reconstruction after CLP. (c) The remaining reconstruction after an INP.

$\Omega(.)$ computes the volume of the occupied region of a reconstruction node. This equation tells us whether node b is significantly covered by node a. When a leaf node is covered by another node or several other nodes jointly,this leaf node should be pruned, otherwise the leaf node should be kept. CLP checks all the leaf nodes and removes those being significantly covered by other nodes. This pruning process is iterated until no more leaf node can be pruned.

After CLP, all neuron regions have been reached by a minimum number of leaf nodes. Then INP is applied to remove the redundant inter-nodes that connect leaf nodes to branch nodes or the root. For each leaf node a, suppose b is the immediate parent node of a. If b is not a branching node or the root and is significantly covered by a, then b is pruned and $a's$ parent is updated as $b's$ original parent. If b is not pruned, then a same check should be executed on bs non-branching-node parent. For each node, this process is iterated until a branching parent node or the root is reached.

2.2.2 Automated Reconstruction using Path Classifiers and Mixed Integer Programming

In this approach, first a directed graph is constructed, which is an over-complete representation for the underlying network of tubular structures. Then a Q-MIP (Quadratic Mixed Integer Programming) problem is formulated to find the most likely arborescence. As well, a path classifier is trained to provide probabilistic weights, which are of great importance to Q-MIP.

Graph Construction

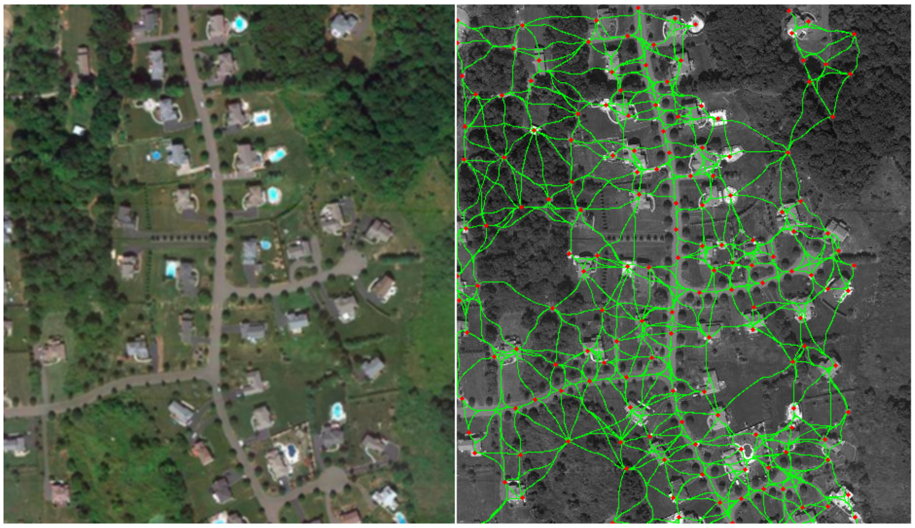

Figure 2.4: Left:Aerial image of a suburban neighborhood, Right:Graph obtained by linking the seed points [43].

There are three steps to build a directed graph G. First, a scale space tubularity measure based on the oriented flux cross-section trace measure [26] is computed. This measure is used to assess if a voxel lies on a centerline of a tubular at a given scale. Second, seed points are sampled from the input image by iteratively selecting the maximum tubularity points and then suppressing their neighboring points. Finally, paths linking the seed points are computed using a 4D minimal path method [29].

Q-MIP Formulation

After constructing a directed graph G, this problem is solved by minimizing the following energy equation

$$\underset{t \in \mathcal{T}(G)}{\arg\min} \sum_{e_{ijk} \in F} c_{ijk} t_{ijk} \tag{2.5}$$

where t is the arborescence in G; $\mathcal{T}(G)$ is the set of all possible arborescence. $F = \{e_{ijk} = (e_{ij}, e_{jk})\}$ is the set of pairs of consecutive edges in G, $e_{ijk} = (e_{ij}, e_{jk})$ represents a pair of consecutive edges e_{ij} and e_{jk}, $e_{ij} = (v_i, v_j)$ is the geodesic tubular path linking seed points v_i, v_j. c_{ijk} encodes the geometric compatibility of consecutive edges e_{ijk}, and t_{ijk} denotes the presence or absence of e_{ijk} in the arborescence. The geometric compatibility c_{ijk} is defined as the probability likelihood ratios assigned to edge-pairs:

$$c_{ijk} = -log\left(\frac{P\left(T_{ijk} = 1 | I_{ijk}, p_{ijk}\right)}{P\left(T_{ijk} = 0 | I_{ijk}, p_{ijk}\right)}\right) \tag{2.6}$$

where I_{ijk} represents image data around the tubular path p_{ijk}, the probability $P(T_{ijk} = 1 | I_{ijk}, p_{ijk})$ denotes the likelihood of the path p_{ijk} belonging to the arborescence, which is computed based on global appearance and geometry of the paths.

By decomposing the indicator variable t_{ijk} as the product of the two variables t_{ij} and t_{jk}, the minimization of the energy function can be formulated as the Q-MIP:

$$\underset{t \in T(G)}{\arg\min} \sum_{e_{ij}, e_{jk} \in E} c_{ijk} t_{ij} t_{jk} \tag{2.7}$$

This Q-MIP problem is NP-Hard, its solution can be found up to an arbitrarily small tolerance from the true optimum using a branch-and-cut strategy [43].

Path Classification

The results of the Q-MIP optimization depends heavily on the probabilistic weights c_{ijk}. A standard approach to computing such weights is to integrate tubularity values along the paths. However, this approach is often unreliable because a few very high values along the path might offset low values, and fail to adequately penalize spurious branches and short-cuts. Furthermore, it is often difficult to balance between allowing paths to deviate from a straight line and preventing them from meandering too much. The path-classification approach computes the probability estimates using a more reliable way. First, a tubular path is broke down into several segments and one feature vector is computed based on gradient histograms for each segment. Given a tubular path s, let $C(s)$ be the centerline and $r(s)$ the corresponding radius

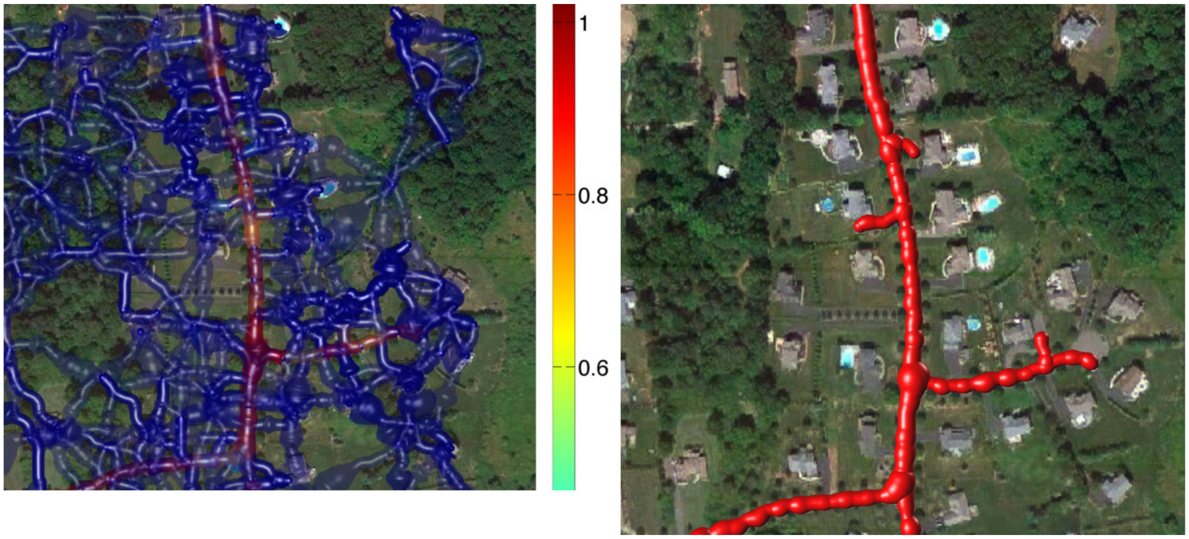

Figure 2.5: Left:The graph with probabilities assigned to paths using the path classification approach. Blue and transparent denote low probabilities, red and opaque high ones. Note that only the paths lying on roads appear in red. Right: Final reconstruction obtained by solving the Q-MIP problem [43].

mappings. The path is divided into equal-length overlapping segments, and for each segment the histogram is computed for points belonging to a certain neighborhood around the centerline with a certain radius.

Then an embedding approach is used to compute fixed-size descriptors from the potentially arbitrary number of feature vectors. To derive from them a fixed-size descriptor, a Bow(Bag-of-Words) approach is used to compactly represent the feature space. The words of the BoW model are generated by randomly sampling a predefined number of descriptors from the training data. For a given path of arbitrary length, the embedding of the HGD descriptors are computed into the codewords of the model. Adapting the sequence embedding approach of [51], the minimum Euclidean distance is calculated from the paths descriptors to each word in the model. This yields a feature vector of minimal distances that has the same length as the number of elements in the BoW model.

Finally, the descriptors are feed to a SVM classifier to compute a probability estimate. To train the SVM classifier, the positive and negative paths are randomly sampled from the ground-truth trees associated to the training images. To obtain negative samples, the tubular graphs are first built in these training images, then paths are randomly selected from these graphs and matching paths are found in the ground truth tree. For a given path, this is done by finding the two nodes of the tree that are closest to the start and end points of the path.

This path-classification approach penalizes paths that mostly follow the true tree structure but cross the background. Thus, it discourages shortcuts, which is something that integrating along the path fails to do.

Chapter 3

Energy Minimization Framework

The energy minimization framework is popular for a variety of computer vision applications. There are mainly two steps in the energy minimization framework. First, build an energy function, then optimize the energy function. Usually it is difficult to design an appropriate energy function, and optimizing the function is also not an easy task. However, it is still a popular framework for several reasons. Firstly, this is a common framework which standard optimization methods can be applied to once the energy function is built. Also one can easily incorporate various prior knowledge into the energy function. Besides, the value of the energy function provides an effective way to evaluate the solution and can be used as a guide in the optimization algorithm [31].

Many of the important developments in computer vision began with a proposal for a better energy, a better algorithm for an energy, or a combination of both [11]. Researchers try to use algorithms that are both effective in theory and fast in practice to optimize new energies. Graph cut is an optimization algorithm in the energy minimization framework that has been widely used in a variety of vision problems [46], including stereo and motion [3, 6, 7, 30, 38, 39, 20, 22], image segmentation [5], image restoration [6, 7, 18, 21], image synthesis [25], multi-camera scene reconstruction [41], and medical imaging [4, 8, 23]. The output result generated by a graph cut algorithm is often a global minima, while in some case [6, 18, 21, 30], it is local minima, but still within a known factor of the global minima [7].

Our algorithm implements multi line segment fitting based on a graph-cut algorithm, which works in the energy minimization framework. The task of line segment fitting is posed as a multi labeling problem, which can be solved by minimizing an energy function. The max-flow/min-cut algorithm can globally optimize the energy function. The way to solve multi-model fitting problem in the framework of energy minimization is shown in Figure 3.1.

In this chapter we will first introduce the way to solve line segment fitting problem in the framework of energy minimization in Section 3.1, then review several well-known concepts on which subsequent chapters are based. Section 3.2 explains the s-t min-cut problem. Section 3.3 explains the move-making algorithms "α-expansion" and "$\alpha\beta$-swap" for optimizing the multi-label energies [7]. We use these move-making algorithms to optimize our energy function.

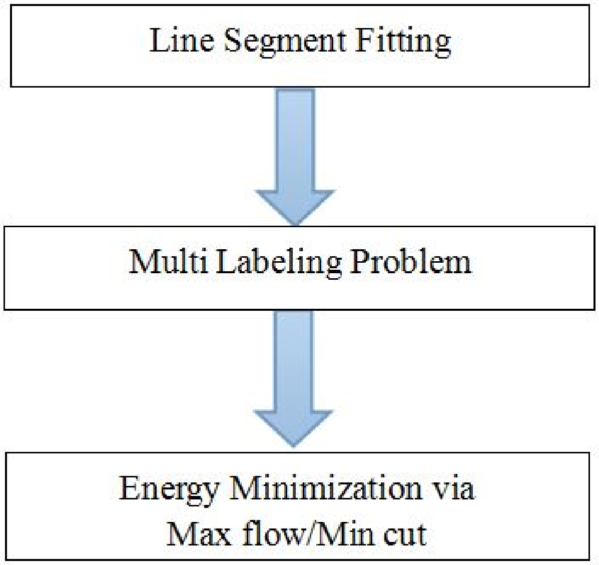

Figure 3.1: Solving line segment fitting problem in energy minimization framework.

3.1 From Multi Line Segment Fitting to Labeling Problem

Traditionally, to get trajectory of a 3D vessel inside 3D medical images, the clinician must manually define some points on the path using 3D orthogonal views. But for a complex structure this path construction task becomes very tedious and one can easily made mistakes. The goal of our work is to get the topology of the 3D vessel structure in a automatic way.

Our approach uses multiple line segments to fit blood vessels, which is a geometric multi-model fitting problem. Geometric multi-model fitting is a typical chicken-egg problem: data points should be clustered based on geometric proximity to models whose unknown parameters must be estimated at the same time [19]. Currently, most existing methods such as RANSAC [42, 47, 54], ignore the overall classification of all points, and just greedily search for models with most inliers that are within a threshold. In this paper we formulate the multi-model fitting problem as an labeling problem, each geometric model is treated as a label, and a global energy function is built to balance the regularity of inlier clusters and geometric errors.

A labeling problem is the task of assigning an explanatory label to each element in a set of observations [11]. Many classical clustering problems are also labeling problems because each label represent a cluster of points, and each point could be assigned to a label. Clearly, we can treat the task of multi-model fitting problem as a labeling problem.

To describe a labeling problem, one needs a set of data points and a set of labels. A discrete labeling problem associates one discrete variable with each data point, and the goal of optimization is to find the best labeling to these variables, which must satisfy some global energy constraints. In computer vision, the data points can be things like pixels in 2D images or voxels in 3D image, disparity measurement in stereo matching, and intensity measurements from CT/MRI. The labels are typically either semantic (car, pedestrian, street) or related to the scene geometry (depth, orientation, shape, texture) [10]. Figure 3.2 shows an example of transforming stereo matching to a multi-label problem.

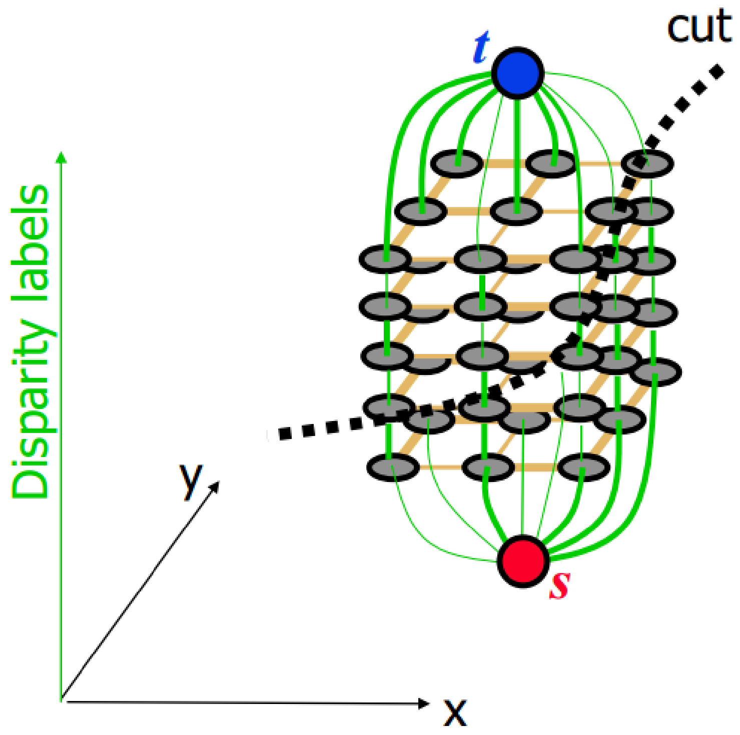

Figure 3.2: Multi-layer graph for stereo problem, each label represents a disparity, which is also a layer. The data points are graph nodes, the vertical arcs are data cost(matching cost), horizon arcs are smoothness cost.

Assume set P contains all data points, ζ contains all labels. We associate a discrete variable with each $p \in P$, and each variable p is allowed to take one label from set ζ. The goal of discrete labeling is to complete a map f:P $\mapsto \zeta$ which assigns a label f_p to each element $p \in$P. In geometric model fitting, the map f assigns labels f_p to 3d image voxels. In this thesis, we are focusing on optimizing the energy function with multiple labels.

3.2 The s-t min-cut problem

To define an $s-t$ min-cut problem, we need a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, and two designated terminals $s, t \in \mathcal{V}$, s is source node and t is sink node, an arc cost $\omega(u, v) \geq 0$ for each $(u, v) \in \mathcal{A}$. The $s-t$ min-cut $C = (\mathcal{S}, \mathcal{T})$ is a partition of $\mathcal{V}$ into two disjoint sets S and T such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$. The cut-set of C is the set $\{(u, v) \in \mathcal{A} | u \in \mathcal{S}, v \in \mathcal{T}\}$. The goal of s-t min-cut is to find and remove the cheapest cut-set C so that there is no path from s to t. The cost of this cut $\omega(C)$ is the total cost of all arcs in C.

