Certification

Application of Image Segmentation and Adaptive Interpolation Techniques to 3D Reconstruction of the Human Temporal Bones

by

Zhenfeng Zhao

A Thesis Submitted to Saint Mary's University, Halifax, Nova Scotia,
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Applied Science

December 2011, Halifax, Nova Scotia

Approved:	Dr. Norma Linney Supervisor Mathematics and Computing Science Dept.
Approved:	Dr. Matthias Schmidt External Examiner Department of Radiology, Dalhousie University
Approved:	Dr. Manohar Bance Supervisory Committee Member Division of Otolaryngology, DAL
Approved:	Dr. Pawan Lingras Supervisory Committee Member Mathematics and Computing Science Dept.
Approved:	Dr. Paul Muir Supervisory Committee Member Mathematics and Computing Science Dept.
Approved:	Dr. Stavros Konstantinidis Program Representative
Approved:	Dr. Kevin Vessey Dean of Graduate Studies
Date:	December 6, 2011

Your file Votre référence
ISBN: 978-0-494-82903-5
Our file Notre référence
ISBN: 978-0-494-82903-5

Abstract

Application of Image Segmentation and Adaptive Interpolation Techniques to 3D Reconstruction of the Human Temporal Bones

By Zhenfeng Zhao

Three dimensional models aid otolaryngologists in understanding the complex anatomical features of the human temporal bone. Many of these models are generated by reconstructing histological sections. The goal of this thesis is to provide improvements on these existing 3D reconstruction methods. Presented are a segmentation framework and a contour finding algorithm for histological slices, followed by Gaussian filtering and error analysis. An adaptive interpolation algorithm based on monotonic piecewise cubics is used to automatically generate missing anatomical structure. Part of the algorithm development was completed on CT scans with proposals for extension to histological slices. The contour finding and Gaussian filtering algorithms output valid data points for interpolation. The adaptive interpolation algorithm produces satisfactory results with the interpolation error for the malleus being 1.80% when half of the data is used. The equivalent 3D model volume difference was 0.24%.

December 6, 2011

ACKNOWLEDGEMENTS

I would like to express my appreciation to Dr. Norma Linney, my supervisor for my graduate studies. She gave me a great amount of instructions in research techniques, analysis methods and guides through the thesis work. She provided helps for me to solve challenging computer vision and math problems.

I want to thank Dr. Paul Muir and Dr. Pawan Lingras who are my supervisor committee members. They helped me in areas of interpolation and artificial intelligence, and comments to my thesis.

I also want to thank Dr. Manohar Bance and Rene van Wijhe at the Ear & Auditory Research Laboratory (EAR-Lab) of Dalhousie University. They provided image datasets to be applied in testing, as well as background knowledge in the medical field.

I would like to thank Graduate Studies and Research and NSERC for their financial support.

The special thanks go to my families. Thanks to my wife, Cailan, for being with me. You gave me unconditional support through this long process. Thanks to my mother. You supported me studying in Canada and showed me the way to the future.

Contents

List of Tables

List of Figures

List of Symbols

α: The percentage of the signal power to keep, while the rest needs to be filtered

β: A fraction that is multiplied with data error rate; the result is used to measure interpolation accuracy

σ: Standard deviation of the Gaussian function curve

σt: Standard deviation of the three-σ rule

σ_f: Standard deviation (σ) in the frequency domain

σ_s: Standard deviation (σ) in the spatial domain

θ: The angle of the contour point to the region center

A: The gain factor in GLPF function

B_{width}: Bandwidth; the width between lower and upper cutoff frequencies

$D(u)$: Distance between u and the origin of the centered filter function

D_0: Specified distance (nonnegative variable) to the origin; the distance equals to the cutoff frequency

$Diff_{interpolant}(z)$: The function of absolute difference between the last two interpolants by evaluating them at z locations

E: Error

E_{approx}: The approximate data error

$E_{contourfiltering}$: Error of contour finding and filtering measured using the $1-S$ method

$E_{contourfind}$: Error of contour finding

$E_{filtering1}$: Error of Gaussian filtering measured using the $1-S$ method

$E_{filtering2}$: Error of Gaussian filtering measured using the $\triangle r$ method

$E_{filteringS}$: Error of Gaussian filtering in the on slice filtering step measured using the $\triangle r$ method

$E_{filteringZ}$: Error of Gaussian filtering in the z-direction filtering step measured using the $\triangle r$ method

$E_{interpolate}$: Error of interpolation measured using the $1 - S$ method

$E_{interpolant,avg}$ The average of $E_{interpolant}(z)$ in all z locations

$E_{interpolant}(z)$: The function of the percentage difference between the last interpolant and the filtered data by evaluating it at z locations

$E_{overall}$: The overall error

$E_{segment}$: Average error of segmentation

$E_{segment1}$: Error of segmentation; comparison of the expert's manual segmentation results and the first manual segmentation done by the thesis author

$E_{segment2}$: Error of segmentation; comparison of the expert's manual segmentation results and the second manual segmentation done by the thesis author

$E_{segment3}$: Error of segmentation; comparison of the expert's manual segmentation results and the manual segmentation done by a third-party tester

E_m: Error of segmentation on one image at location z ($z = z_0 + m\triangle z$; index of m)

$E_{m,contourfind}$: Error of contour finding on one image at the location index of m

$E_{m,contourfiltering}$: Error of contour finding and filtering measured using the $1 - S$ method at the location index of m

$E_{m,filtering1}$: Error of Gaussian filtering measured using the $1 - S$ method on one

image at the location index of m

$E_{m,interpolate}$: Error of interpolation measured using the $1-S$ method at the location index of m

E_{true}: The true data error

$f(x)$: A function of x

$f(x,y)$: An image function

$f_s(n)$: On slice signal, representing a list of contour points in one slice

$f_z(m)$: z-direction signal, representing a list of contour points in z-direction with the same angle.