Figure 3.3 shows an instance of the min-cut problem. The capacity of an edge is a mapping $c : A \to R_+$, denoted by c_{uv}. It represents the maximum amount of flow that can pass through this edge. A flow of an edge is a mapping $f : A \to R_+$, denote by f_{uv}. There are two constrains with a flow: the flow f_{uv} can not exceed its capacity c_{uv}, and the sum of the flows leaving a node must equal with the sum of flows entering this node, except for the sink node and source node. The value of flow is defined by $|f| = \sum_{u \in v} f_{sv}$, where s is the source node. It represents the amount of flow passing from the source to the sink. The maximum flow problem is to maximize $|f|$, and the maximum value of an s-t flow is equal to the minimum capacity over all $s-t$ cuts [36]. In optimization theory, max-flow/min-cut theorem is stated as follows:

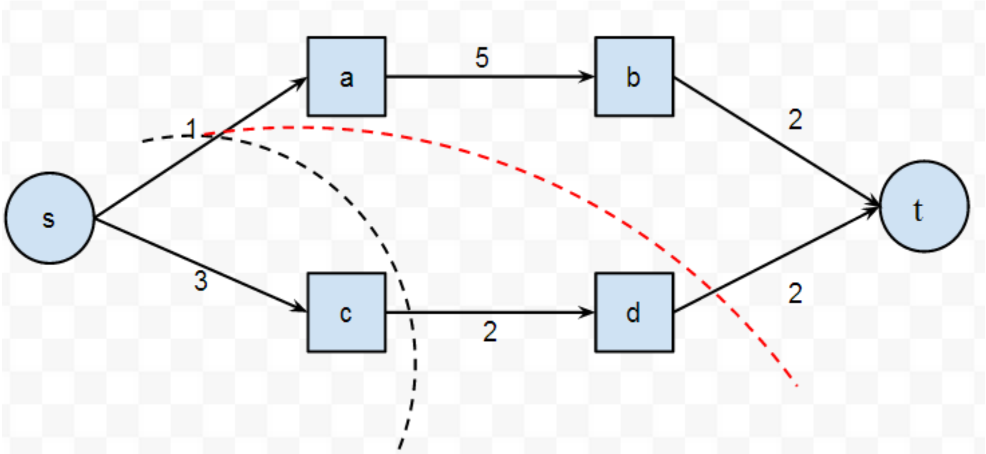

Figure 3.3: A simple min-cut problem instance. There are 9 possible s-t cuts: $C = \{sa, sb\}, \{sa, bd\}, \{sa, dt\}, \{ac, sb\}, \{ac, bd\}, \{ac, dt\}, \{ct, sb\}, \{ct, bd\}, \{ct, dt\}$. Two of the cuts $\{sa, bd\}, \{sa, dt\}$ have minimal cut cost $\omega(C) = 3$, which means the min-cut solution is not unique.

Theorem 3.2.1 *A. f is a maximum flow.*

B. there exists an s-t cut which capacity is equal to the value of f.

C. The residual graph after cut contains no directed path from s to t.

Max-flow/min-cut theorem implies that in a flow network, the min-cut can be computed in a low order polynomial time by a number of classical $s - t$ maximum flow algorithms [31]. Cormen et al. in [28] describe an augmenting path strategy to compute the minimum cut of a graph. Goldberg and Tarjan [17] proposed an push-relabel approach to solve the minimum cut problem.

3.3 Multi-Label Energy Minimization

The multi-label energy implies that the label set has cardinality $|\mathcal{L}| > 2$. The $s-t$ cut reduction is inherently fit for two label problems because there are two terminals s and t. There are special local search algorithms for direct energy minimization, also called move-making algorithms in the computer vision literature [11]. We will explain two move making algorithms in that use graph cuts: the "α-expansion" algorithm and "$\alpha\beta$-swap" algorithm. The two algorithms find approximate solutions by decreasing the energy on appropriate graphs iteratively, which leads to significantly better solutions than the previous algorithms based on 'standard' moves. In our experiment we use both "α-expansion" algorithm and "$\alpha\beta$-swap" algorithm.

3.3.1 "α-expansion"

The α-expansion algorithm performs local search on multi-label energies. Given current labeling f, the idea of one α-expansion move is as follows: all variables should either switch to a particular label α or keep its current label. For a particular label α there are an exponential number of possible moves. Intuitively, α- expansion means the label α can expand its region. The α-expansion algorithm is implemented as shown as below.

```
1. start with an arbitrary labeling $f$
2. while true
3.   for each label $\alpha \in \mathcal{L}$
4.       Find $f_t$=argminE($f$) among $f_t$ within one "$\alpha$-expansion" move of f
5.       if E($f_t$) <E($f$)
6.           $f$=$f_t$
7. if converged
8. return f
```

The α-expansion algorithm could only be used to approximately minimize the energy $E(f)$

with metric interaction penalty V. V is called a metric on the space of labels $\mathcal{L}$ if it satisfies

$$\begin{aligned} V(\alpha,\beta) &= 0 \Leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) &\leq V(\alpha,\gamma) + V(\gamma,\beta) \end{aligned} \tag{3.1}$$

where labels $\alpha,\beta,\gamma \in \mathcal{L}$.

3.3.2 “$\alpha\beta$-swap”

The $\alpha\beta$-swap algorithm performs local search using a different kind of moves. Given a current labeling f, the idea of one $\alpha\beta$-swap move is as follows: all variables with $f_p \in \{\alpha,\beta\}$ choose a new label in α,β. That is each variable with label $f_p \in \{\alpha,\beta\}$ could either swap to the other label or just keep its current label. The $\alpha\beta$-swap algorithm is implemented as shown as below.

```
start with an arbitrary labeling $f$
while true
  for each pair of labels $\alpha,\beta \in \mathcal{L}$
      Find $f_t$=argminE($f$) among $f_t$ within one ”$\alpha\beta$-swap” move of f
      if E($f_t$) <E($f$)
          $f$=$f_t$
if converged
return f
```

The $\alpha\beta$-swap algorithm could be used to approximately minimize the energy $E(f)$ with semimetric interaction penalty V. V is called a semimetric on the space of labels $\mathcal{L}$ if it satisfies

$$\begin{aligned} V(\alpha,\beta) &= 0 \Leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \end{aligned} \tag{3.2}$$

3.4 Label Cost in Graph Cut

In a basic energy $E(f) = \sum_p D_p(f_p)$, the optimal f_p can be computed by minimizing data cost independently. However, we might want to use as few unique labels as necessary. For example, in model fitting we do not want to use too many models, we prefer to fit the data points with fewer models. Then we can introduce a new energy term, which balances the individual preference (the data cost) against the global preference which tends to have fewer unique models. This new energy term is called as label cost and it penalizes each existing unique label in f.

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{l \in \mathcal{L}} H(l) \cdot \delta_l(f) \tag{3.3}$$

$$E_{labelcost} = \sum_{l \in \mathcal{L}} H(l) \cdot \delta_l(f) \tag{3.4}$$

where $H(l)$ is the non-negative label cost of label l, $\delta_l(f)$ is the indicator function

$$\delta_l(f) = \begin{cases} 1 & \exists_p : f_p = l \\ 0 & otherwise \end{cases} \tag{3.5}$$

The label cost can be viewed as a special case of global interactions recently studied in computer vision by Werner [16] and Woodford [19]. Werner proposed a cutting plane algorithm to make certain high-order potentials tractable in an LP relaxation framework [16]. However, this algorithm is very slow. In [10] they proposed a fast algorithm to minimize energy with label cost, by extending the $\alpha - expansion$ algorithm to incorporated label cost at each expansion. Figure 3.4 and 3.5 shows $\alpha - expansion$ with label cost in both directed and undirected graph for label β.

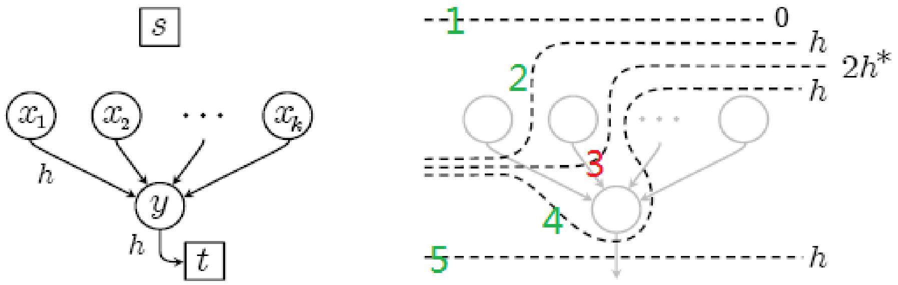

Figure 3.4: An example of $\alpha - expansion$ with label cost in directed graph, from [11]. Left: s is source node and t is sink node. $X_1, X_2..X_k$ are variables. y is an auxiliary node. Right: several possible cuts on this graph.

In Figure 3.4, an auxiliary node y is introduced to construct a subgraph. The cost between each variable node X_p to auxiliary node y is h. y is connected with sink node t. The cost between y and node t is h, which means the max-flow of this subgraph could not exceed h. So in a minimal s-t cut, the subgraph contributes cost is 0 if no X_p is assigned to label β(cut 1), as there is no cut on this subgraph. Otherwise it is h (cut 2,4,5), in such case there is at least one X_p is assigned to label β. Cut 3 could not be a min-cut as its cost is greater than h.

In Figure 3.5, if all variable nodes are cut to sink t and none are assigned to label β, then

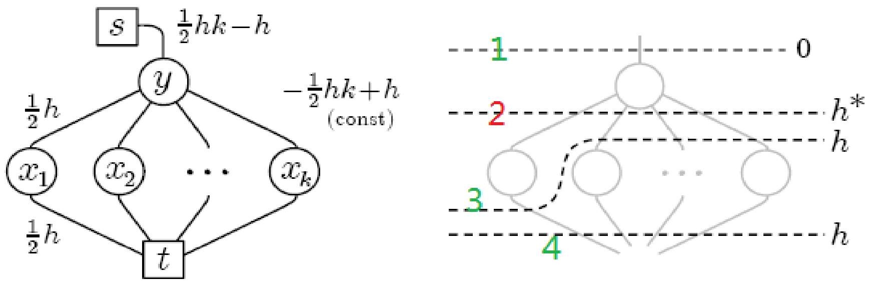

Figure 3.5: An example of $\alpha -$ *expansion* with label cost in undirected graph,cited from [11]. Left: each variable node X_p is connected with sink t and auxiliary node † at cost h/2. Right: several possible cuts on this graph.

the min-cut value of this subgraph is 0 (cut 1). If at least one node is assigned label β, then the the subgraph contributes cost h (cut 3,cut 4). Cut 2 could not be a min-cut with respect to auxiliary variable y.

Chapter 4

3D Visualization

The 3D visualization plays a very important role in our work, that is visualization of the input 3D volumes and output results. As well, we also need to develop some assistant tools to help us to evaluate and debug the results of model fitting in a convenient setting. In this chapter, we introduce our development platform and relevant technologies that are used in our work.

4.1 3D Development Platform

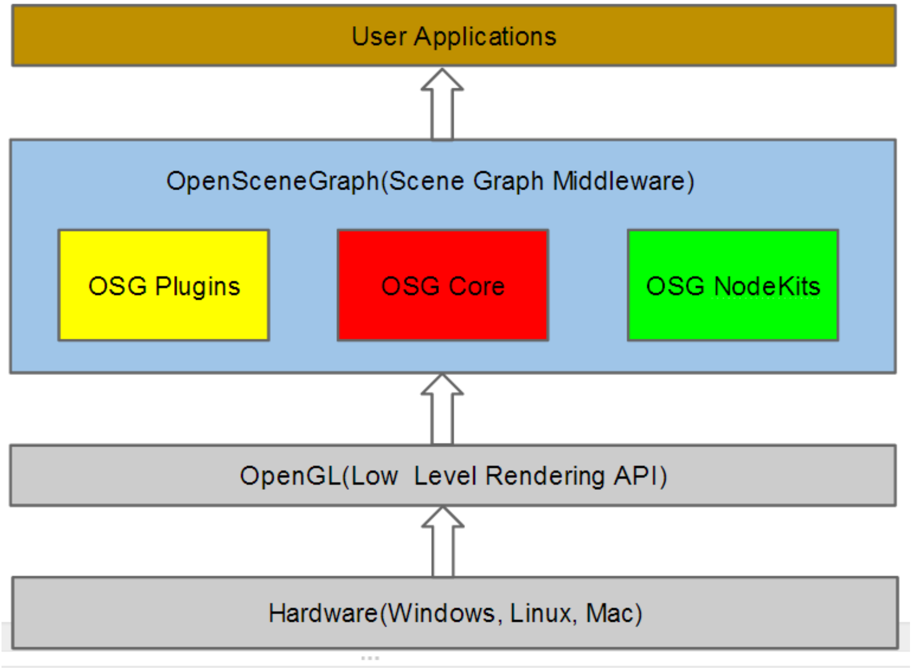
User Applications
OpenSceneGraph(Scene Graph Middleware)
OSG Plugins
OSG Core
OSG NodeKits
OpenGL(Low Level Rendering API)
Hardware(Windows, Linux, Mac)

Figure 4.1: The architecture of OpenSceneGraph.

We develop our 3D programs based on OSG (OpenSceneGraph) [33], which is an open source 3D graphics application programming interface. It is entirely based on OpenGL [52], which ensures it to be a multi-platform application programming interface for drawing 3D and 2D graphics. But it goes beyond OpenGL by providing common function to many 3D applications, such as texture mapping, level of detail control, 3D file and image loaders. Because of its rich features and open source license, OSG has been used by application developers in fields such as visual simulation, virtual reality, scientific visualization and computer games. Besides, OSG is written in Standard C++ language, using the standard template library (STL) for containers, which makes it quite compatible with our main program.

OSG is a well-designed rendering middleware application, which is actually a deferred rendering system based on the theory of a scene graph. A deferred rendering system records rendering commands and rendering data in a buffer so that the commands can be executed at a later time. This deferred rendering design allows the system to perform various optimizations before rendering, as well as implement a multi-threaded strategy for handling complex scenes. A scene graph is a general data structure which represents the 3D worlds as a graph of nodes that contain logical and spatial relationship information.

Typically a scene graph is represented as a hierarchical graph, which contains a set of nodes. Usually there is a top root node, several group nodes which can contain any number of child nodes. There are also a number of leaf nodes, which serves as the bottom layer of the graph. A group node can have an unlimited number of child nodes, and child nodes that belong to the same group node can be treated as one unit and share the information of the parent node. An operation performed on a parent node will be propagated to all the child nodes by default. A typical scene graph tree does not allow isolated elements and directed cycles. Sometimes a child node may have more than one parent node, which means this node is considered to be ”instanced”, and the scene graph can be defined as a directed acyclic graph (DAG). Instancing produces many interesting effects, including data sharing and multi-pass rendering [49].

4.2 Maximum Intensity Projection

The maximum intensity projection (MIP) is a volume rendering method for 3D data that projects in the visualization plane the voxels with maximum intensity that fall in the way of parallel rays traced from the viewpoint to the plane of projection [48]. If using orthographic projection, then two MIP renderings from opposite viewpoints will be symmetrical images. In OSG it is quite convenient to use MIP, first we need to build a 3D texture for the input volume and sent it to the scene graph, then we set the rendering option to be MIP_MODE and render a new frame. OSG will automatically transform the 3D volume into 2D result by comparing and

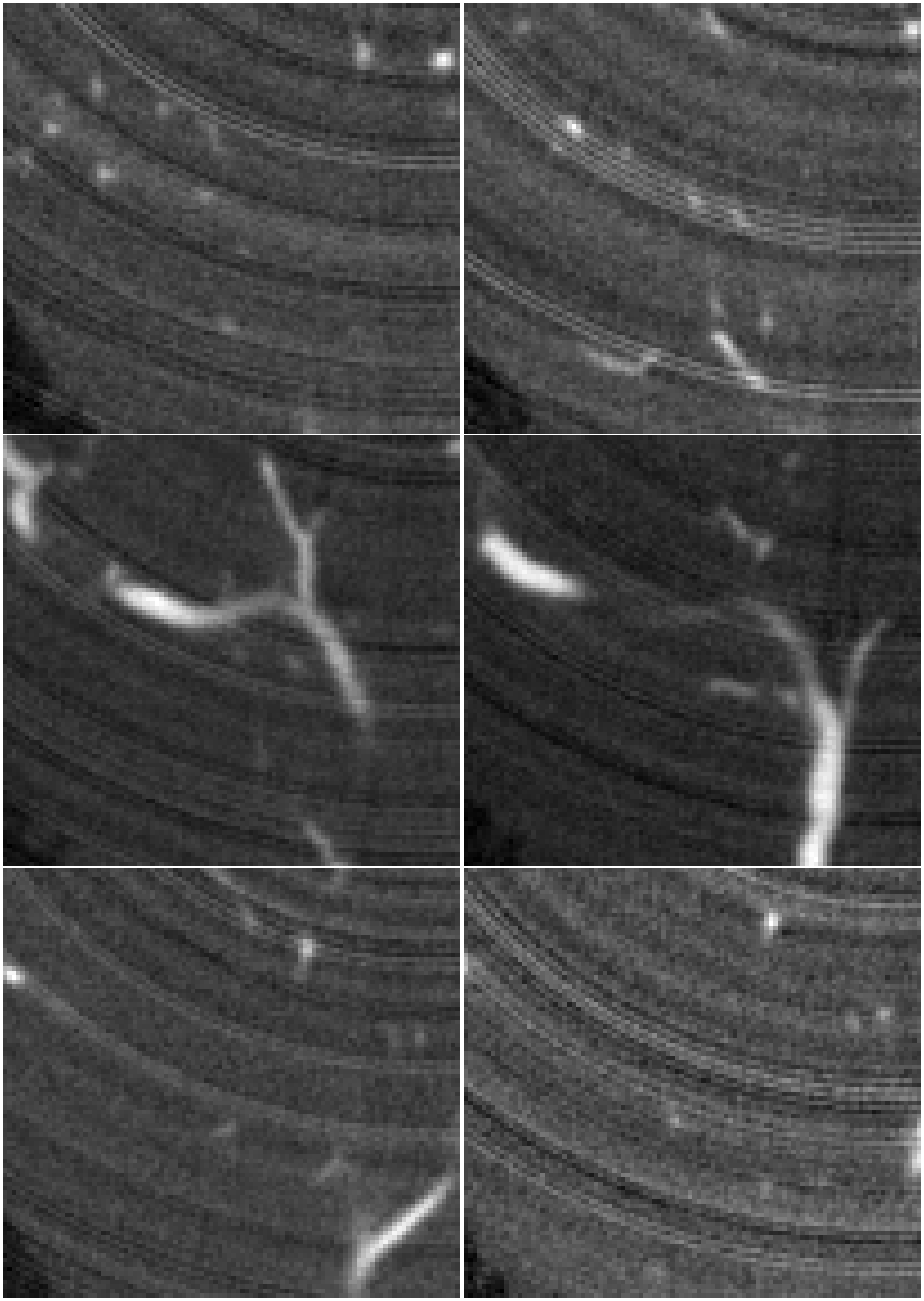

Figure 4.2: Six slices of a small cube of the original data, these slices are cross sections. The slice numbers are 3,13,17,23,27 and 31 from left to right and up to down. It is hard to understand the structure of the vessels from the cross sections..

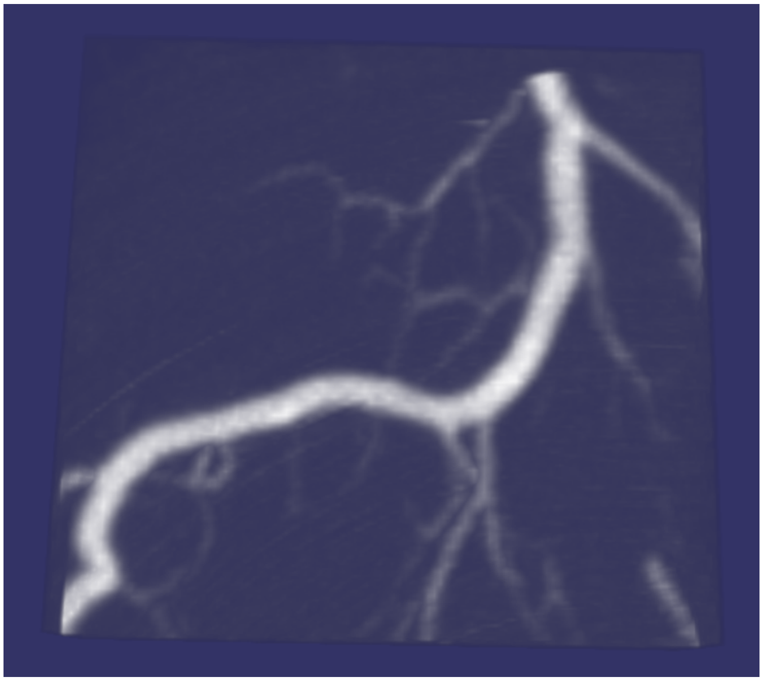

Figure 4.3: The 3D visualization results with MIP on the same cube, the structure is quite clear.

selecting the points with maximum intensity.

Figure 4.2 shows several slices of an input data cube, of size 102x96x34, and these slices are cross sections. There are 34 slices and we show six of them, namely slices numbered 3,13,17,23,27 and 31. From these slides, it's impossible to figure out the structure of the vessels, but from the MIP projection, the structure becomes clear, which is illustrated in Figure 4.3.

However, the 2D MIP projection looses depth information, and cannot provide complete information about the original 3D data. To improve the sense of 3D, in our program we allow to rotate the 3D data, during the rotation we continue rendering the new MIP results under a different view. This way, the users can perceive the relative 3D positions of the data components. So we use the orthogonal projection, no adjustment for distance from the camera made in this projection, so that objects on the screen will appear the same size no matter how close or far away they are. There are also some drawbacks in orthogonal projection, it is difficult for user to distinguish between left and right, front and back. However, as we can rotate to a proper angle if we are confused, these drawbacks are no longer important for our application.

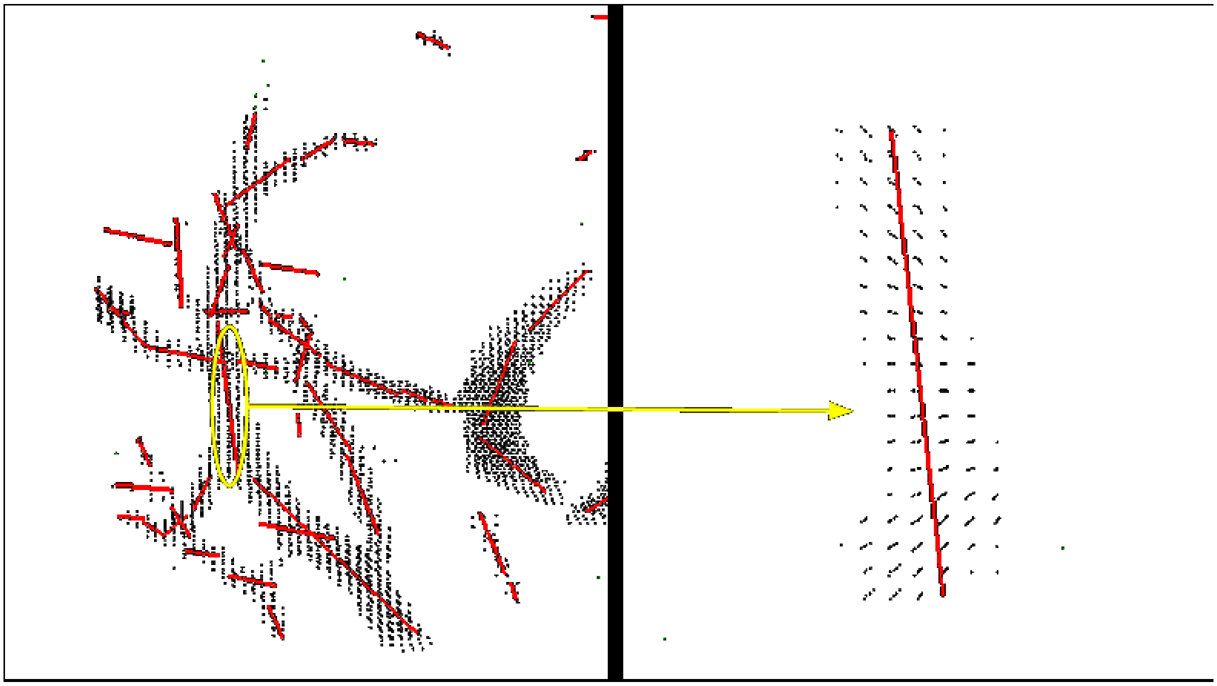

Figure 4.4: Left: the 3D scene contains vesselness points and the line segments fitted to these points. The yellow circled area shows our segment of interest. Right: the detailed result after isolating the segment of interest, now the rotation center is reset to this segment, notice that the rotation center has moved to this segment, which enables us rotate and observe it more clearly.

4.3 Isolating 3D Objects of Interest

Beside 3D visualization, we also need some assistant tools for visualizing and debugging the model fitting results. For example, if we want to know how well a specific segment fits the vessels points that are assigned to it, we need to rotate this segment to an appropriate angle, and zoom it into a proper scale, so that we can observe the fitting details more clearly. If we try to perform this test on the full model, this is cumbersome for the user and also computationally inefficient, especially when there are many segments in the results. Usually, there will be more than 200 line segments for fitting a medium data set (200*120*100), and a lot of segments overlap with each other. The overlap makes it more difficult to observe the fitting quality, even after careful rotation and zooming in.

So we developed a 3D segment isolation tool to solve this problem. The isolation tool is quite easy to use, the user just need to double click the segment of interest, then other segments will be hidden from the view and the chosen segment will be displayed on the screen alone, together with the vessel points assigned to it. If the user double click this segment again, then other hidden segments will be visible again. Figure 4.4 shows an example of using the isolation

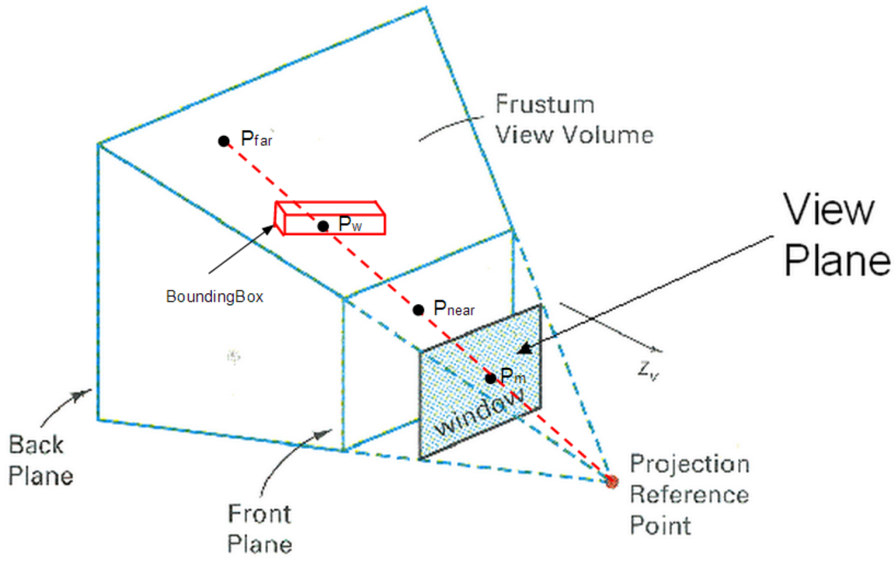

Figure 4.5: The figure of the mechanism for object isolation. The truncated pyramid is the perspective projection view volume, objects outside this volume will not be rendered. The red rectangle is a the bounding box surrounding a line segment. We first build $l(P_{near}, P_{far})$, then visit all nodes within this volume to find the intersected objects, which should be the surrounding box of the segment of interest. In OSG the surrounding box shares a same parent node with its surrounded segment, so we can easily get the pointer of the segment.

tool.

To implement the isolation tool, we need to transform the coordinates of the clicked points. There are two important coordinate systems in OSG, the world coordinates and the screen coordinates. The world coordinate system is also known as the universe coordinate system. This is the base reference system for the overall model in 3D, to which all other model coordinates relate. The screen coordinate system refers to the physical coordinates of the pixels on the computer screen, based on current screen resolution. When clicking on the screen, we can get the mouse position x and y. Let the point P_m(x,y) belong to screen coordinate system. We need to convert it to a point P_w in world coordinate system. To perform the transformation, we need to use a series of matrices, such as projection matrix and model view matrix, which are provided in the public interfaces of OSG.

In Figure 4.5, we build a ray that starts from the project reference point and passes through P_m. Its intersection point with the Front Plane is p_{near}, the intersection point with the Back Plane is p_{far}. We can calculate p_{near} and p_{far} by back projection with the projection matrices. The world coordinate point P_w is located on the line segment $l(p_{near}, p_{far})$. We visit all the nodes to find the nearest object intersected with $l(P_{near}, P_{far})$. This is computationally fast in

OSG due to its scene graph structure, we only need to visit nodes in the view volume, not all scene graph.

To isolate a segment, the user need to click right on it. This is quite a tough task as the segments are too ‘thin’ to be clicked on easily. To solve this problem, we add a hidden bounding box for each line segment. Each box is added to the direct parent node of its corresponding segment. If the box is clicked, we can easily find out its corresponding segment as they share the same parent in the scene graph. The surrounding boxes are set to be invisible, so that the user is not distracted by them.

4.4 Window Center Adjustment

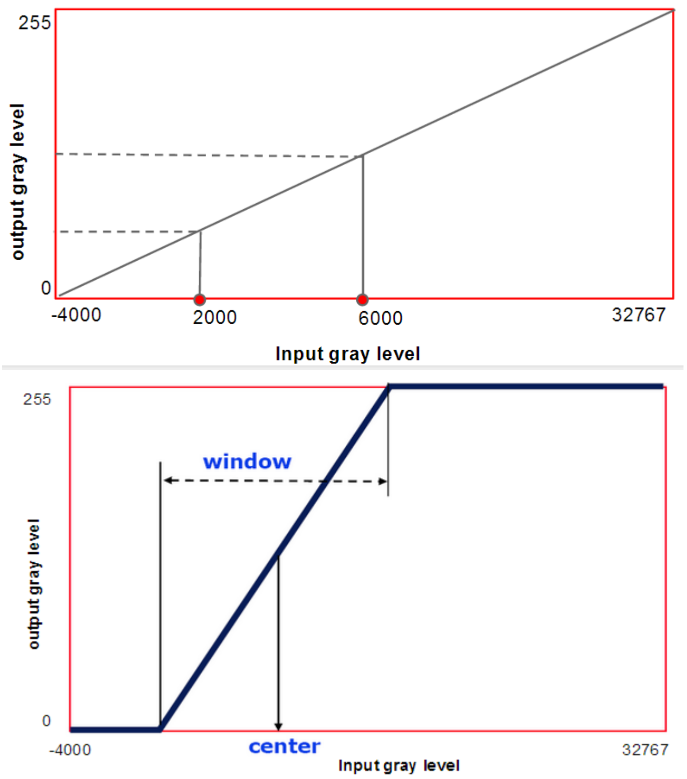

Figure 4.6: Suppose the full intensity range of the data is between -4000 and 32767, and most of the voxels have intensity range between 2000 and 6000. When we use linear interpolation, pixels in the range between 2000 and 6000 get mapped to a very narrow range of output intensities. If we use Window Center adjustment (bottom image), pixels in the range from 2000 to 6000 get mapped into the full output range.

The intensity of each voxel in the original 3D image ranges from -4000 to 32767. However, the monitor’s dynamic range is between 0 and 255. The input gray level greatly exceeds the

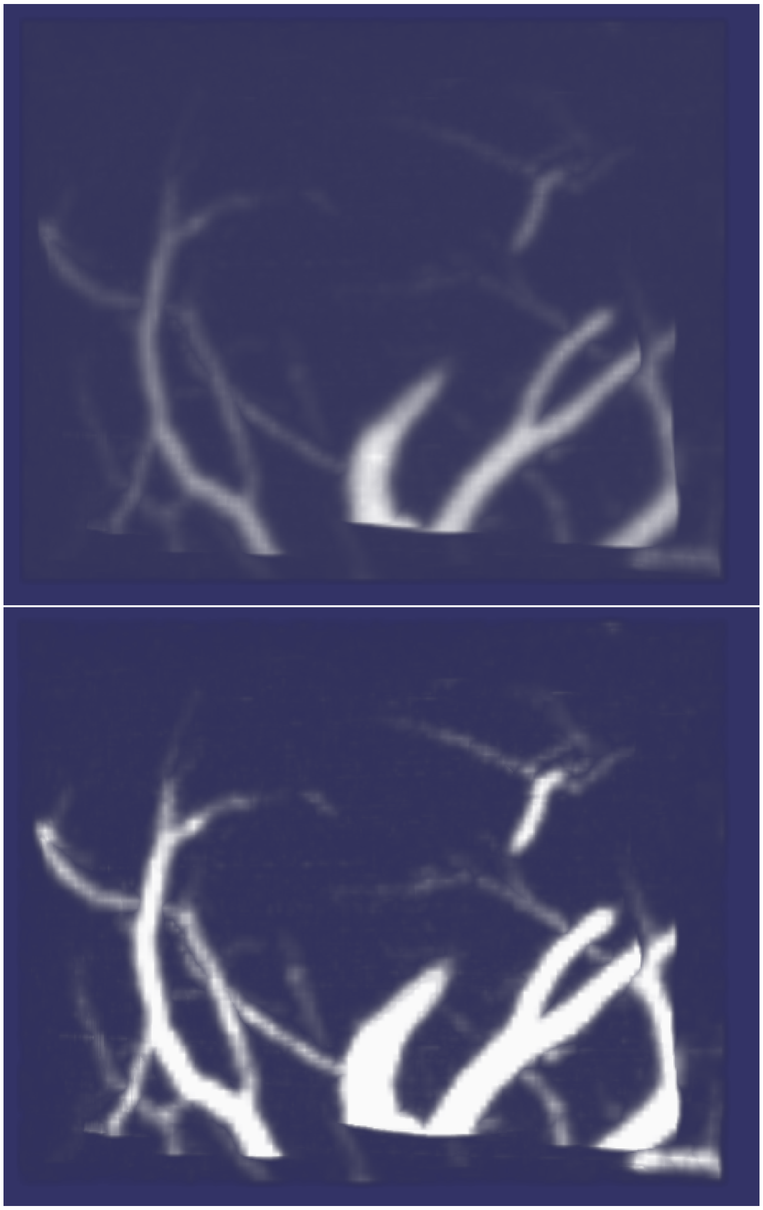

Figure 4.7: A comparison of visualization results with and without Window Center Adjustment. Here we map voxels whose intensity ranges between 2000 and 6000 to the whole output range. After adjustment, the thin vessels are brighter than before.

output gray level, therefore, when we map the output gray range to the input gray range, we are likely to loose a lot of important details, as shown in Figure 4.6. Most voxels have intensity in the range from 2000 to 6000, but after mapping them to a narrow range of intensities, lots of important details were lost.

We use a window center adjustment approach to solve this problem. We set a window on the input gray level, centered in the middle of the range of interest. The window size should not exceed the interest range size. Then we map the input gray level within the window to 0-255, according to Equation 4.1.

$$f(I) = \begin{cases} 0 & I < I_c - W/2 \\ 255 & I > I_c + W/2\ . \\ (I - I_c + W/2) \cdot (255/W) & \text{otherwise} \end{cases} \tag{4.1}$$

where I_c is the center of the window, W is the window size, I is the input intensity value. With window-center approach, we can assign our input range of interest to the largest possible output range. Therefore the details are easier to observe.

Chapter 5

Piecewise Line Segment Model

To reliably extract the vessel structure from the input data, we use a model fitting approach in the energy minimization framework. A model fitting approach has an advantage that it imposes a strong prior, or regularization, on what the vessels should look like. A prior on the data helps to extract the structure more reliably, since unlikely, under the prior (model) configurations are prohibited. Using energy optimization framework for model fitting is a principled approach that allows us to incorporate all the desired problem constraints into an objective function which is then minimized. In addition, we are able to use graph cut algorithms for optimization which have proved to be successful for a variety of vision problems.

We cannot fit our model for the vessel structure to the raw input data directly. The model should be fitted only to those voxels that have a high chance of actually being a vessel. Since vessels have brighter intensity, we could threshold the intensity data and fit the model to voxels that are above the prescribed threshold. However, intensity information is not so reliable. The vesselness measure, discussed in Section 2.1 is much more reliable to detect vessel voxels. Therefore we compute the vesselness measure at each voxel, threshold it, and fit the model to the voxels that are above some prescribed threshold of the vesselness measure. Throughout the thesis, we often refer to these voxels as ”thresholded vesselness voxels” or simply ”vessel voxels”. The work of computing vesselness measures from the raw data is finished by a partner in the lab.

We model a blood vessel as a structure consisting of a sequence of straight line segments. This is a good approximation provided that line segments are of appropriate length. To have a good fit, for a very curved blood vessel, a sequence of shorter line segments are required. For a less curvy blood vessel, a sequence of longer line segments will suffice. We formulate a global energy function that reflects the goodness of fit of the piecewise linear segment model and use the graph cut algorithm for optimization.