$F(u)$: Fourier transform of the signal

$F(u,v)$: The Fourier transforms of the image

$\mathbb{F}$: Fourier transform operation

$\mathbb{FFT}$: Fast Fourier transform operation

$\mathcal{F}_{cutoff}$: Cutoff frequency

$\mathcal{F}_s$: Sampling rate; the number of samples taken within a unit distance, when taking samples from a continuous signal

$g(x)$: Output signal in the spatial domain

$g(x,y)$: Output image

$G(u)$: Fourier transform of the output signal (in the frequency domain)

$G(u,v)$: Fourier transform of the output image

$h(x)$: Filter function in the spatial domain

$h(x,y)$: 2D Filter function

$h_{Gaussian}(x)$: Gaussian filter in the spatial domain

$H_{Gaussian}(u)$: Gaussian filter in the frequency domain

$H(u)$: Fourier transform of the filter

$H(u, v)$: The frequency domain filtering function

i: Generic index

$I(u)$: Imaginary part of $F(u)$

$Interpolant_{last}(z)$: The last interpolant function

$Interpolant_{secondlast}(z)$: The second last interpolant function

$Interpolant_i(z)$: The interpolant computed in level i of the adaptive interpolation algorithm

k_i: The index of the current interval

K_i: The number of intervals in M points

L: Total x-axis length of the signal function $f(x)$

m: The index of contour points in z-direction; index of slice

M: The number of contour points in z-direction (after contour finding); the total slice number

MI: Mutual information

n: Index of contour points on slice

N: Number of contour points on each slice (after contour finding)

NMI: Normalized mutual information

$P(u)$: Power function of a signal

$P_s(u)$: Power function of a signal in the on-slice direction

$P_{s,avg}(u)$: Averaged power curve in the on-slice direction

$P_{z,avg}(u)$: Averaged power curve in the z direction

P_T: Total signal power

$P_{s,T}$: Total signal power in the on-slice direction

Q_{max}: Number of points in the image contour (after contour finding) that has the largest number of points

r: The radius from the contour point to the region center

$r\prime$: Radius of a contour point after filtering on slice

$r\prime\prime$: Radius of a contour point after filtering in z-direction

$r_z(\theta)$: The radius function on slice with fixed z

$r_\theta(z)$: The radius function in the z direction with fixed θ

$r(z,\theta)$: The 2D radius function in the z direction and on slice

$r_{filtered}(z)$: The discrete r function of filtered data

$R(u)$: Real part of $F(u)$

$\triangle r$: The difference in two radii

ROI: Region of interest

$ROI_{contourFind}$: The result image region of contour finding

$ROI_{filtering}$: The result image region of filtering

$ROI_{interpolate}$: The result image region of interpolation

$ROI_{segment}$: The result image region of segmentation

ROI_m: The image region at the location index of m

$ROI_{m,contourFind}$: The result image region of contour finding at the location index of

m

$ROI_{m,filtering}$: The result image region of filtering at the location index of m

$ROI_{m,interpolate}$: The result image region of interpolation at the location index of m

$ROI_{m,segment}$: The result image region of segmentation at the location index of m

S: Similarity index

S_m: Similarity index at the location index of m

t: Index of a particular contour point in the current interval

T: The number of points in the current interval

u: Frequency variable

$\triangle u$: An increment, or unit, of a fixed interval in the frequency domain

$V(x)$: The piecewise function

$v_i(x)$: A third degree polynomial function in a subinterval of a piecewise function

W_s: The filter width of a spatial GLPF in the on-slice direction

W_z: The filter width of a spatial GLPF in the z direction

x: Horizontal coordinate

(x, y): The coordinate of a point or a pixel in an image

(x_c, y_c): The center coordinate of all contour points in one slice

X: The number of data points in an interval

X_1: Starting value of intervals (1st level)

X_i: The number of points in one interval in ith level

$\triangle x$: An increment, or unit, of a fixed interval in the spatial domain

y: Vertical coordinate

z: The z coordinate of the contour point

Acronyms

DFT: Discrete Fourier transform

DLL: Dynamic-link library

FFT: Fast Fourier transform algorithm

GLPF: Gaussian low-pass filter

ILPF: Ideal low-pass filter

PCHIP: Piecewise Cubic Hermite Interpolation Package

SPD: Spectral power distribution

Chapter 1

Introduction

There are a large number of complex anatomical features within the small space occupied by the middle and inner ear in the human temporal bone. Otolaryngologists desire to study and better understand the complex structure to improve diagnostic tools and hearing-aid devices. Three dimensional (3D) models are used for studying the inner ear structure and realizing medical conditions in the area. Our collaborators, medical researchers from the EAR-Lab [17] at Dalhousie University, desired improved 3D models to aid in the research work that creates better diagnostic tools. The existing models are built from histological sections (see below) using a commercial software package for visualizing and manipulating bio-medical data in 3D, called Amira (see Section 2.6).

Histological sections (slices of the human temporal bone) are created post-mortem from tissue samples that are prepared as slides and then digitized. The process is very time consuming and filled with inaccuracies. The 3D model is built from these slices. Researchers desire a more accurate and robust 3D model. The thesis investigates possibilities for improving the process of building a 3D model from histological sections of the human temporal bone.

Since histological sections are created post-mortem and the process is fraught with inaccuracies, there is no gold standard for the creation of the model. In other words, there is no way to tell how closely the model represents the original anatomy.

To aid in our investigation, the EAR-Lab [17] provided us with a second dataset. This dataset included a complete computerized tomography (CT) scan of the two temporal bones of this study, the incus and the malleus. The CT scan was non-destructive and consisted of a set of cross sectional images of the incus and the malleus. This dataset was investigated to develop algorithms with the goal to applying them to the histological section data. As explained in detail in the experimental data section (refer to Section 4.1), the CT scans were taken from a more complete dataset, which can be used as test data. These test results can assist research based on histological sections.

The techniques investigated included registration, the process of aligning digital images so that same structures appearing in multiple images correspond, and segmentation, the process of isolating the structure of interest in a digital image. We also considered the techniques of filtering or removing noise from the images, and interpolation, a method of constructing missing data points using a set of given data points. All of these techniques were investigated using the CT scan data with the goal of extending the methods to histological sections.

Our research primarily explored segmentation and interpolation techniques suitable for histological sections. Several segmentation algorithms were studied and implemented, e.g., thresholding, k-means clustering, region growing, and Canny edge detection. The region growing algorithm was modified and improved to fit the characteristics of histological sections; we call the resulting algorithms two-threshold region growing and patch growing algorithms. We propose a segmentation framework that uses semi-automatic selection of the region of interest (ROI), and performs median filtering, patch growing, and Canny edge detection automatically. The segmentation framework produces both region and edge images. Then, these images are processed by the contour finding algorithm. After that, the final segmentation results are obtained.