5.1 Overview

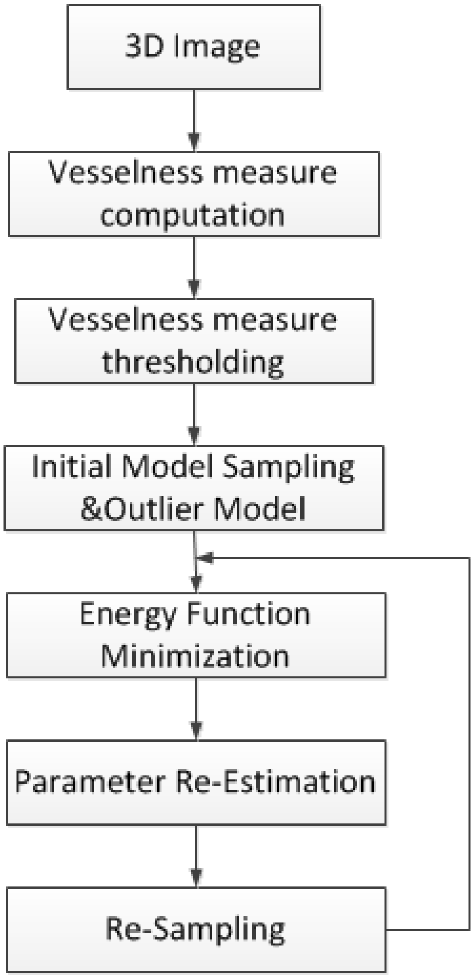
3D Image
Vesselness measure computation
Vesselness measure thresholding
Initial Model Sampling &Outlier Model
Energy Function Minimization
Parameter Re-Estimation
Re-Sampling

Figure 5.1: The flow diagram of our line segment fitting approach.

Figure 5.1 shows the flow diagram of our approach. Our approach is based on fitting line segments to the thresholded vesselness voxels. Each label corresponds to a particular line segment. Since the number of possible lines segments is infinite, and the number of possible segments is finite, in the discrete energy minimization approach, we need an efficient approach to handle labels. We start by sampling a set of line segments (labels) that have a good chance of being a good fit. We use a density based random sampling strategy with angle agreement. The random sampling strategy is to propose line segments that directly connect vessel voxels. The density based approach samples more segments around thin vessels and less around thick vessels. This way, we have approximately the same number of line segments per length of the vessels for the thin and for the thick vessels. This prevents undersampling in the area of thin vessels and oversampling in the area of thick vessels. Oversampling is not a problem, other than computational efficiency. Undersampling can result in less accurate results. The angle agreement makes sure that the sampled line segments agree with the vessel orientation estimated by the vesselness measure, recall that we explained vessel orientation in chapter 2.1. Such line segments have a higher chance of being a good fit for the data. In addition to the regular line segment models, we also use an outlier model to fit outlier points, such as noise.

We build a global energy function based on line segment model, which consists of data cost, smoothness cost and label cost. Our energy function is designed to encourage a good geometrical fit of vesselness voxels to the line segments (through the data terms), nearby line segments to have similar orientations (through the smooth term), and to use not too many line segments in the fit (through the use of label cost). Then we minimize the global energy function via graph cut algorithm. In our work, we use two different smoothness terms: one is the Potts model [46] and the other is the piece-wise smooth model [46]. We use two graph cut based optimization algorithms: $\alpha-$expansion to optimize the global function for the Potts model, and $\alpha-\beta$ swap for the piece-wise smooth model.

The output of a graph cut algorithm is an approximately optimal labeling result, which partitions the input vesselness points into several clusters. Each cluster consists of all voxels that are assigned to the same label, that is to the same line segment. Since the set of the initial line segments may have been far from optimal, we use these initial labeling results to find a possibly better set of line segment models. That is we use the vessel points currently assigned to a line segment to estimate a better line segment through this set of points. This gives us a better label (line segment) than the current one assigned to these vessel points. We use a local greedy search to estimate the line segment through a set of points.

After parameter re-estimation, we update the set of labels, that is the set of line segments with the re-fitted models, and remove any line segments that failed to get any support voxels. We also need re-sample additional line segments around the outliers. Ideally, all outliers should be noisy points and do not need to be fitted by a ’regular’ segment. However, there are usually still some ‘good’ vessel voxels that are misclassified as outliers, if there are no initial segments near them. The density based sampling strategy cannot guarantee enough initial segments for all vessels, unless we sample much more segments than needed. For computation efficiency, we use a cheaper approach, that is we sample more line segment models around the regions with outlier labels after each iteration. Usually there is not many outliers, so we do not need re-sample too many new segments.

Then we minimize the global energy function again, using the updated set of labels. This process is iterated until the convergence of the energy. It is guaranteed that the energy is decreased at each step, since re-estimating line segments brings both the data and the smoothness cost of the energy down.

5.2 Model Sampling

5.2.1 Random Sampling Strategy

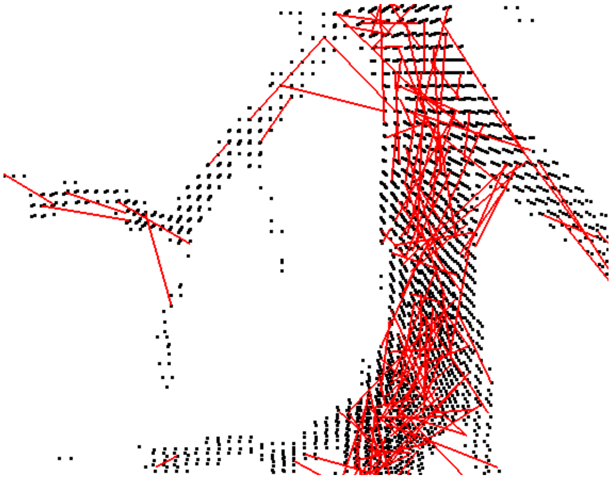

Figure 5.2: Random sampling result on a image data of size 102x96x34. There are many segments around thick vessels, but very few segments around thin vessels.

We now discuss our random sampling strategy for the initial set of line segments (labels). We first randomly choose one point P1 from vessel voxels and take it as one endpoint of the segment. Then we build a size-adjustable cube centered at P1 and randomly select voxel P2 from all the vessel voxels that fall into this cube. P1 and P2 construct a new line segment $L_{P1,P2}$. We do not want very long or short segments. By setting the cube size properly we can control the maximum length of $L_{P1,P2}$. For short segments, we check the length of $L_{P1,P2}$. If its length is smaller than a threshold, we discard it and select P2 again. This is repeated until we find a line segment of proper length. If this step has been repeated too many times (higher than a threshold), and we still can not find a proper segment, we will re-select P1 as the old one is probably an isolated noisy point, and, therefore, we do not want any candidate line segments around it.

5.2.2 Density Based Random Sampling

From practical experiments, we found that we need to sample around 400 line segments for a dataset whose size is about 100x50x80. However, we notice that thick vessels often get most

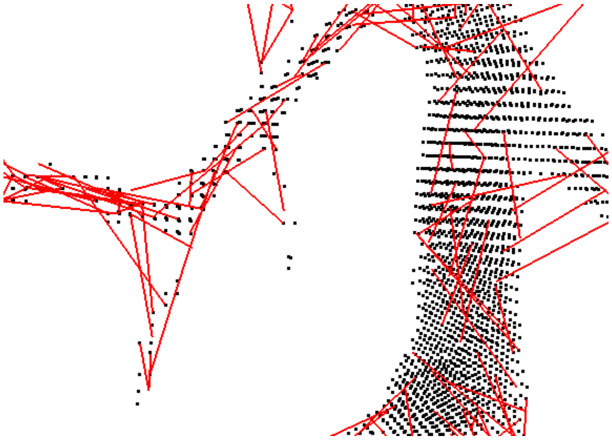

Figure 5.3: Sample results based on the density of vessel points. This strategy samples less points around thick vessels and more points around thin vessels, making the overall distribution of line segments more equal throughout the dataset. This ensures that there is neither oversampling (leading to longer computational time) no undersampling leading to poor model fit).

segments, while thin vessels get few. This problem is illustrated in Figure 5.2. The reason is that we only choose endpoints of segments from vessel points, and since thick vessels contains more vessel points, they get more line segments sampled around them. On the contrary, thin vessels do not get enough line segments, some thin vessels have no nearby line segment at all. The situation is especially bad for the endpoints of the thin vessels. If there are no nearby line segments around a vessel, all its points will be fitted by an outlier model, since an outlier model fits, albeit at a higher than normal cost, to any vessel point.

To solve this problem, we use a new sample strategy based on taking the density of vessel points into consideration. We first build a density map before sampling, for each vessel voxel, stores the number of other vessel voxel in the neighborhood of some fixed size. Now, when we sample point P1, we sample with probability inversely proportional to the density map. That is the higher the density, the less likely we are to sample that point as P1. This way, we can sample more segments around thin vessels, which is illustrated in Figure 5.3.

5.2.3 Random Sampling with Angle Agreement

The above approach does not take the inner parameters of line segment into consideration, such as angle. So the direction of segments is not controlled. However, we know in final result the line segment should have similar direction to the direction of the vessel that it fits. For each vessel voxel, we have the estimation of the vessel direction from the vesselness measure, see Section 2.1. Therefore we add the angle agreement between the sampled line segment and the

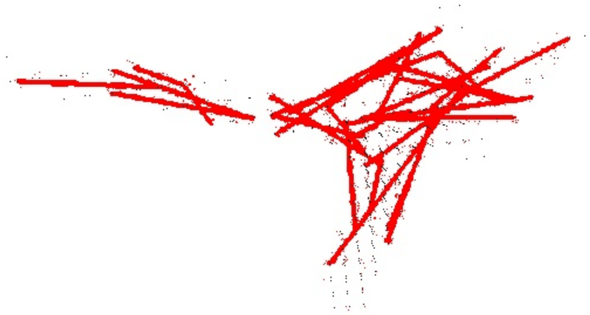

Figure 5.4: Sample result without angle agreement.

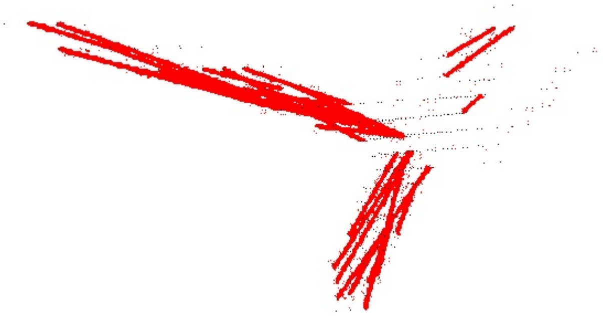

Figure 5.5: Sample result with angle agreement.

estimated vessel orientation. This is done as follows. After we generated a new line segment, we calculate the angle β between the average vessellness orientation measure of the segment endpoints and the orientation of the line segment (P1,P2). If β is bigger than 15 degree, we believe the direction of this segment is too different from the orientation of the vessel that its endpoints pass through, so we discard P2 and select a new one. If there is no proper P2, we re-select P1 as P1 may be a noise point.

Figure 5.4 shows the sampling result without vessel angle agreement. These line segments have random orientations that are far from the orientations of the underlying vessels. Figure 5.5 is the result with angle agreement, most sample segments have similar directions with the underlying vessels.

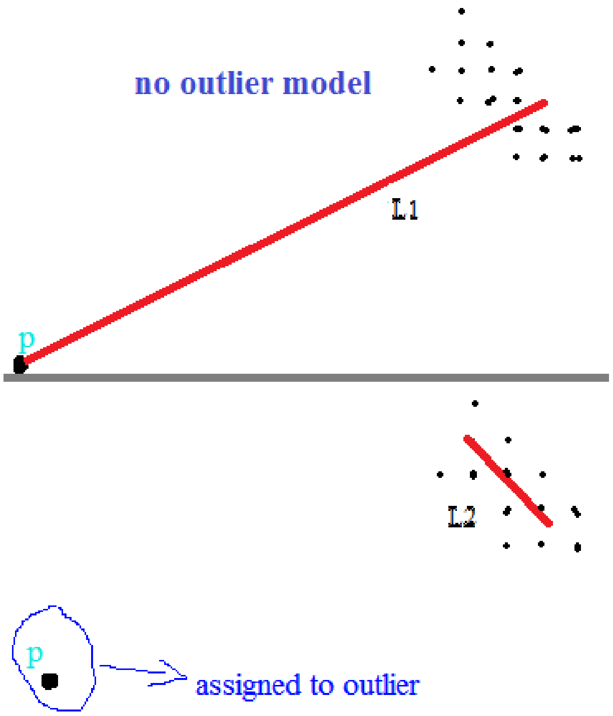

Figure 5.6: Result comparison with and without outlier model. In the first image, a wrong segment L1 fits both inlier and one outlier point marked with ”p” in the image. . In the second image the outlier model was added and L2 only fits the inliers. The data cost between noise P and L2 is very high, if there is no outlier model, L1 is definitely a better option than L2.

5.2.4 Outlier Model and Re-Sampling

There is a lot of noise in the original image data, and noise points should not be fitted by any line segments. If we have no outlier model, noise points will have to be fitted by valid line segments. If the line segment also contains true vessel points, not just the noise points, the segment is likely to deviate from its optimal position as illustrated in the first image of Figure 5.16.

To solve this problem, we add an outlier model to the set of valid segment labels. An outlier model is a special label, and it is not actually a segment. Its meaning is as follows: any point assigned to an outlier label is considered not to belong to any line segment. The data cost of the outlier model is set to a constant value $T_{outlier}$. Usually the distance (that is geometric fit) between a noise voxel and the line segment is much longer than the distance between the true vessel voxel and a segment. So the data cost of noise voxel is much higher than that of a true vessel voxel. If we set an appropriate $T_{outlier}$ value, the noise voxels will prefer to get assigned to the outlier model instead of a regular line segment. Suppose T_{ave} is the data cost averaged among true vessel voxels, T_{error} is the average data cost if an outlier voxel is assigned to its nearest line segment. Then $T_{outlier}$ should be much higher than T_{ave}, but lower than T_{error}.

Outlier model is not only useful for the noise voxels, but also it can temporarily accept those vessel voxels that are currently not well fitted by any segments. There are always some tiny vessels that failed to get enough models sampled around them, even with the density based sampling approach. They can be labeled as outliers ”temporarily”, and we could re-sample models around the regions with outlier labels at the end of each iteration. Figure 5.6 shows a comparison of results with and without outlier models.

5.3 Global Energy Function

We build a global energy function based on the line segment model, which consists of data cost, smoothness cost and label cost.

$$E(f) = \sum_{p\in\mathcal{P}} D_p(f_p) + \sum_{pq\in\mathcal{N}} V_{pq}(f_p, f_q) + \sum_{l\in\mathcal{L}} H(l)\cdot\theta_l(f) \tag{5.1}$$

where $\sum_{p\in\mathcal{P}} D_p(f_p)$ is the data cost, $\sum_{pq\in\mathcal{N}} V_{pq}(f_p, f_q)$ is the smoothness cost, $\sum_{l\in\mathcal{L}} H(l)\cdot\theta_l(f)$ is the label cost. In the following sections we will introduce how these three energy terms are designed in our work.

5.3.1 Data Cost

Data cost measures the cost of assigning a label to a vesselness voxel. In this thesis, the data cost is how well a vessel voxel fits to a line segment. Each three dimensional line segment L_{ab} has parameters $\theta_{Lab} = \{a, b, \sigma\}$, where a, b are endpoints of the segment, σ^2 is the variance of the data, the initial σ is set to a small constant value. We here model a line segment from point a to point b as an infinite mixture of isotropic Gaussian $\mathrm{N}(\mu, \sigma^2)$ for each μ interpolating a and b [10]. The probability that point p belongs to L_{ab} are shown in equation 5.2.

$$Pr\left(x|a, b, \sigma^2\right) = \int_0^1 \mathcal{N}\left(x|\,(1-t)\,a + tb, \sigma^2\right) dt. \tag{5.2}$$

q belongs to the line segment L_{ab}, x is a vessel point, the probability that x appears near q should follow the gaussian distribution with mean equal to the point q. We assume that q follows a uniform distribution on segment L_{ab}. Figure 5.7 shows how to calculate the data cost on this segment model. This particular model is chosen because it encourages endpoints a, b not to extend too far beyond the actual vesel voxels that are assigned to this line segment. That is a

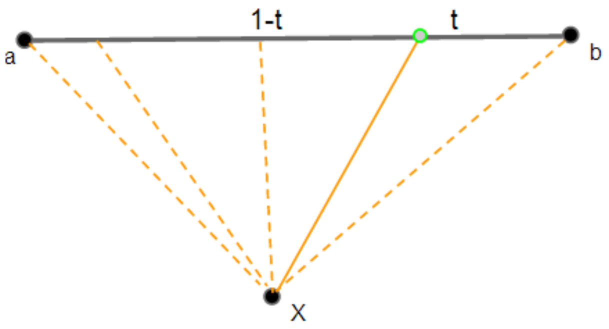

Figure 5.7: Line segment model. Here t is a point on the segment, x is a vessel point. We assume that the probability of x to appear at some distance from t follows gaussian distribution, and t is evenly distribution on the segment. So the data cost is an infinite mixture of Gaussian distributions.

'compact' line segment is encouraged for a set of vessel points.

In three dimensions, the above integral evaluates to

$$\begin{aligned} Pr\left(x|a,b,\sigma^2\right) = \frac{1}{4\pi\sigma^2||a-b||} \cdot exp\left(-\frac{D_x^2}{2\sigma^2}\right) \\ \cdot\left(erf\left(\frac{(x-b)\cdot(a-b)}{\sqrt{2}\sigma||a-b||}\right) - erf\left(\frac{(x-a)\cdot(a-b)}{\sqrt{2}\sigma||a-b||}\right)\right) \end{aligned} \tag{5.3}$$

where erf(.) is the error function, which is also called the Gauss error function. It is related to the cumulative distribution Φ, the integral of the standard normal distribution by $\Phi(x) = \frac{1}{2} + \frac{1}{2}erf(x/\sqrt{2})$. D_x is the orthogonal distance from x to L_{ab}.

Given a set $X_l = \{x_p : f_p = l\}$ of inliers for label l, we calculate the maximum-likelihood estimators of $\theta = \{a, b, \sigma\}$ by minimizing the negative-log likelihood function

$$D_p(f_p) = -\sum_p logPr(p|a,b,\sigma^2) \tag{5.4}$$

E_{data} is our data term, but it is difficult to compute the derivative of the erf function. So we do not use gradient descent to estimate the parameters, instead we use a local greedy search based approach to calculate the parameters, which will be discussed in the following section.