The other focus of our research was interpolation. Relevant questions are which interpolation technique is suitable for our application, and how should interpolation be applied to produce accurate results. The interpolation algorithm should achieve high accuracy while optimizing data usage (i.e., using relatively small amounts of the total amount of available data).

After studying and experimenting with linear splines, cubic splines and monotonic piecewise cubic splines, we concluded that monotonic piecewise cubics would be the most suitable for our application. Interpolation is explained in Section 2.5.

An adaptive interpolation algorithm was designed to obtain better approximations. Because data points being interpolated need to be smoothed, with outliers removed, an automatic filtering algorithm was designed, using a filter based on a Gaussian function. After that, error analysis of segmentation, contour finding, and filtering algorithms were done for the CT test data. The resulting error data was used as basis for the adaptive interpolation algorithm. A stand-alone software program was designed to include contour finding, filtering, error analysis and adaptive interpolation algorithms. With several parameters of the program being set, final results from the interpolation process were obtained; these were used to generate 3D models.

In summary, the thesis explores techniques for histological sections and describes the methods applied to the CT data. These methods were developed, with the goal of extending them to the histological sections. This thesis aims to improve existing 3D model generation methods for medical researchers. We present a collection of algorithms to segment structures or regions from registered images, find the contour points (data points representing the contour of the segmented ROIs) from the segmentation results, perform filtering to obtain smooth contour points, analyze errors associated with collecting contour points, and apply an adaptive interpolation algorithm to generate missing data from the sequence of images.

Chapter 2 reviews the literature and relevant techniques in this field, including reviews of the medical background, image registration, geometrical transformations, image segmentation, interpolation, and 3D reconstruction. Chapter 3 describes the theory behind the methods and applications in this thesis. In Chapter 4, we present methods and algorithms that are designed and developed to solve our research problems. The methods consist of: 1) image registration; 2) image segmentation algorithms; 3) obtaining an equal number contour points on every segmented region of interest (ROI); 4) filtering control points using automatically developed GLPFs; 5) analyzing errors in above steps and passing error results to the adaptive interpolation algorithm; 6) adaptive interpolation using monotonic piecewise cubics to generate missing structure; and 7) 3D reconstruction. The result is a complete, smooth and detailed 3D model, as well as data usage information for contour points.

Chapter 5 presents the data, results and discussion for histological sections and CT scans. The results are in two categories. First, for histological sections, sample results of registration, segmentation, and contour finding are shown. Second, results of Gaussian filtering, error analysis, adaptive interpolation, and 3D reconstruction on CT scans are presented. Chapter 6 summarizes the thesis and presents our conclusions. It also relates CT test results to histological sections, and presents future work.

Chapter 2

Background

Our research has applied computer vision techniques to address issues with human temporal bone analysis using serial CT scans and histological sections. This chapter reviews the literature and relevant techniques in these fields, through the following sections: medical background, image registration, geometrical transformations, image segmentation, interpolation, and 3D reconstruction.

2.1 Medical Background

In this section, middle and inner ear anatomy of the human temporal bone will be briefly introduced. It is the bone from which our experimental data was acquired. Surrounded by the temporal bone, the incus and the malleus are the two middle ear bones that were the focus of our experiments and testing. The first type of image data, a series of stained histological section images of human temporal bone, and the method used to obtain these images will be introduced. This will be followed by a brief description of the second type of image data we considered, CT scans.

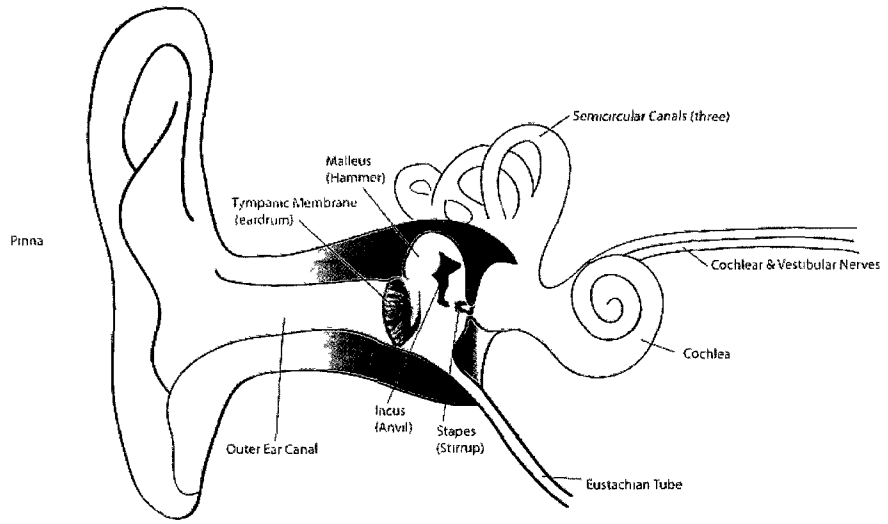

Figure 2.1: A schematic drawing of the human outer, middle and inner ear (EAR-Lab [17]).

2.1.1 Human Temporal Bone

There are a large number of complex structures within the small space of the human temporal bone. Otolaryngologists often find it fairly challenging to understand the three-dimensional (3D) anatomy of the human temporal bone [84]. Figure 2.1 depicts the outer, middle and inner ear structures contained within the temporal bone, while Figure 2.2 zooms in on the middle ear bones and surrounding structures. The incus and the malleus are the two main bones to be considered in our work. They are located close to the tympanic membrane and external ear canal. The length of the incus and the malleus is approximately the radius of the ear canal, so it is less than 10 mm.

It is very useful to have detailed knowledge of the microscopic anatomy of the human temporal bone in order to understand surgical relationships and interpret radiological images of the temporal bone in patients with otologic problems. Analysis using histological sections is useful for identifying the underlying tissue types as well as the structure of each individual tissue component.

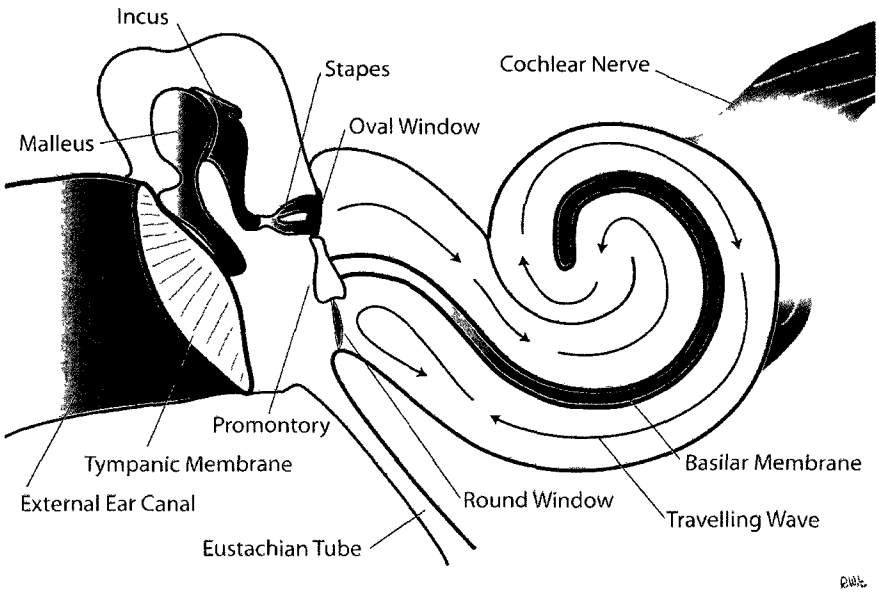

Figure 2.2: A schematic drawing of the middle ear bones and surrounding structures (EAR-Lab [17]).

2.1.2 Incus and Malleus

As mentioned above, in our research, we will focus on two bones in our research, the incus and the malleus, as in Figure 2.2. They are both in the middle ear. The incus is a small bone with an anvil shape. The malleus is hammer shaped. Photos of these bones are given in Figures 2.3 and 2.4.

2.1.3 Histological Sections

Histological section images differ from other types of medical images (refer to Section 2.1.4). Figure 2.6 shows a histological cross-section of the human inner ear. It is an image sample supplied by the EAR-Lab [17]. These images often have relatively severe distortion and displacement issues, making it difficult to perform subsequent computer-based analyses on them.

Histological sections are created post-mortem, by staining and cutting with a

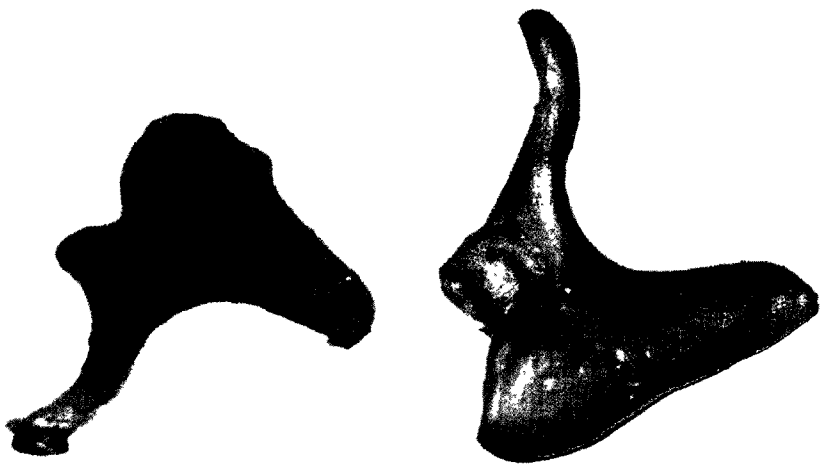

Figure 2.3: Photos of the incus bone in the human inner ear (EAR-Lab [17]).

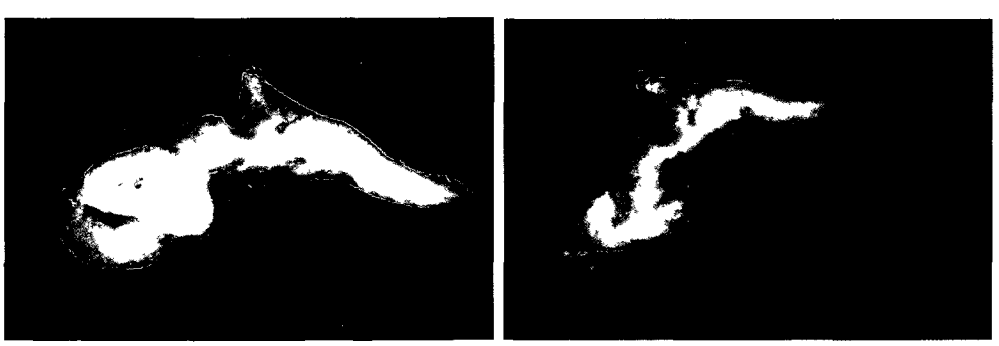

Figure 2.4: Photos of the malleus bone in the human inner ear (EAR-Lab [17]).

microtome, taking sample tissue slices, and digitizing using a microscope. The manual preparation process for histological sections introduces noise, artifacts and issues of distortion and deformation. In our dataset, digital images were created from the slides and our analysis was carried out on the digital images.

Preparation and digitization methods for histological sections of human temporal bones are complex. Wang et al. [84] presented a preparation and digitization method for histological sections of human temporal bones. Preparation of the human temporal bone generally involves fixation, decalcification, and embedding in a supporting medium such as celloidin. This is followed by slicing the sample into sections, or cutting parallel to the long axis of the specimen at a thickness of 20 microns. Initially every tenth section is stained and mounted for microscopic examination. Digitization is then performed by a special microscope.

Auer et al. [3] showed that there are two important elements in this acquisition process. These are the specific microscopes used and the quality of manual interaction. The quality of the digital images depends on several factors, such as background illumination of the microscope, the microtome (the block and knife used to cut the sections), the embedding medium, and the precision of the manual operations. The quality of the images may vary from one image to the next. Artifacts can be introduced during the cutting process. For example, tissue segments can overlap or be missing. Figure 2.5 shows two examples of images of histological sections. The distance between these two tissue slides is 100 microns. However, there are significant differences between the two; obvious distortions and displacements are evident. Color artifacts are introduced from factors such as the microscope background illumination during image acquisition.

Generally, the acquisition process produces artifacts such as holes, varied intensities, blurred regions, and different inter-slice intensities. Some slices are completely corrupted and cannot be used, and this results in non-uniform slice spacing.