We also tested a simplified model, which measures the probability that point x belongs to an infinite line model L, as opposed to a line segment. In 3D each line has parameters $\theta_l = \{A, B, C, D, \sigma\}$ where $Ax + By + Cz + d = 0$ defines the line and σ^2 is the variance of data. The parameters A, B, C, D have been scaled so that $A^2 + B^2 + C^2 = 1$. The initial line models are generated the same way as line segments models, by selecting two random points from vessel points and fitting A, B, C, D accordingly. The initial σ is set to a small constant value. So the

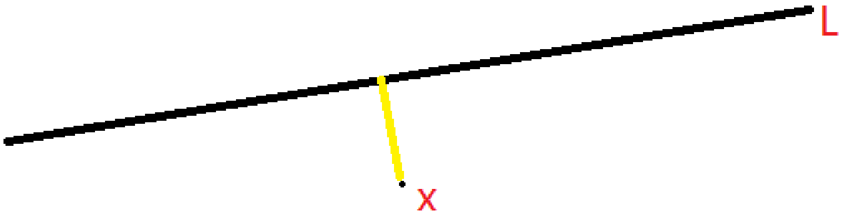

Figure 5.8: Line Model.

data cost of line model is

$$D_p\left(f_p\right) = -log\left(\frac{1}{\sqrt{2\pi}\sigma}exp\left(-\frac{(Ap^x + Bp^y + C)^2}{2\sigma^2}\right)\right) \tag{5.5}$$

This line model can result in line segments that extend far beyond the actual vessel voxels assigned to it, see Figure 5.9. We will compare the results of the two models in the experiment section.

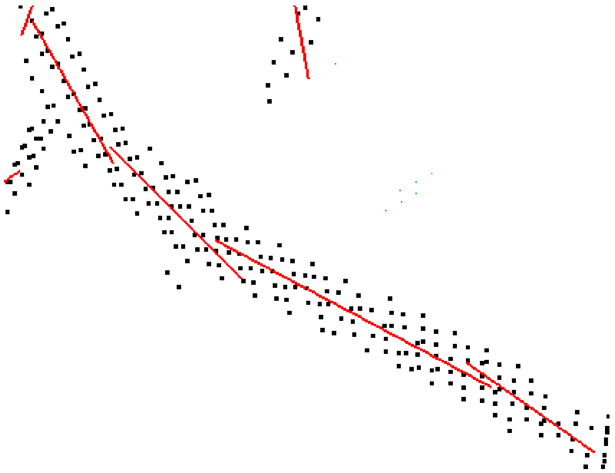

Figure 5.9: Result generated by line model based approach.

5.3.2 Smoothness Cost

We tested two different smoothness terms: one is based on Potts model, the other is based on piece-wise smooth model. Potts model is the simplest smoothness term that allows discontinuities. It can be modeled by the following neighbor interaction function:

$$V_{p,q} = u_{p,q} \cdot T(f_p, f_q) \tag{5.6}$$

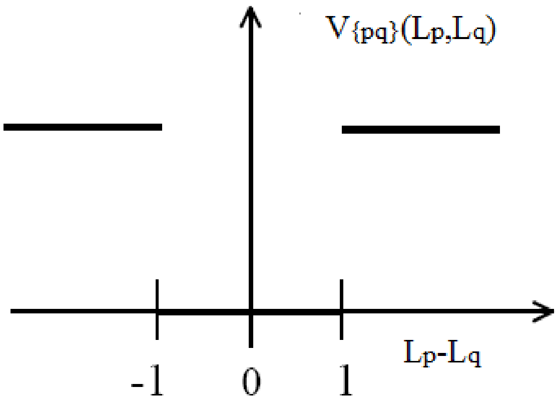

Figure 5.10: The Potts model, here L_p and L_q are two different labels. In our work, we set a constant penalty T_{potts}. If two points belongs to different labels the penalty is T_{potts}, otherwise the penalty is 0.

where

$$T(f_p, f_q) = \begin{cases} 1 & (f_p \neq f_q) \\ 0 & (otherwise) \end{cases} \tag{5.7}$$

This neighbor interaction function gives penalty $u_{p,q}$ if two neighboring points are assigned different labels. $u_{p,q}$ does not depend on the size of the difference in labels L_p and L_q, what matters is if they are different or not. The graph of Potts model is shown in Figure 5.10.

The Potts model encourages a labeling consists of different regions, where voxels in the same region have the same labels. Therefore, Potts model is informally called a piecewise constant model. In our work, the Potts model provides a smoothness penalty between different line segments, that is voxels that belong to the same vessel tend to be assigned to the same label to lower the smoothness cost. However, the Potts model does not care about about the difference/similarity between different labels. Different labels, even if they are very similar, are penalized by the same amount as different labels that are very different. This leads to a problem in our fitting result, which is shown in Figure 5.11.

We want to relate the smoothness cost with the inner parameters of the line segment, such as direction and position. If two line segments are neighbors, they usually belong to the same underlying vessel, and, therefore, they should have similar orientation. Besides, the distance between the two segments should also be a small value. To model this, we use a piecewise smooth model as the smoothness term.

In piecewise smooth model, $V(L_p, L_q) = max(T, \theta(L_p) - \theta(L_q))$, where $\theta(L_p)$ and $\theta(L_q)$ are parameters of segments L_p and L_q, such as angle and spatial coordinates. T is a constant value to truncate the cost. The piecewise smooth model encourages labeling consisting of several

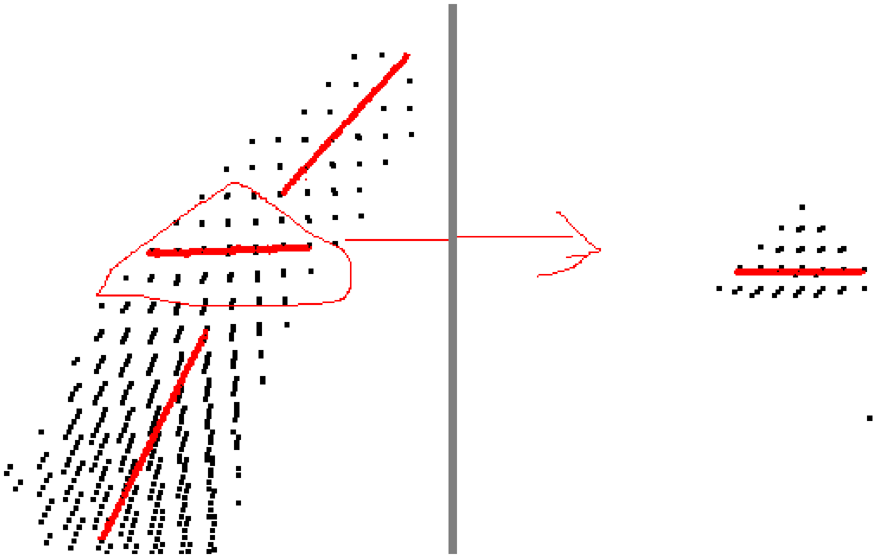

Figure 5.11: A snapshot of the Potts model result. The triangular shaped area is fitted by a horizontal line segment , however, we want a vertical segment as neighboring segments should have similar directions. The energy of the Potts model is not related to the difference/similarity of line segments (labels), so it can not regularize the relative orientations of the nearby segments.

regions where voxels in the same region tend to have similar, but not necessarily equal labels. Figure 5.12 shows a truncated linear piecewise smoothness cost. Such smoothness cost can be used to encourage nearby line segments to have a similar orientation and spatial coordinates, since under this model similar labels cost less than labels that are not similar. .

W use a piece-wise smooth term proposed by Olsson and Boykov [35]. They proposed a curvature optimization framework to approximate surfaces from a cloud of point measurements corrupted by noise and outliers. Instead of using graphic models with higher order interactions, they extended the label space to encode both orientation and position of a curve, as shown in Figure 5.13.

This piece-wise smooth model could integrate both the angle and spatial information in a principled way, which makes it quite suitable to solve our problem. Suppose $\tilde{p}$ and $\tilde{q}$ are two neighboring points, $\tilde{p}$ is fitted by line segment L_p, $\tilde{q}$ is fitted by L_q. Then the smoothness cost based on Olsson and Boykov model is defined as

$$V(\tilde{p},\tilde{q}) = \left(\frac{|p-p'|}{|p-q|^2} + \frac{|q-q'|}{|p-q|^2}\right) \cdot |p-q| \tag{5.8}$$

Where p is the projection of $\tilde{p}$ on segment L_p. Here we assume $\tilde{p}$ is a noisy point around L_p, p is the estimated true position on L_p. Then $\left(\frac{|p-p'|}{|p-q|^2} + \frac{|q-q'|}{|p-q|^2}\right)$ minimizes the curvatures at p and

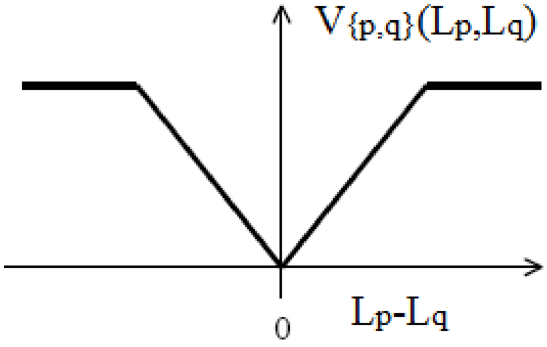

Figure 5.12: A truncated linear piecewise smoothness cost.

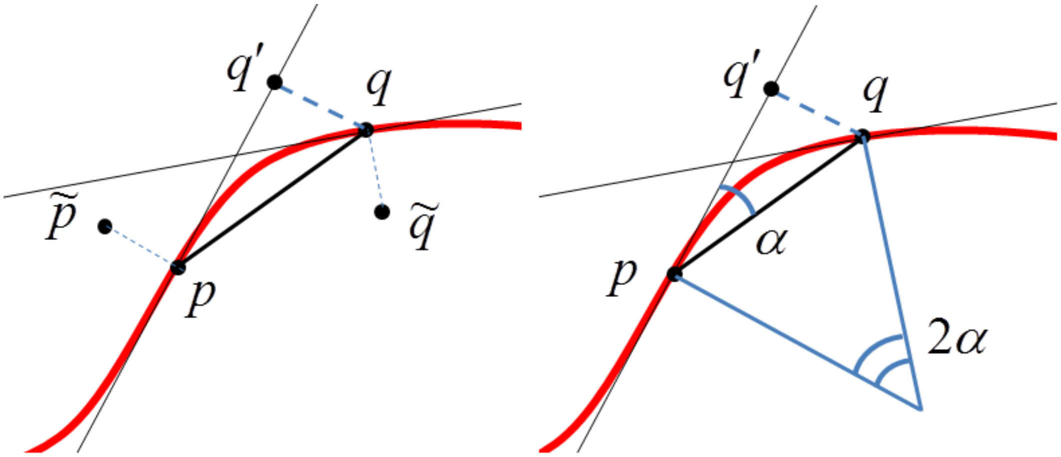

Figure 5.13: A 2D example of Olsson and Boykov model. Points $\tilde{p}$ and $\tilde{q}$ are noisy measurements of points p and q on the underlying curve. The quotient $\frac{|q-q'|}{|p-q|^2}$ yields half the curvature at p under the assumption that p and q belong to a constant curvature segment.

q, where $|p - q|$ minimizes the distance. This way, we ensure neighboring points are on an underlying piecewise smooth curve. The fitting result is much improved compared with Potts model, as shown in Figure 5.14.

5.3.3 Label Cost

The goal of model fitting is to find the best fitting model for each cluster of points. And for each cluster, we need to find the best fitting model which minimizes the data terms of points that belong to this cluster. However, there are two approaches to minimize the data terms, either finding one model which fits the whole cluster very well, or finding multiple models, where each model fits a smaller part of the cluster much more precisely than one model per cluster

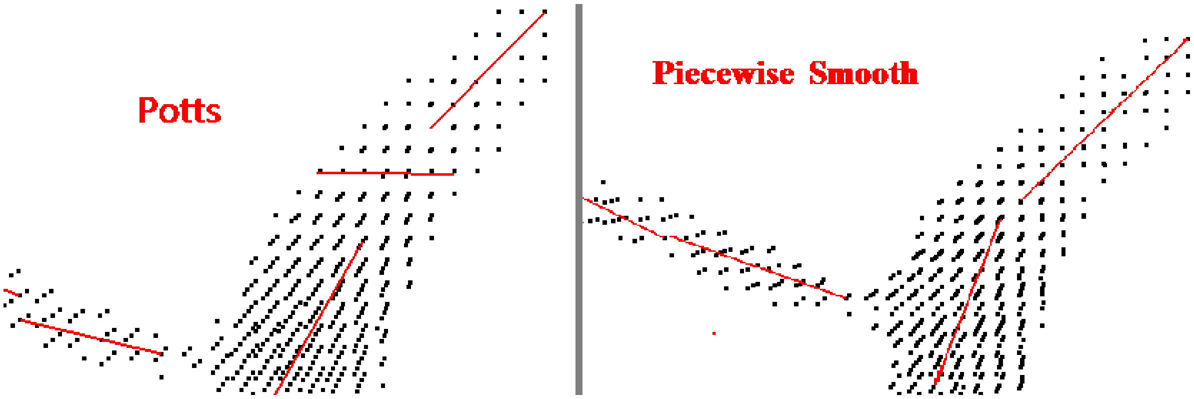

Figure 5.14: Comparison results between Potts model and piece-wise smooth model.

ever could. In the limit, if we add one line segment per data point, we will fit the data perfectly, but will use too many models. We do not want too many line segments fitting one vessel, so we add a label cost to the energy function. That is for each line segment (label) that is being 'used' (i.e. some vessel voxels are assigned to it), we add a penalty T_v. This encourages as few models as possible in the fitting result. If T_v is too small, than too many line segments are used. If T_v is too large, then only very few line segments are allowed. Experimentally, we found that the value of 320 for T_v gives us good results, resulting in neither too few nor too many line segments.

5.4 Parameter Estimation

5.4.1 Exhaustive Grid Search

We first use the graph-cut algorithm to optimize the energy function defined in Equation (5.1) to get an approximately optimal labeling, then estimate the model parameters based on the labeling results. However, re-estimating the parameters of a line segment model is not an easy task, as the derivative of the function in Equation (5.3) is difficult to compute. Instead we use an exhaustive grid search approach to find the approximate minimum solution. Figure 5.15 illustrates how this approach works.

Figure 5.15 is a example of the exhaustive grid search on 2D image data, which is easier to understand than 3D. To fit a point set $\mathcal{S}_0$, we first build a grid covering all points in $\mathcal{S}_0$, Let $\mathcal{S}_g$ be the set that contains all grid corner points and $\mathcal{S}_0 \in \mathcal{S}_g$. We then perform exhaustive grid search for every pair of points (p, q), $p, q \in \mathcal{S}_g$, and $p! = q$. That is we take p, q as the endpoints of line segment $L_{p,q}$, compute the data cost $E_{data}(r, L_{pq})$ of assigning r to $L_{p,q}$ according to Equation (5.3), $r \in \mathcal{S}_0$. We also need to search all possible values of σ, however σ usually varies in

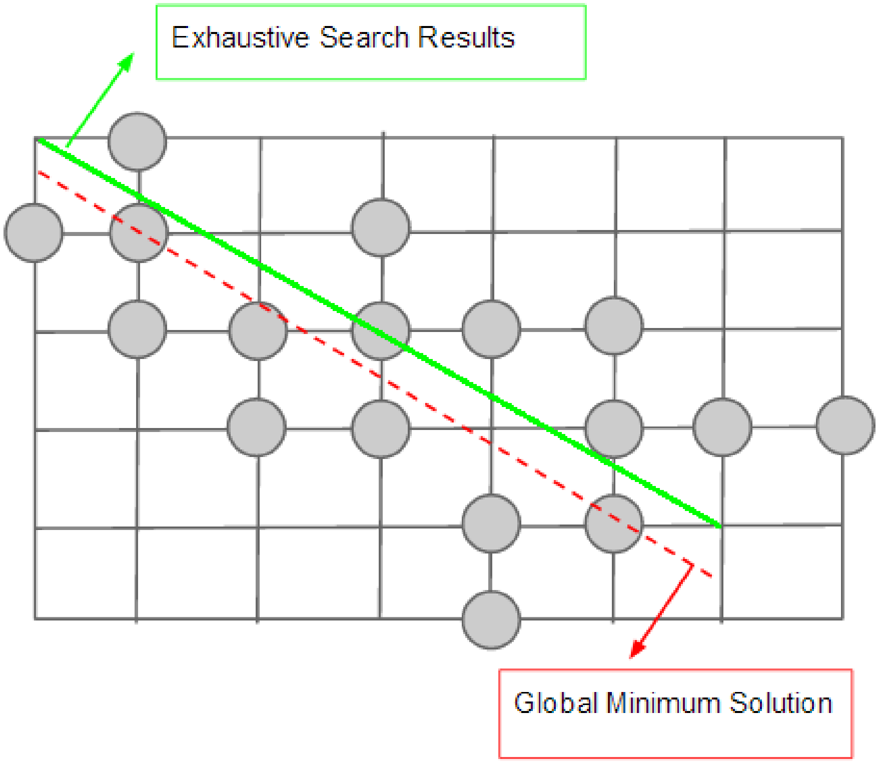

Figure 5.15: 2D example of our exhaustive grid search approach. The grid is our search space, the red line segment is the global minima, the green one is our exhaustive search result. There is some different between the two line segments, as we only search points on the grid corners. But the endpoints of the global minimum solution may be located not exactly on the grid corners. Finding a more precise solution would require a sub-pixel search, which is too computational expensive.

a small range [0.5-5]. Compared to the huge search range of a, b, the influence of σ on the algorithm speed is small.