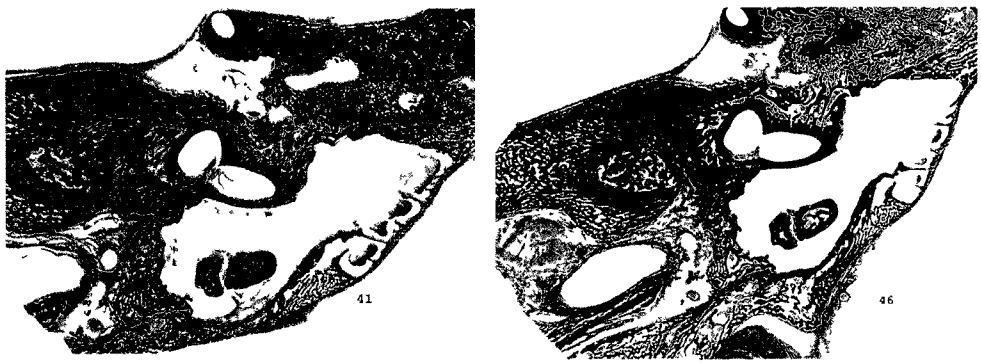

Figure 2.5: Two stained neighboring histological section sample images From left to right (a-b): a. Experimental image data No. 41; b. Experimental image data No 46 (Temporal Bone Foundation, Boston, USA.)

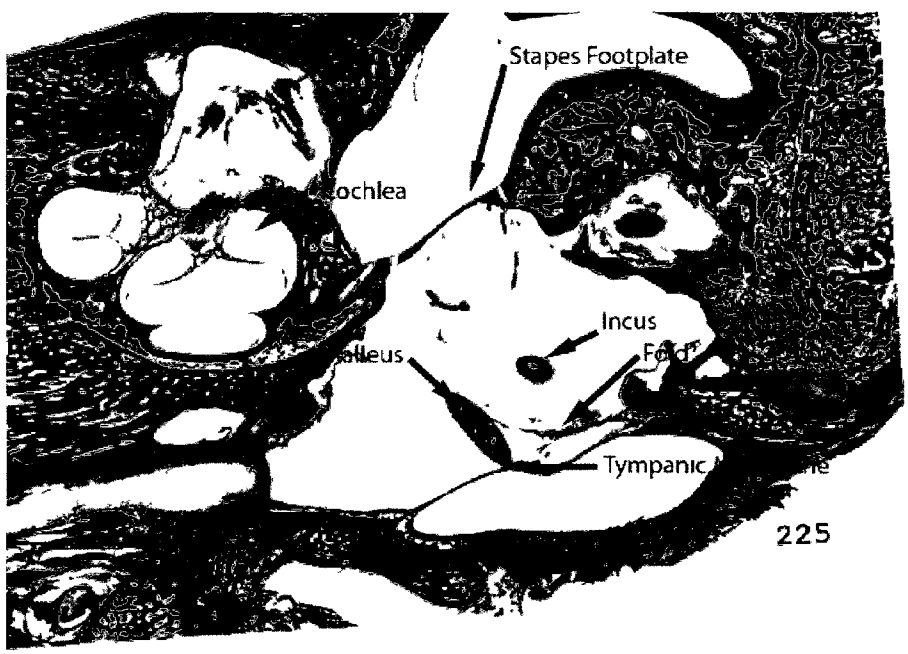

Figure 2.6: A gray scale histological section of the temporal bone region, with six objects labeled. (EAR-Lab [17] and Temporal Bone Foundation, Boston, USA.)

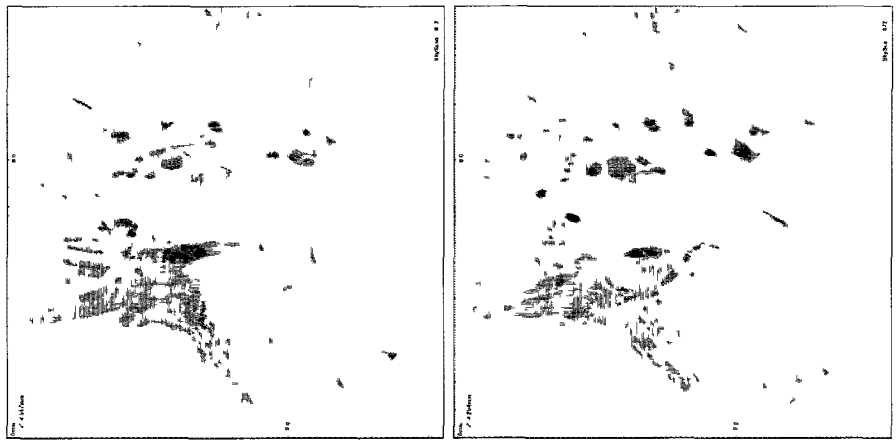

Figure 2.7: Two CT scans samples. From left to right (a-b). a. Experimental image data No. 280; b. Experimental image data No. 300 (Temporal Bone Foundation, Boston, USA.)

2.1.4 CT Scans

CT, computed tomography, is a medical imaging method. Tomography is the process of scanning through a 3D object and generating two-dimensional (2D) cross-section images. The medical device producing CT images is called CT scanner. CT scans are non-destructive. CT scans use tomographic principles to obtain cross sectional images of anatomical structures automatically, in contrast to histological sections, which are very labor intensive. Tomographic principles allow the combination of a series of x-ray views from many different angles to produce a cross-sectional image. A complete CT scan consists of a series of cross-sectional images at different depths which can be combined to give a 3D representation of the 3D object.

Two sample CT scans are shown in Figure 2.7. They are taken from roughly the same locations as the two histological section samples in Figure 2.5. CT scans are a typical type of medical images. Other common types of medical images are MRI (Magnetic Resonance Imaging), PET (Positron Emission Tomography), and ultrasound (US).

2.2 Image Registration

In image processing, image registration is the fundamental task of finding a mapping, or transformation, between two images both spatially and with respect to intensity [9]. Minimizing the effect of the displacement problem is the purpose of registration [75]. Medical image registration is useful for diagnosis and staging of diseases, planning treatment, guiding surgery or interventions, studying disease progression, and researching cohorts of patients [48] [9].

There are generally two categories of medical image registration: rigid registration and non-rigid registration. Rigid registration is based on rigid-body transformation; this includes only rotation and translation. Non-rigid registration is based on deformable transformations with more degrees of freedom [12]. CT scans, the main image type considered in the research, require only rigid registration. Since there are distortion and deformation issues when histological sections are acquired, non-rigid registration is required when processing them.

Dawant [12] reported that the studies of medical image registration have had a rapid increase since 1990, and it now represents a substantial portion of medical image processing research. However, most of the literature only covers rigid image registration. Auer et al. [3] stated that most registration methods are only able to perform rigid-body motion and are sensitive to noise and artifacts. Thus the current trend is the study of non-rigid registration.

Least squares (refer to Section 3.1.1) is widely applied in image registration [55] [57] [56] [18] [28] [59]. It uses a sum of squares of differences (SSD) cost function. A SSD cost function method in [56] produces rigid transformation parameters which result in accurate registered images. The contrast and brightness adjustment of the reference image is optimized in [55] to correspond with the other image so that the best match over all linear intensity remappings is found. An efficient feature-based

SSD-type image alignment algorithm in [57] is suitable for aligning different types of medical images.

Mutual information (MI) [68] is another important registration technique. It measures the information in the first image that is shared by the second one. In Luo's thesis research [45], he applied mutual information registration, together with gradient information, to solve a CT image registration problem. A similar method was applied by Li [43] in her thesis on music retrieval. Mutual information (MI) has a large number of applications in rigid registration [43] [48] [74] [61] [62] [63]. Mutual information based registration has become exceedingly widespread [34] [47] [73]. Likar et al. [44] reported that MI maximization was a powerful tool for CT, MRI, and PET images.

There are many applications of intensity based registration [3] [9] [58] [81]. Lazebnik et al. [41] and Hellier et al. [30] developed methods utilizing landmark based registration where corresponding landmarks in the source image and target image are identified. The research of Johnson et al. [35] indicates that better image correspondence is produced by applying landmark and intensity information together rather than applying either one alone. Hsu et al. [32] proposed a registration algorithm that extracts image features with a hierarchical design. In this type of algorithm, the images are registered globally and subdivided into multiple images for local registration. One source image can be divided into 4, 16, 64, and then 256 sub-images hierarchically. Since including only one step in the registration process usually does not give satisfactory results, there are studies that have developed hierarchical registration algorithms [44] [3].

As this thesis mainly focuses on CT images, where rigid registration is sufficient, non-rigid registration and issues in registering histological sections are discussed in Appendix A.1.

2.3 Geometrical Transformations

Geometrical transformations play an important role in image registration. Global, rigid, affine, and projective transformations are most frequently used [48] [85]. Global transformations are applied to the entire image, while local ones involve subdividing the image into a number of regions and transforming sub-regions of the image individually. Local transformations can be done hierarchically.

Geometrical transformations are performed with the goal of aligning images so that corresponding points of two or more images will appear at the same position. Luo [45] states that the process of registration is finding an optimal transformation to match information in two or more images.

Several options for extending the process to non-rigid registration are affine transformations, projective and curved transformations [50] [85]. Affine transformations include rotation, translation, shearing, and scaling.

Auer designed a hierarchical non-rigid registration algorithm that is able to align images, using a transformation based on elastic thin plate splines [3]. A thin plate spline is based on the physics analogy of bending a thin plate of metal. It was first applied in the registration of remote sensing images by Goshtasby [27] and then in the registration of medical images by Bookstein [7].

2.4 Image Segmentation

Segmentation is the process of partitioning a digital image into multiple sets of pixels. The representation of an image can be simplified by segmentation so that analysis is easier [16] [70]. During the segmentation process, a label is assigned to every pixel in an image, and certain visual characteristics are shared by pixels with the same label. In this section, several segmentation techniques are reviewed, e.g., thresholding,

clustering, region growing, edge detection and semi-automatic segmentation.

Thresholding is a simple image processing technique. The image pixels with values greater or less than a certain threshold value are classified to be image foreground or background (more details are given in Section 3.2.1). It is commonly used in a wide range of applications. For example in recent years, Pham [60] addresses threshold and positive color selection in quantitative image analysis of histological sections, in order to decrease effects of observer biases.

Clustering is the process of organizing objects into groups whose members are similar in a certain way [1]. The most commonly used method of cluster detection in practice is k-means. Section 3.2.2 describes a form of k-means clustering that was first published by MacQueen in 1967 [46].

Region growing starts from one or more seed points and groups pixels or sub-regions into larger regions based on predefined criteria. This method has several advantages over conventional segmentation techniques. Further description of the region growing technique is given in Section 3.2.3.

Since 1980, numerous edge detection techniques have been investigated [11] [14] [40] [33] [5] [76] [71] [6]. Canny [11] suggested that good detection, good localization, and low spurious responses were the three key criteria of a good edge detector. Canny also described how to define the optimal detector for an isolated step edge. Detailed information for this technique is given in Section 3.2.4.

Semi-automatic segmentation (often referred to interactive segmentation) is widely used in graphics editors and image manipulation programs as interactive tools. Magnetic Lasso in Adobe Photoshop is a great example of this kind of segmentation, and is one of the most popular tools of this type. It accepts input about the region of interest (ROI) from the user and applies algorithms that find the best curve to fit the edge of the image. Intelligent Scissors Tool is another segmentation tool, created by Mortensen [52] [53] [15].

Two other important tools of this type are Siox [21] and Livewire [4]. Livewire is similar to the Magnetic Lasso Tool in Adobe Photoshop, and can find image boundaries. It is implemented as an open source image segmentation tool, e.g., ImageJ, which is a Java-based image processing program.

Simple Interactive Object Extraction (SIOX) is one of most popular segmentation methods in the open source world for selecting foreground objects from images. This technique uses two brushes for foreground and background marking. It takes very little user interaction using a free hand selection tool, to label the foreground and background. Then, foreground objects can be extracted. The foreground selections can be refined by further markings either on the foreground or the background [20] [21] [22]. SIOX can be found in the world's most popular graphics programs, e.g., GIMP, Inkscape, ImageJ and Fiji (plug-in).

2.5 Interpolation

The process of determining the value of the function at positions that lie between its samples is interpolation. Interpolation techniques have been discussed in a wide range of applications in image registration, especially in non-rigid registration.