The data cost of fitting $\mathcal{S}_0$ with $L_{p,q}$ is

$$E_{data}(\mathcal{S}_0, L_{pq}) = \sum_{r \in \mathcal{S}_0} E_{data}(r, L_{pq}) \tag{5.9}$$

The optimal line segment is computed via

$$L_m = arg \min_{L_{p,q}} \left(E_{data}\left(\mathcal{S}_0, L_{pq}\right) + E_{smooth}\left(\mathcal{S}_0, L_{pq}\right)\right) \tag{5.10}$$

5.4.2 Least Squares Based Search

This exhaustive grid search method could generate results which are very close to the globally optimal solution, but too computational expensive. Suppose there are N points in $\mathcal{S}_0$, the exhaustive grid search algorithm complexity is O(N^3). In the experiment it takes about 8 hours to process a small data cube (25x21x25). So we use a faster approach which is based on the least squares method and local search to estimate the model parameters. Least squares is a standard approach to the approximate solution of over determined systems, the name least squares means that the overall solution minimizes the sum of the squares of the errors made in the results.

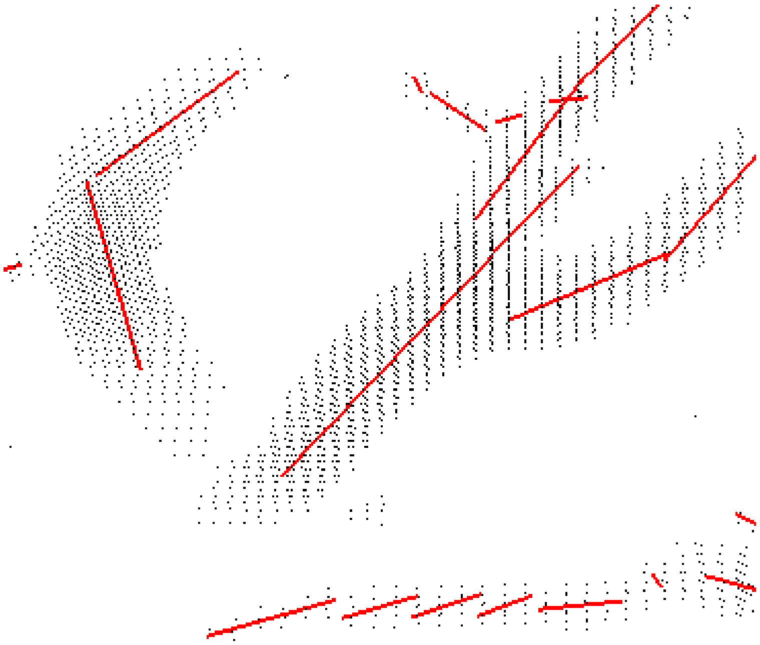

Figure 5.16: A snapshot of fitting results based on the least squares approach. These segments are roughly correct, but their precision is not good enough, as this approach only minimizes part of the energy function in equation 5.3.

We notice that the first part of Equation (5.3) is $\frac{1}{4\pi\sigma^2\|a-b\|}$ and the second part is $exp(-\frac{D_x^2}{2\sigma^2})$. There are three unknown parameters σ,$\|a-b\|$ and D_x. For σ we can simply iterate over a range of likely values, indeed if we set the search step to be 0.5, there are at most 20 possible values in the range [0.5,10]. $\|a-b\|$ is the length of the segment, which is independent from the segment direction. That means if we can find the correct direction of the segment, then the length of the segment can be calculated in the same way as σ. The third parameter D_x is the orthogonal distance from point x to the segment. So we use the least squares method to minimize the sum of D_x^2, which can provide us the optimal direction of segments. The third part of Equation (5.3) is erf function, and least square method can not optimize it. This approach could only generate a local minima, although there maybe some distance to the global minima, as shown in Figure 5.16.

5.4.3 Fast Local Search Approach

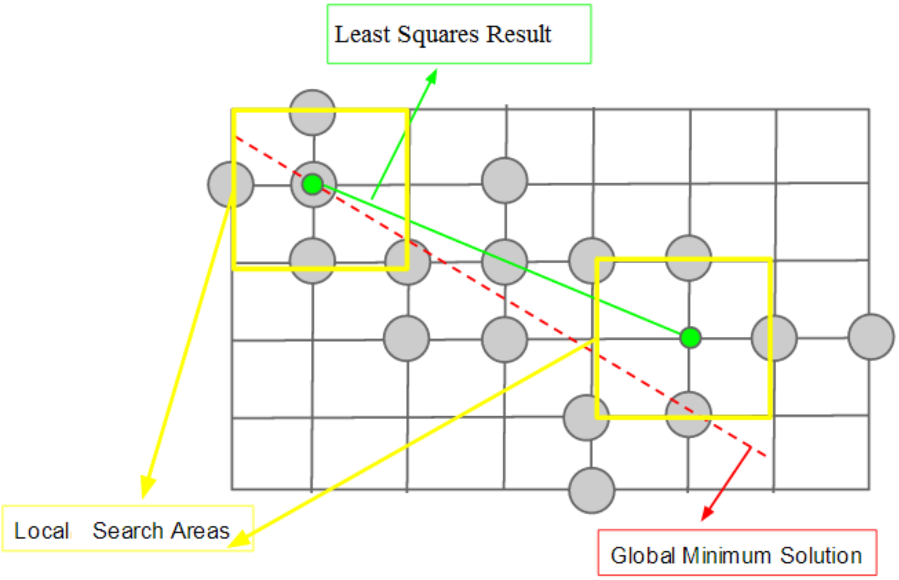

Figure 5.17: The local search approach, the green segment is the least squares solution, the red segment is the globally optimal solution. We locally search the endpoints within the regions surrounded by the yellow boxes. In this example, a globally optimum solution will be found as both the endpoints of the globally optimum solution are inside the yellow boxes.

We are not satisfied with the local optimal solution generated by the least squares approach, so after least squares we use a fast local search approach to get more accurate results. We do not search all points in $\mathcal{S}_0$ due to huge computational expense, instead we search a small area to find a better solution iteratively.

Figure 5.17 shows how the local search approach works. Before running the local we already have a segment L by the least squares approach, which is shown with green color. Then, we build two small boxes around the endpoints of L, and search within those two areas. Specifically, we try all possible voxels in one of the boxes for first endpoint of the interval, and all possible voxels in the other box for the second endpoint. That is, all possible combinations of the endpoints are tried, where the first endpoint comes from one box and the second endpoint comes from the other box. The pair of endpoints that minimizes the objective function is selected, and then this process is repeated again, moving the search boxes around the newly found

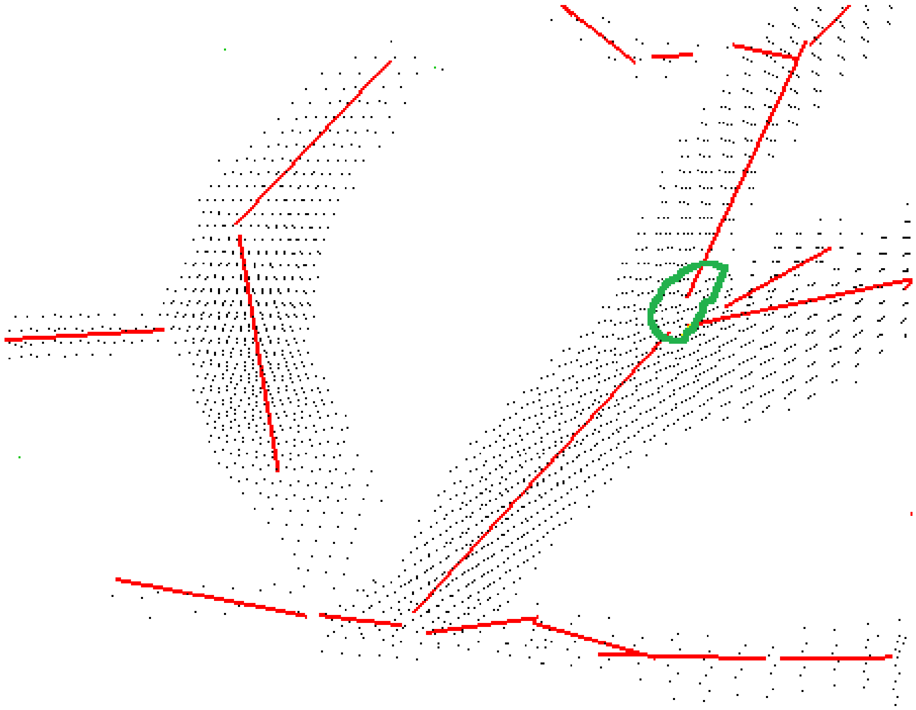

Figure 5.18: A snapshot of fitting result based on local search. The green circle shows much improved results compared with least squares solution in Figure 5.16.

endpoints This process is iterated until convergence, which is guaranteed since the number of points is finite and the energy always goes down. The size of the box is first set to be a larger value, to ensure the algorithm will not be easily trapped into local minima. After several iterations we use a smaller size, as we know we are more likely to be near the global minima. This approach is fast enough to allow sub-pixel precision, that is we can search the grid at floating point precision for the spatial coordinates in the local areas, if necessary. Experimentally, we found that half-pixel precision is sufficient, while one-pixel precision sometimes is not accurate enough. Figure 5.18 shows a snapshot of the local search result.

Chapter 6

Experimental Results

In this chapter, we present experimental results of the proposed algorithm, compare results generated by using different energy terms and analyze the difference.

6.1 Data Term Experiments

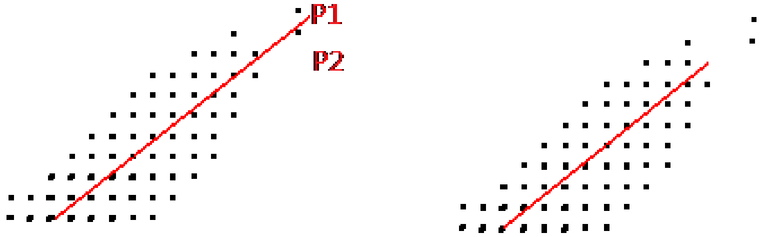

Figure 6.1: Left: fitting result generated by line model. Right:fitting result generated by line segment model.

We test two type of data terms, the first one based on the line segment model, and the second one based on the line model. Figure 6.1 compares the fitting results of the two models on a cluster of points. There is only one label, so the smoothness cost is 0, and the label cost is equal. It is clear that the segment generated by the line model is longer compared to that of the line segment model. This is because there is no 'length' parameter in the energy function for the line model. No matter how long the line segment is, the corresponding data cost in the energy function is the same. As the length is not part of the estimation in the case of the line model, we estimate the length by the following 'guess' strategy. We first build a bounding box for each cluster of voxels, then extend the line until it falls beyond the bounding box. As the

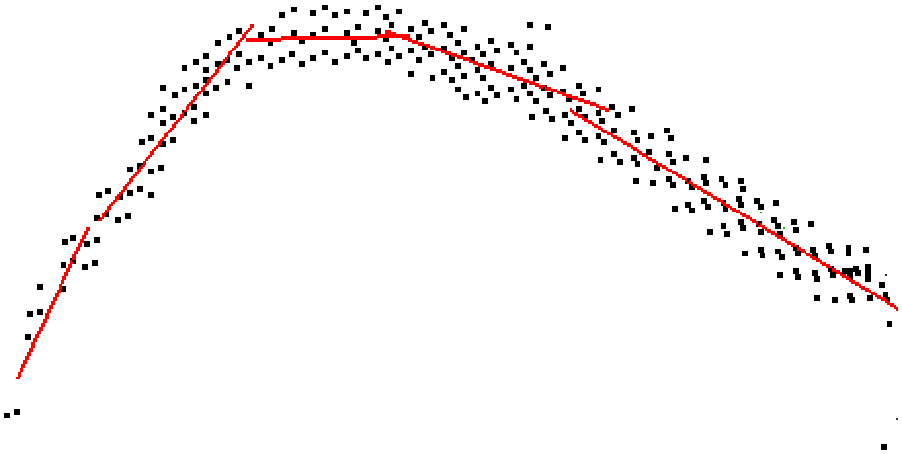

Figure 6.2: A snapshot of fitting result based on the line model. As we have to 'guess' the length, the line segments often extend beyond what they should be.

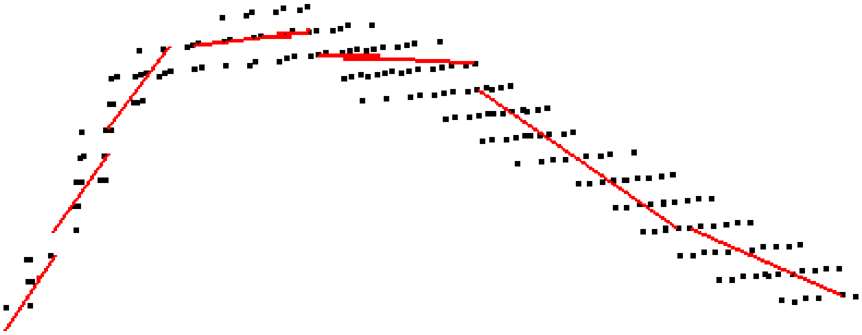

Figure 6.3: A snapshot of fitting result based on the line segment model.

bounding box contains all voxels in the cluster, including noise and outliers, this greedy guess strategy often generates longer line segments. However, there is no such problem in the line segment model. As the length parameter is part of the data term, we can directly compute it by minimizing the energy function.

Figure 6.2 and 6.3 are the fitting results generated by the two model. The smoothness term is based on Potts model, and label cost is set to be the same constant value. As the line model encourages long line segments, there are only 5 line segments in the result. On the contrary, the segments generated by the line segment model are neither too long nor too short, and there are 7 segments. For a greater accuracy of the vessel structure model, the line segment seems to be a better model.

6.2 Label Cost Experiments

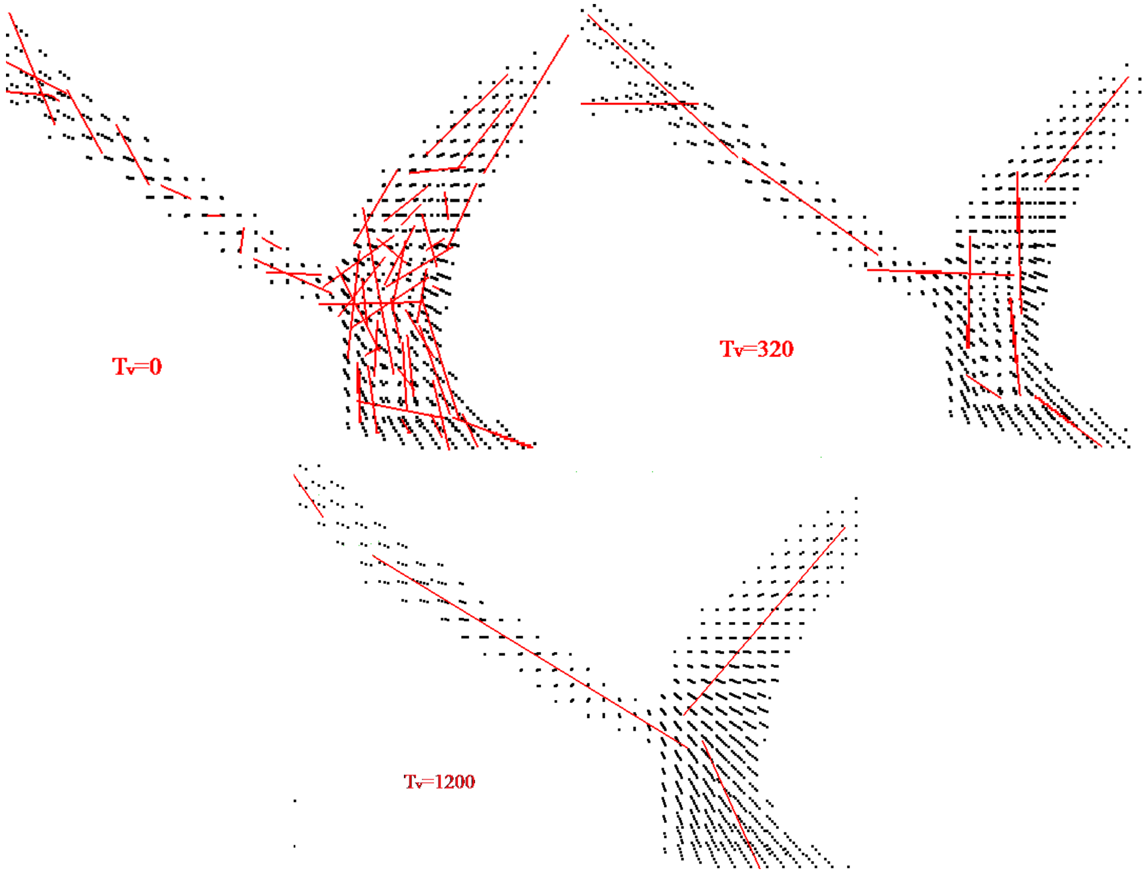

Figure 6.4: Comparison of of results with different label cost. There is no smoothness cost, and the data cost is based on the line segment model. T_v is the value of the label cost. But there is no good value that can generates very reasonable results. Without smoothness cost, label cost alone does not do a good job.

We now evaluate what effect the label cost has on our fitting results. Figure 6.4 is the comparison of results with different label cost. For simplicity of comparison there is no smoothness terms, and the data terms are based on the line segment model. When the label cost $T_v = 0$, there are much too many segments fitting one vessel. This is because the lower the label cost, the more models are allowed. When $T_v = 1200$, there are only 3 segments, each segment fits a very large part of a vessel. From this comparison, we conclude that choosing an appropriate label cost is very important, especially when there is no smoothness cost. It is also clear that without smoothness cost, label cost alone does not generate very good results.