There are generally two categories of interpolation techniques, deterministic and statistical [25]. Since statistical interpolation is rarely applied in image registration [25], only deterministic interpolation methods are discussed in this thesis. Lehmann et al. [42] introduced several interpolation techniques. Schnabel et al. [67] described an interpolation method using finite-element methods.

Several common types of interpolants are nearest neighbor, linear, polynomial (e.g., quadratic, cubic or higher order), cubic spline, and monotonic piecewise cubic interpolants. Nearest neighbor interpolation finds the nearest sample point to the desired point and assigns it the value of the nearest sample point [25]. This can

be viewed as piecewise constant interpolation. Linear interpolation assumes a linear relationship among sample point values.

Polynomial interpolation is the interpolation of a given dataset by a polynomial. It has comprehensive applications in image processing. A spline is a piecewise polynomial function [13]. A piecewise function has different representation in each piece of the domain. The most common spline interpolants are linear, quadratic, and cubic splines.

There is general agreement on the important role of cubic splines in image processing [42] [31] [66]. Spline interpolation may involve polynomials of higher degree than cubic.

2.6 3D Reconstruction

3D reconstruction from a serial image dataset is used to generate 3D virtual models so that the shape and appearance of real objects can be captured. After the processes of registration, geometrical transformation, segmentation and interpolation, 3D reconstruction is the end process required to build 3D models of a human temporal bone.

3D reconstruction of virtual models using histological sections as well as CT and MRI images has been done in many studies. In some of the early work in this field, Takagi et al. [77] [78] developed computer-generated wire-frame models of various structures in the inner ear from histological sections. In order to display the reconstructed images of human temporal bones in 3D, Takahashi et al. [79] also developed a 3D imaging technique. In some other recent studies, Rusinkiewicz et al. [65] proposed a new 3D model reconstruction method which allows the user to rotate an object by hand and see a continuously-updated model as the object is scanned.

In several related studies, a complete 3D virtual model of the major human ear

structures was constructed using computer-aided design (CAD) software by Jones et al. [36], and the 3D modeling procedure used to form the histological section images was demonstrated. Auer et al. [3] proposed a segmentation tool that allows a 3D reconstruction of the most important tissue components. Braumann et al. [8] reconstructed 3D data from tumoral tissue, and also introduced a list of image processing steps for the reconstruction of tumor invasion fronts from histological sections. Ju et al. [37] conducted a 3D volume reconstruction of a mouse brain from histological sections where a Gaussian filter (refer to Section 3.3.4) was applied.

Wang et al. [84] developed a human temporal bone 3D virtual model using histological sections and the Amira software [2], which is discussed below. Wang's work is relevant to this thesis, because, in our thesis, the target region is the human temporal bone, and the tool we use is Amira.

Amira is a commercial software package for 3D modeling and visualization. It is a product from Visage Imaging, Inc. The version 4.1 of Amira is installed in the graduate student lab computer (Intel Duo CPU 3.16GHz, 1.93 GB of RAM; Window XP). Amira has limitations in performing registration, segmentation and 3D reconstruction. In Amira, only rigid registration methods are provided, and the segmentation methods are mostly interactive and manual. Therefore, results for 3D reconstruction obtained using Amira are not satisfactory.

Chapter 3

Theory

This chapter describes the theory behind the methods and algorithms used in this thesis. They are: 1) image registration; 2) image segmentation; 3) noise and filtering; 4) interpolation; and 5) 3D reconstruction.

3.1 Image Registration

Image registration maps two or more images into a single coordinate system through geometrical transformations. It corrects for the displacement of the images. Images from a single CT scan as provided in this study do not have displacement issues so they do not need to be registered. Registration is needed for histological sections due to the acquisition process and the other issues discussed in Section 2.1.3. This section introduces the least squares technique applied in our research involving histological sections.

3.1.1 Least Squares

The least squares registration method is used to align two successive images by transforming the first image to align with the second. After each transformation the sum

of squared differences (SSD) of gray level values of the two images is computed. The process is iterated until the minimum SSD is reached.

For two images, A and B, the SSD between A and B, is

$$SSD(A;B) = \sum_{j=0}^{H-1} \sum_{i=0}^{W-1} (G_A(x_i, y_j) - G_B(x_i, y_j))^2, \tag{3.1}$$

where G_A and G_B are the gray values for the images A and B, W and H are width and height of images A and B in pixels, and (x_i, y_j) is the coordinate of a pixel. A lower SSD value indicates a higher similarity between images [56] [55]. The more pixels that have the same gray values, the better the alignment quality that will be achieved.

3.2 Image Segmentation

The objective of applying segmentation techniques in this thesis is mainly to identify regions of interest (ROIs) in the histological sections. Although manual segmentation was done on the provided CT scans, they still needed further segmentation to find edges and contours, so we also have applied segmentation techniques to the CT scans. This section introduces the segmentation techniques relevant to this thesis. Thresholding, k-means clustering, region growing and Canny edge detection are described.

3.2.1 Thresholding

In the process of thresholding, a certain threshold value is used to partition an image into two pixel groups. In a grayscale image, if the gray value of an individual pixel is greater than the threshold, typically, it is given a value of "1". Otherwise, it is given a value of "0".

As the simplest method of the image segmentation techniques, thresholding can

create only binary images. A grayscale image can be transformed to one having black and white pixels after thresholding [69]. Adaptive thresholding uses different threshold values for different image regions. It is sometimes called local or dynamic thresholding [69]. Choosing the appropriate threshold value is often the key to the success of the thresholding process; this is typically done manually (chosen by the user), or automatically (by a computer algorithm).

3.2.2 Clustering

Clustering refers to categorizing source data into a number of groups, or clusters. It is a convenient technique for discovering data distribution and patterns in underlying data. The discovery of dense and sparse regions in a dataset is the primary goal of clustering [1]. Clustering can be considered the most important algorithm for unsupervised learning. (This means it deals with finding structure in a collection of unlabeled data.)

The k-means clustering algorithm [46] take as input a predefined number of clusters, k. Means stands for an average: the average location of all the members of a single cluster. This location is called a centroid. k centroids need to be determined for every data object; this involves grouping the members of the data object into k clusters. At first, members of each clusters were found by measuring the distance between the initialized centroid and them. Within each iteration of the algorithm, data objects are reassigned to the k clusters and centroids are recalculated. The algorithm terminates once the centroids do not change from one iteration to the next.