When the smoothness cost is used, the label cost still performs a meaningful job. In Figure 6.5 we use Potts model with smoothness parameter set to 10. We still need the label cost

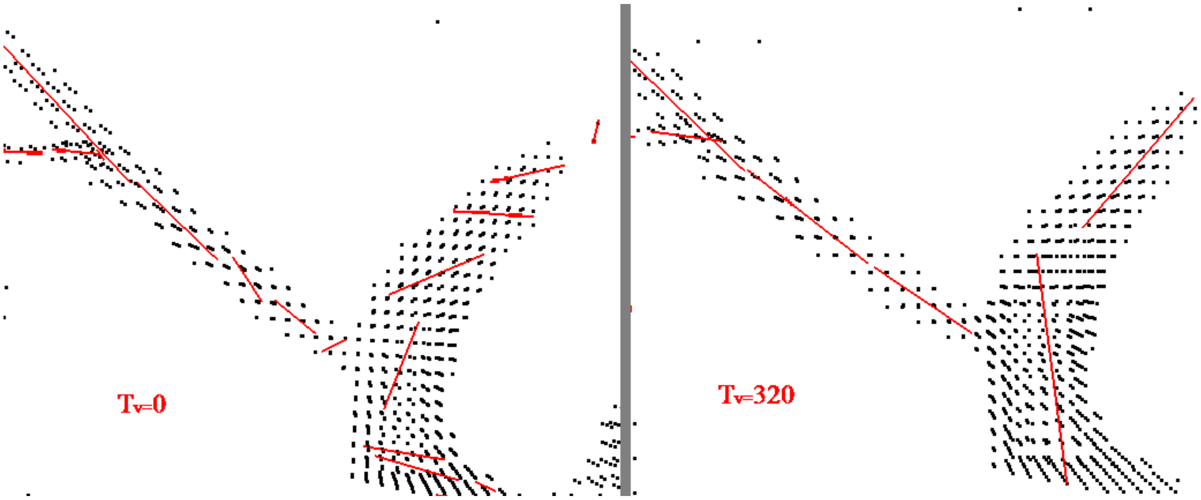

Figure 6.5: Comparison of results without label cost, left ($T_V = 0$) and with the label cost, right ($T_v = 320$). The smoothness cost is set to 10. The result is better with non-zero label cost.

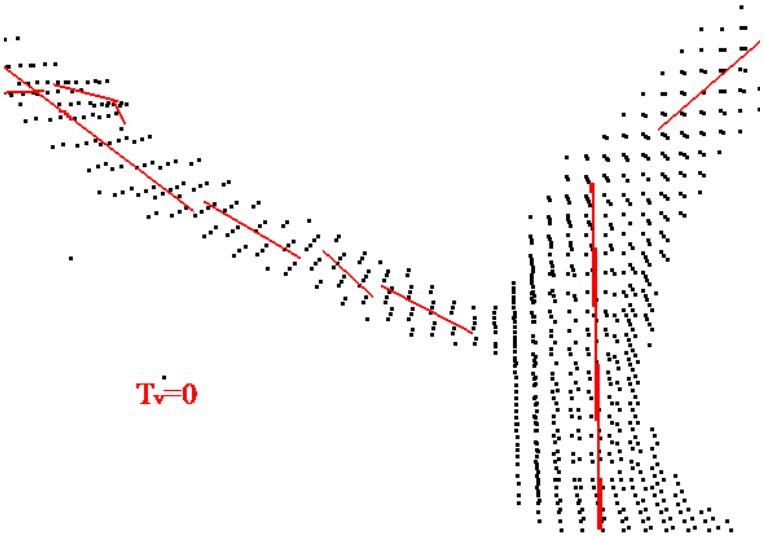

Figure 6.6: Fitting result without label cost, the smoothness cost is set to 30.

to reduce the number of models. We can get similar results by tuning the parameter of the terms. In Figure 6.6, we increased the smoothness cost to 30 and set label cost to be 0. The fitting result is somewhat similar to the one shown in Figure 6.5, but not as good. In addition, experiments show that it is easier to find a good parameter range if we use both Potts model and label cost.

We do not do experiment with piece-wise model and labelcost.The piece-wise model can result in good fitting results even without label cost. This will be illustrated in the following sections.

6.3 Smoothness Term Experiments

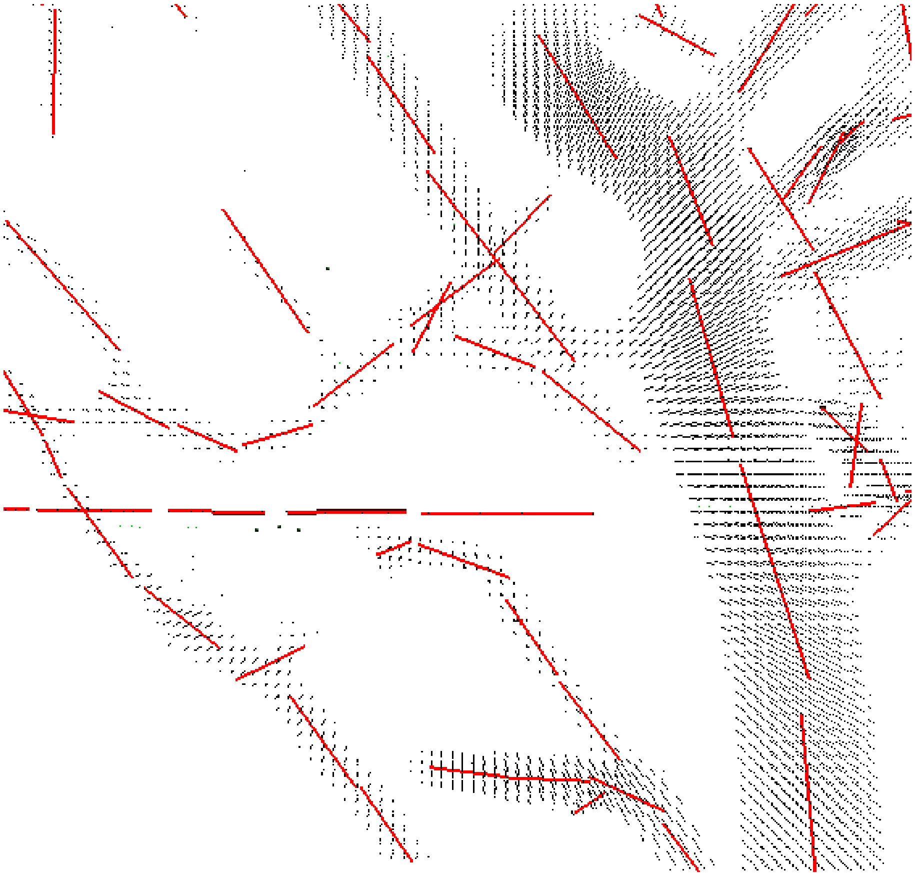

Figure 6.7: A snapshot of fitting result generated by Potts model, with constant label cost and line segment model data cost.

We now evaluate different smoothness terms for vessel fitting. Figure 6.7 is the fitting result generated by Potts model on a cube (111x44x111). Figure 6.8 is the result generated by the piece-wise smooth model. The result generated by piece-wise smooth model is better than Potts model. This is because the Potts model does not care about the similarity between the line segments fitted to nearby pixels. On the contrary, the piece-wise smooth model integrates both the angle and spatial information in a principled way, nearby line segments are encouraged to have a similar orientation and spatial coordinates.

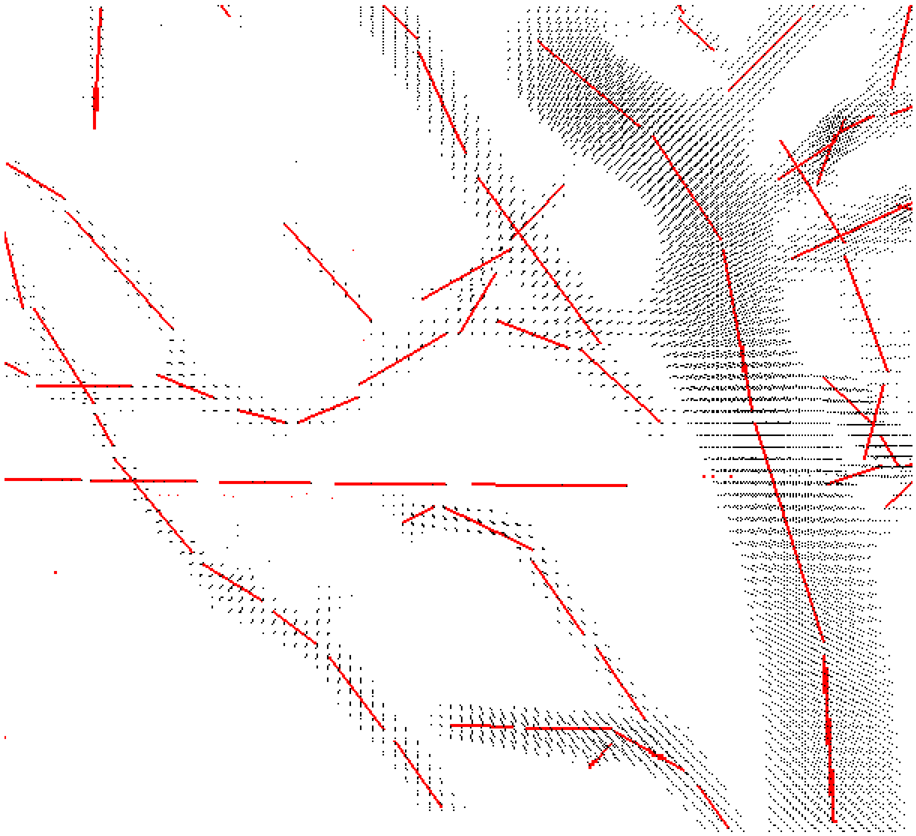

Figure 6.8: A snapshot of fitting result generated by piece-wise smooth model. There is no label cost, the data cost is based on the line segment model.

6.4 Comparison of Different Parameter Re-Estimation Approaches

We now compare the fitting results between different approaches to re-estimating the parameters of the line segment models. Figure 6.9 is the result generated by the exhaustive grid search approach. The smoothness cost is based on the piece-wise smooth model.

There are two steps in the fast local search approach, first we need to run a least squares based fast fitting approach, then we use the local search approach on the results generated by the least squares approach. Both approaches use piece-wise smooth model. Figure 6.10 is the result generated by the local search approach, Figure 6.11 is the fitting results of the least squares approach.

There is some difference between the results in Figure 6.9 and 6.10, but this difference is insignificant. This illustrates the effectiveness of our fast local search approach. Table 6.1 shows the running times of these two approaches. The processors of the our experiment PC

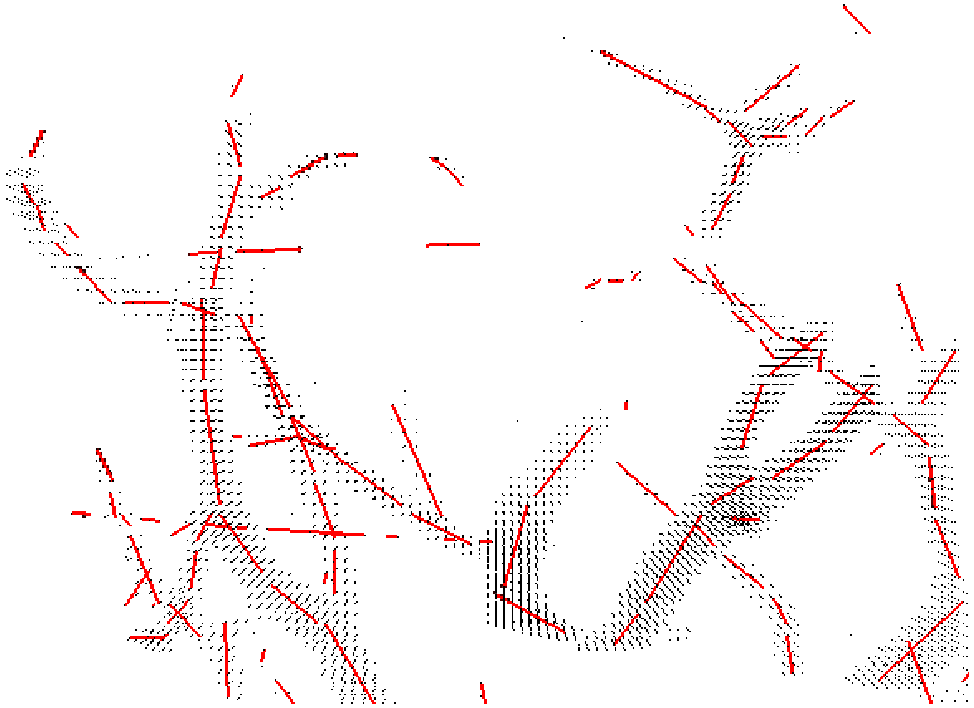

Figure 6.9: A snapshot of fitting result generated by exhaustive grid search approach.

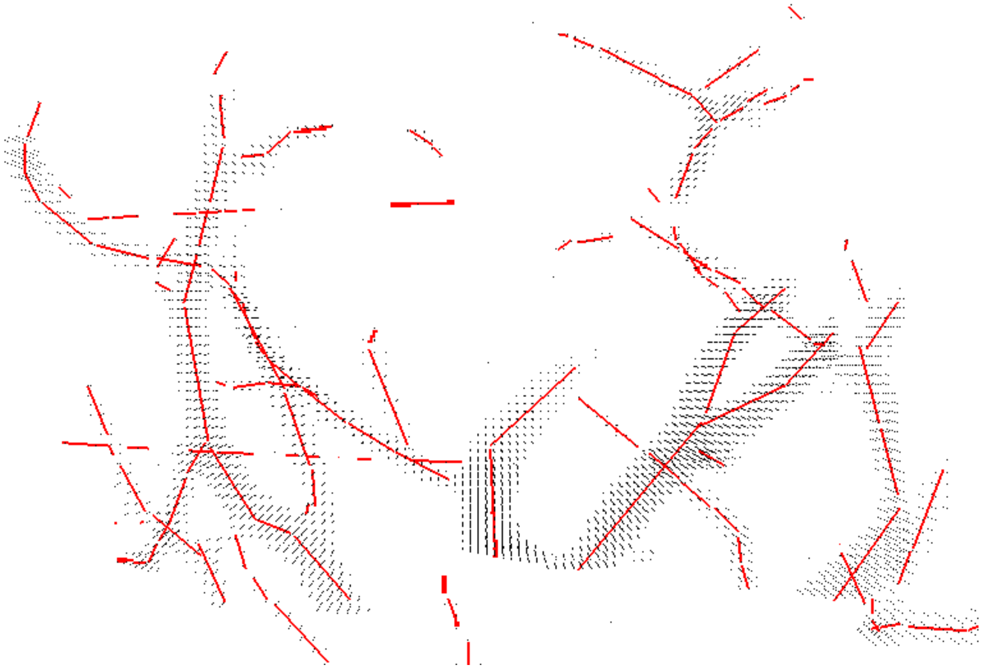

Figure 6.10: A snapshot of fitting result generated by local search approach. The input of this approach is the results generated by least squares based approach.

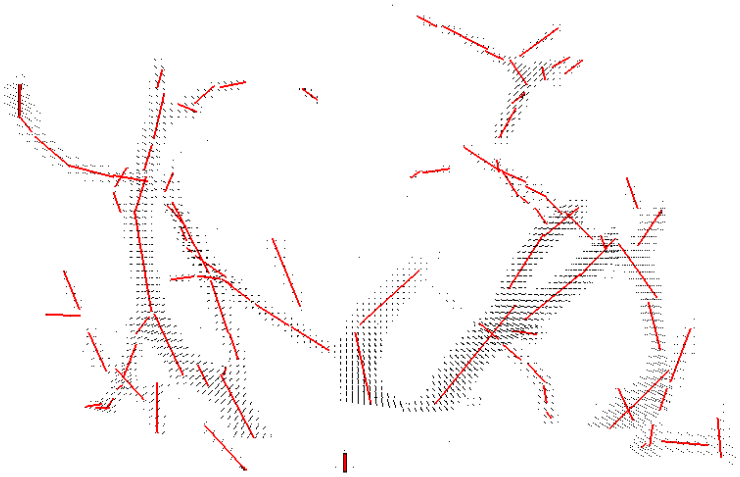

Figure 6.11: A snapshot of fitting result generated by least squares based search approach.

are Intel(R) Core(TM)i5-2400 CPU @3.10GHZ(4 CPUs); the memory size is 4096 MB. We have not use any parallel computational technology in our current experiments. In the future, we will try to run our program on better machines, and use some parallel technologies such as multi-thread and CUDA.

Size	Fast Local Search		Greedy Grid Search
	Least Squares	Local Search	
88x67x70	3min	50min	6 hours
111x44x111	13min	216min	30 hours
151x84x151	22min	382min	>50 hours

Table 6.1: Running times of these two approaches.

6.5 Estimation of σ

σ is one parameter of the line segment model, which can be called as 'line width'. The 'line width' is directly proportional to the radius of the fitting vessel. Figure 6.12 shows the an example of σ estimation. In most areas the estimation of σ is reasonbale, but incorrect estimations still exist in some areas. For example, the value of σ in the yellow circle is much bigger than it should be. The reason is this model fits both the vessel voxels and noise points, which increases the width of the segment incorrectly.

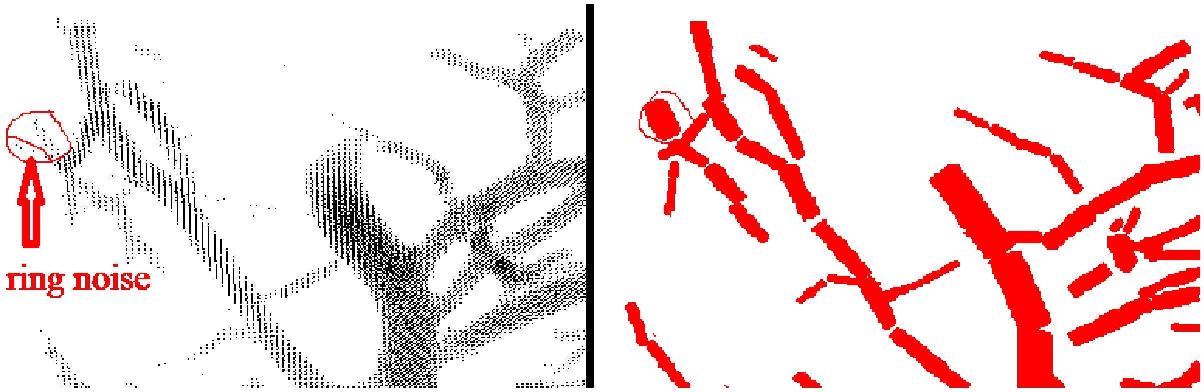

Figure 6.12: The left image is the input vesselness measure, the right image is the corresponding σ estimation. σ is directly proportional but not equal to the radius of the vessel.

6.6 Experiment of the Energy

We run both $\alpha-expansion$ and $\alpha-\beta$ swap for 10 times, and record the converged energy values. Figure 6.13 is the graph of the converged energy values of $\alpha - expansion$. The average value is 54160.9, the standard deviation is 489.5. Figure 6.14 shows the converged energy values of $\alpha - expansion$. The average value is 17336.9, the standard deviation is 141.7. It is clear that both $\alpha - expansion$ and $\alpha - \beta$ swap can converge to a certain energy value, with a small standard deviation. The average energies of $\alpha - expansion$ and $\alpha - \beta$ swap are different. This is because $\alpha - expansion$ optimizes the Potts model, while $\alpha - \beta swap$ optimizes the piece-wise smooth model.

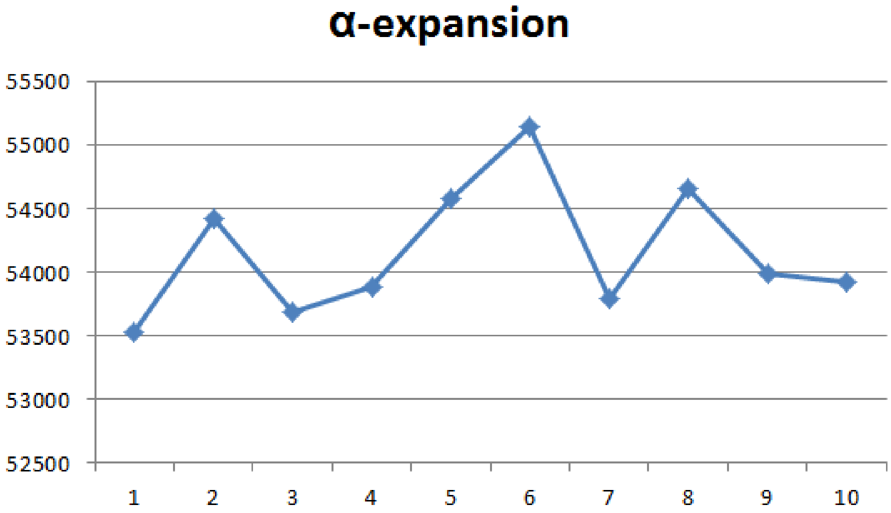

Figure 6.13: The converged energy values of $\alpha - expansion$.

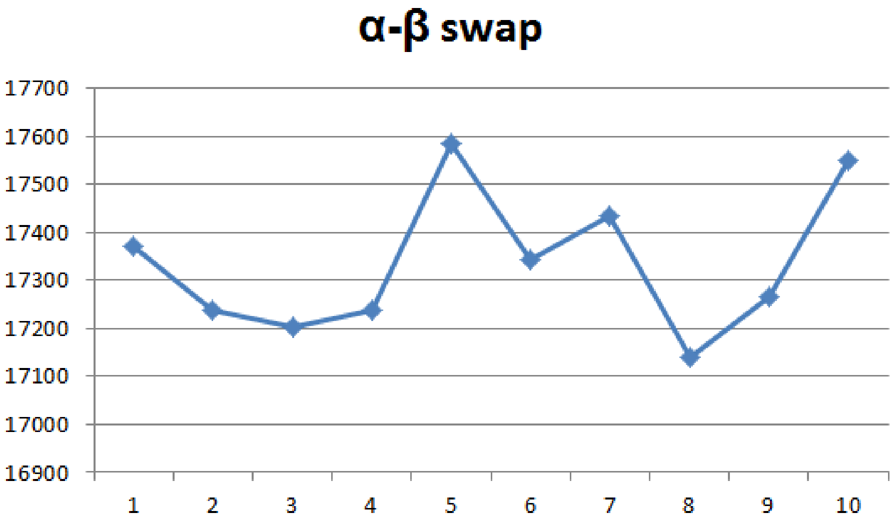

Figure 6.14: The converged energy values of $\alpha - \beta swap$.

6.7 Remaining Challenges

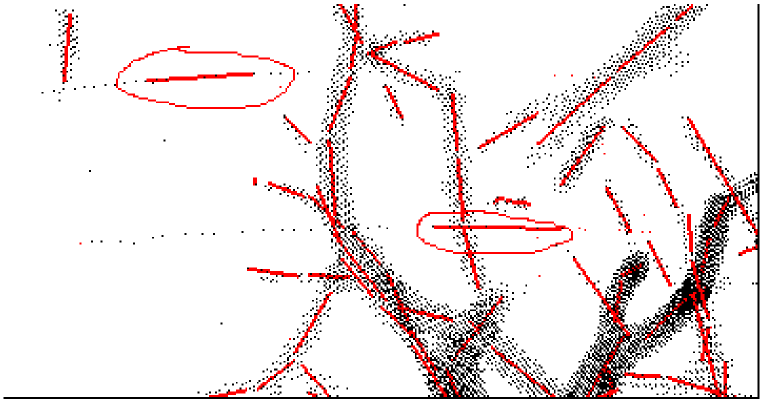

Figure 6.15: A snapshot of fitting result with rings (yellow circle).

We run the fast local search approach on more data cubes. Experiment results show that although this piece-wise smooth line segment model can generate reasonable fitting results, there are still some problems. One problem is the ring noise. There is ring noise in the original input data, which is caused by CT scanner. Before computing the vesselness, we have pre-processed the input data with some ring removal algorithms, but there are still some rings left. Therefore we fit line segments to the rings that have high enough vesselness measure, as shown in Figure 6.15.

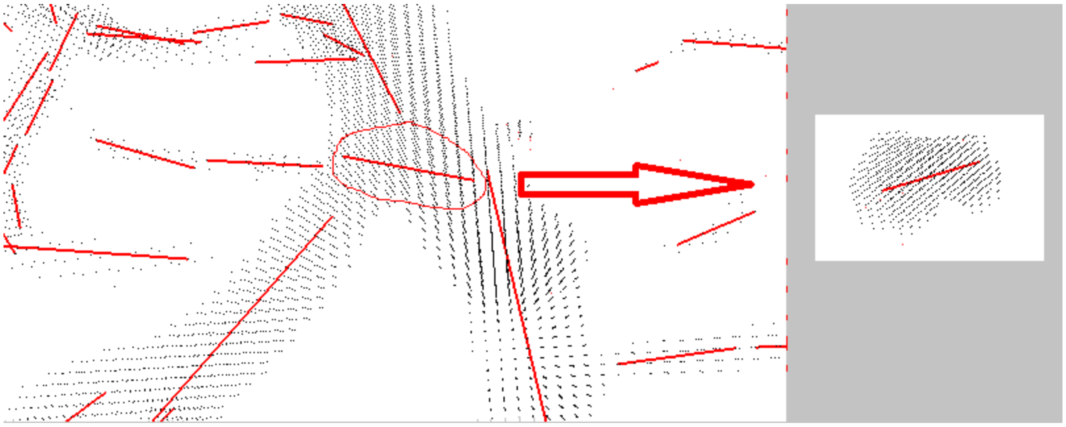

Figure 6.16: A bad case of fitting around a bifurcation. As the ambiguous area among the three vessels are not divided properly, an extra line segment is used to fit vessel voxels within the yellow circle.

Another problem is fitting around bifurcations. There are usually three vessels on a bifurcation, two vessels meet and merge into one bigger vessel. It is difficult to divide the ambiguous area among these three vessels properly. Figure 6.16 shows fitting failure around a bifurcation.

Chapter 7

Conclusion and Future Work

In this thesis, we formulate blood vessel automatic extraction as a geometric multi-model fitting problem, which can be optimized via graph cuts algorithm. Our input data are micro-CT images of a mouse heart. We first extract the points that are likely to be blood vessels by estimating the vesselness measures. Then we propose a piecewise line-segment model to fit the thresholded vesselness measure voxels.

7.1 Summary

Before performing multi-model fitting, we first sample a set of line segment models on the vessel voxels by a density based random sampling strategy with angle agreement. This sampling strategy can sample more segments on thin vessels and less on thick vessels, and control the direction of the segments. In addition to these regular models, we also add an outlier model to model outlier points. Experimental results illustrate the importance of the outlier model, especially when fitting noise voxels.

After sampling, we build a global energy function based on the line segment model. Our energy function consists of data cost, smoothness cost and label cost. The data cost measures the probability that a vessel voxel belongs to a segment. We test two different data cost, one is computed as an infinite mixture of isotropic gaussian distribution on line segment; the other measures the probability that points belong to an infinite line model L. The line model is much simpler than the line segment model, however, the latter can generate better results. We compare two different smoothness terms: the Potts model and the piecewise smooth model. Both of the models encourage the labeling to be consistent, but the piecewise smooth model can also regularize the relative positions between neighboring segment pairs. The label cost penalizes overly-complex models, that is label cost prefers to explain the data with fewer, cheaper models. But label cost alone does not generate good results, it should be used together with the Potts model.

We minimize the global energy function via graph cut algorithm. If the smoothness term is a metric then we use $\alpha - expansion$ to optimize the global function, and if it is a semi-metric we use $\alpha - \beta$ swap. The output of graph cut is the optimal labeling result, we re-estimate the line segments from these labeling results by re-fitting the optimal labeling with line segments of best fit. In our work, we test two different approaches: the exhaustive grid search and fast local search. The exhaustive grid search approach can generate approximate global minima reliably but is computational expensive. The fast local search gives a locally optimal solution in a relatively short time. Experimental results show that these solutions give a reasonable fit in practice.

After parameter re-estimation, we update the set of labels, that is the set of line segments with the re-fitted models, and remove any line segment models that failed to get any support. Then minimize the global energy function again, using the updated set of labels. This process is iterated until the convergence of the energy.

7.2 Limitations

There are some limitations of our algorithms. First is the running speed. The fast local search approach can generate results on small data cubes in a relatively short time, as shown in table 6.1. But it is still too slow for the whole dataset. Besides, we have to tune new parameters for new data set, which greatly increases the running time on the big datasets. The slow running speed is caused by our over-complex line segment model. Besides, we can not validate our fitting results as there is no ground truth. We have to manually check the fitting results.

7.3 Future Work

Although the approach presented in this work can generate reasonable fitting results, there are still some issues to investigate, which are listed as follows:

- It is difficult to fit well around bifurcations. There are usually three vessels around a bifurcation. It is quite a challenging task to accurately fit line segments at the bifurcation areas, as the ambiguous area has to be divided among the three vessels. If the area is not divided properly, then the fitting result is not satisfying.

- We should either find a faster parameter re-estimation method, or propose a simpler model to replace the line segment model. As the parameter re-estimation of such complex model takes too much time to fitting big datasets. Parallel computation is another choice.

- Although we have pre-processed the input data with some ring noise removal algorithms, there are still some ring artifacts left. So we need to improve the ring removal algorithms.
- We do not deal with the partial voluming effects in our present work. This problem should also be solved in the future work.

Bibliography

[1] http://www.flowtech-inc.com/products.htm/.

[2] GA Ascoli, K Svoboda, and Y Liu. Digital reconstruction of axonal and dendritic morphology diadem challenge, 2010.

[3] Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 489–495. IEEE, 1999.

[4] Yuri Boykov and Marie-Pierre Jolly. Interactive organ segmentation using graph cuts. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2000*, pages 276–286. Springer, 2000.

[5] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 26–33. IEEE, 2003.

[6] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *Computer vision and pattern recognition, 1998. Proceedings. 1998 IEEE computer society conference on*, pages 648–655. IEEE, 1998.

[7] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[8] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[9] Paarth Chothani, Vivek Mehta, and Armen Stepanyants. Automated tracing of neurites from light microscopy stacks of images. *Neuroinformatics*, 9(2-3):263–278, 2011.

[10] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96(1):1–27, 2012.

[11] Andrew T Delong. Advances in graph-cut optimization: Multi-surface models, label costs, and hierarchical costs. 2011.

[12] Luke Domanski, Changming Sun, Raquibul Hassan, Pascal Vallotton, and Dadong Wang. Linear feature detection on gpus. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 649–656. IEEE, 2010.

[13] Duncan E Donohue and Giorgio A Ascoli. Automated reconstruction of neuronal morphology: an overview. *Brain research reviews*, 67(1):94–102, 2011.

[14] Martin A Fischler, Jay M Tenenbaum, and Hans Christoph Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. *Computer Graphics and Image Processing*, 15(3):201–223, 1981.

[15] Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. In *Medical Image Computing and Computer-Assisted InterventationłMICCAI98*, pages 130–137. Springer, 1998.

[16] Daniel Freedman and Tao Zhang. Interactive graph cut based segmentation with shape priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 755–762. IEEE, 2005.

[17] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[18] DM Greig, BT Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

[19] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012.

[20] Hiroshi Ishikawa and Davi Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *Computer VisionłECCV'98*, pages 232–248. Springer, 1998.

[21] Hiroshi Ishikawa and Davi Geiger. Segmentation by grouping junctions. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 125–131. IEEE, 1998.

[22] Junhwan Kim, Vladimir Kolmogorov, and Ramin Zabih. Visual correspondence using energy minimization and mutual information. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1033–1040. IEEE, 2003.

[23] Junmo Kim, John W Fisher III, Andy Tsai, Cindy Wible, Alan S Willsky, and William M Wells III. Incorporating spatial priors into an information theoretic approach for fmri data analysis. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2000*, pages 62–71. Springer, 2000.

[24] Cemil Kirbas and Francis KH Quek. Vessel extraction techniques and algorithms: A survey. In *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*, pages 238–245. IEEE, 2003.

[25] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 277–286. ACM, 2003.

[26] Max WK Law and Albert CS Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In *Computer Vision–ECCV 2008*, pages 368–382. Springer, 2008.

[27] Max WK Law and Albert CS Chung. An oriented flux symmetry based active contour model for three dimensional vessel segmentation. In *Computer Vision–ECCV 2010*, pages 720–734. Springer, 2010.

[28] Charles E Leiserson, Ronald L Rivest, Clifford Stein, and Thomas H Cormen. *Introduction to algorithms*. The MIT press, 2001.

[29] Hua Li and Anthony Yezzi. Vessels as 4-d curves: Global minimal 4-d paths to extract 3-d tubular surfaces and centerlines. *Medical Imaging, IEEE Transactions on*, 26(9):1213–1223, 2007.

[30] Michael H Lin and Carlo Tomasi. Surfaces with occlusions from layered stereo. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–710. IEEE, 2003.

[31] Yu Liu. Classic mosaics and visual correspondence via graph-cut based energy optimization. 2011.

[32] Ju Lu. Neuronal tracing for connectomic studies. *Neuroinformatics*, 9(2-3):159–166, 2011.

[33] Paul Martz. Openscenegraph quick start guide. *Louis-ville: Skew Matrix Software*, 200:20, 2007.

[34] Erik Meijering. Neuron tracing in perspective. *Cytometry Part A*, 77(7):693–704, 2010.

[35] Carl Olsson and Yuri Boykov. Curvature-based regularization for surface approximation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1576–1583. IEEE, 2012.

[36] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998.

[37] Hanchuan Peng, Fuhui Long, and Gene Myers. Automatic 3d neuron tracing using all-path pruning. *Bioinformatics*, 27(13):i239–i247, 2011.

[38] Sébastien Roy. Stereo without epipolar lines: A maximum-flow formulation. *International Journal of Computer Vision*, 34(2-3):147–161, 1999.

[39] Sébastien Roy and Ingemar J Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Computer Vision, 1998. Sixth International Conference on*, pages 492–499. IEEE, 1998.

[40] Yoshinobu Sato, Shin Nakajima, Hideki Atsumi, Thomas Koller, Guido Gerig, Shigeyuki Yoshida, and Ron Kikinis. 3d multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. In *CVRMed-MRCAS'97*, pages 213–222. Springer, 1997.

[41] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 345–352. IEEE, 2000.

[42] Philip HS Torr. Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998.

[43] E Turetken, Fethallah Benmansour, and Pascal Fua. Automated reconstruction of tree structures using path classifiers and mixed integer programming. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 566–573. IEEE, 2012.

[44] Engin Türetken, Fethallah Benmansour, Bjoern Andres, Hanspeter Pfister, and Pascal Fua. Reconstructing loopy curvilinear structures using integer programming. 2013.

[45] Engin Türetken, Germán González, Christian Blum, and Pascal Fua. Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, 9(2-3):279–302, 2011.

[46] Olga Veksler. *Efficient graph-based energy minimization methods in computer vision*. PhD thesis, Cornell University, 1999.

[47] Etienne Vincent and Robert Laganiére. Detecting planar homographies in an image pair. In *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pages 182–187. IEEE, 2001.

[48] Jerold W Wallis, Tom R Miller, Charles A Lerner, and Eric C Kleerup. Three-dimensional display in nuclear medicine. *Medical Imaging, IEEE Transactions on*, 8(4):297–230, 1989.

[49] Rui Wang and Xuelei Qian. *OpenSceneGraph 3.0: Beginner's Guide*. PACKT publishing, 2010.

[50] Yu Wang, Arunachalam Narayanaswamy, Chia-Ling Tsai, and Badrinath Roysam. A broadly applicable 3-d neuron tracing method based on open-curve snake. *Neuroinformatics*, 9(2-3):193–217, 2011.

[51] Daniel Weinland, Mustafa Özuysal, and Pascal Fua. Making action recognition robust to occlusions and viewpoint changes. In *Computer Vision–ECCV 2010*, pages 635–648. Springer, 2010.

[52] Richard Wright, Nicholas Haemel, Graham M Sellers, and Benjamin Lipchak. *OpenGL SuperBible: comprehensive tutorial and reference*. Pearson Education, 2010.

[53] Ting Zhao, Jun Xie, Fernando Amat, Nathan Clack, Parvez Ahammad, Hanchuan Peng, Fuhui Long, and Eugene Myers. Automated reconstruction of neuronal morphology based on local geometrical and global structural models. *Neuroinformatics*, 9(2-3):247–261, 2011.

[54] Marco Zuliani, Charles S Kenney, and BS Manjunath. The multiransac algorithm and its application to detect planar homographies. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–153. IEEE, 2005.

Curriculum Vitae

Name:	Xuefeng Chang
Education	University of Western Ontario London,Ontario M.Sc. (Computer Science), expected, January 2014
	Beihang University Beijing, China M.Sc. (Computer Science), received, January 2011
	Beijing University of Chemical Technology Beijing, China B.S. (Computer Science), received, June 2008
Related Work Experience:	Research Assistant Department of Computer Science The University of Western Ontario, London, Ontario September 2012 - December 2013
	Teaching Assistant Department of Computer Science, The University of Western Ontario, London, Ontario September 2012 - April 2013, September 2013 - December 2013