K-means clustering was studied and implemented in our research. The algorithm, implementation and sample results are given in Appendix B.1, with a brief discussion to point out that it is not suitable for our goal of segmenting only one or two objects from images.

3.2.3 Region Growing

Region growing is a procedure that groups pixels or sub-regions into larger regions based on predefined criteria. The basic approach is to start with a set of "seed" points. If neighboring pixels have properties similar to the seeds, we append them to the region surrounding the seed point [25]. The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available. A possible property for examining neighboring pixels could be gray levels or pixel values.

The similarity criteria could be a fixed or a variable tolerance in a gray-level range. In our application, the standard deviation of the seed points is selected as the similarity criteria. If the difference between neighboring pixels and the seed is less than the standard deviation of the seed points, they are included in the seed points. Since the seed points are changed over a number of iterations, the variable tolerance should be recalculated. When no more pixels satisfy the criteria for inclusion in the region, the process of growing stops [25].

3.2.4 Canny Edge Detection

This method was proposed by Canny [11] [49]. The method is basically used to detect edges in images. Edges carry important information about an image. Edges are characterized by abrupt changes of intensity or color in the images.

Canny edge detection starts with Gaussian filtering (refer to 3.3.4) to smooth the image. Then, it uses the Sobel operator (refer to Section 4.3.4) to compute the magnitude and orientation of the gradient using finite-difference approximations for the partial derivatives. After that, it applies non-maxima suppression to the gradient magnitude to generate a thin line in the output image. Non-maxima suppression relates the edge direction to a direction that can be traced in an image. Further

details are given in Section 4.3.4. Finally, it detects and links edges using a double thresholding algorithm with hysteresis. The edge points are output, if their gray values are above a high threshold. Hysteresis is applied to determine whether connected segments to these points contain points with gray values higher than a low threshold. This method can efficiently generate direction information for edges in several orientations and then integrate them into a single set of thin edges [11].

The design for obtaining edge profiles is based on the specification of detection and criteria of localization [11]. The specification of the number of directions is found by non-maxima suppression. There are three performance criteria: good detection (neither failure in marking real edge points nor false marking of non-edge points), good localization (the marked edge points should be close to the real edge center) and only one response to one edge (when the same edge has two or multiple responses, only one should be marked true) [11]. The Canny edge detection algorithm is described in Section 4.3.4.

3.3 Noise and Filtering

Noise is random fluctuations in a signal, and is unwanted. Filtering can smooth the signal so that it has only valid data, with invalid noise removed. Noise exists in both CT scans and histological sections. The sources of noise are data collection and data processing steps. Acquisition of CT scans includes a small amount of noise introduced through the automatic CT scanner. The manual processes of fixation, decalcification, embedding, cutting, staining and digitizing of histological sections all introduce inaccuracies and variability, implying substantial noise. In our algorithms, image segmentation is not done perfectly and introduces noise. In the contour finding algorithm, the selection of control points on the segmented edge contour represents a significant source of noise. Such noise needs to be filtered so that a valid signal can

be input to the interpolation algorithm.

A digital image can be thought of as a set of spatial domain signals. The spatial domain is the normal image space with pixel values corresponding to the pixel locations (x and y coordinates). The frequency domain is the domain for analysis of signals with respect to frequency. This section describes median filtering as a spatial domain technique, and Gaussian filtering as a frequency domain technique.

An important theoretical tool in this area, the Fourier transform, is described, as are the discrete Fourier transform, the fast Fourier transform, and the power spectrum. The Shannon sampling theorem is briefly introduced. The theory behind development of a Gaussian low-pass filter (GLPF) is the emphasis of this section. Important questions are answered, e.g., how to build a suitable GLPF, with appropriate parameters, to smooth the frequency domain signal. Both one-dimensional (1D) and 2D Gaussian filtering methods are introduced, as appropriate for 1D signals or 2D images, respectively.

3.3.1 Median Filter

Median filtering is a non-linear filtering technique used to remove noise from images or other signals. The idea is to examine an input sample and determine if it is a valid representative of the signal.

Median filtering replaces the value of the pixel by the median of the gray levels of the neighboring pixels. The window (sometimes also refers to a mask) is usually a rectangle that includes an odd number of samples. The mask includes all pixel samples in the rectangle. For example, it has 9 samples if it is 3 pixel by 3 pixel or 25 samples if 5 pixel by 5 pixel. The median of the sample values is determined and this value is assigned to the pixel at the center of the mask, replacing the previous value. This calculation is repeated, with the mask traversing the entire image.

Median filters are popular since they provide excellent noise-reduction capabilities. They are particularly useful for reducing certain types of random noise, e.g., speckle noise (granular noise with gray dots) and salt-and-pepper noise (black or white noise pixel; looks like salt and pepper in an image). They have an edge-preserving feature, and do less blurring than mean filters with similar dimensions. A mean filter applies the mean value in the image mark, unlike the median filter uses the median value. Median filtering is a spatial domain technique, because it modifies the pixel values directly in the 2D space.

3.3.2 The Fourier Transform

Although spatial domain filtering techniques can smooth given signals or images effectively, there is another group of techniques for filtering and smoothing that operates in the frequency domain. The frequency domain is the space that is defined using the Fourier transform of a signal.

The Continuous Fourier Transform

A spatial domain signal can be transformed to the frequency domain using the Fourier transform, which transforms a real variable x and the continuous function $f(x)$ to a frequency domain representation [26]. This transform is defined by

$$\mathbb{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}dx, \tag{3.2}$$

where $\mathbb{F}$ denotes the Fourier transform process, $F(u)$ is the resulting function in the frequency domain, u is a given point in the frequency domain, and $j = \sqrt{-1}$.

The inverse Fourier transform can transform a continuous function in the fre-

quency domain, $F(u)$, back to a continuous function in the spatial domain, $f(x)$:

$$\mathbb{F}^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ux}du. \tag{3.3}$$

Equations (3.2) and (3.3) are called the Fourier transform pair [25].

The Discrete Fourier Transform

Assuming we take N samples, $\triangle x$ units apart, starting at x_0, a continuous function $f(x)$ can be evaluated at the discrete points $(x_0, x_0+\triangle x, x_0+2\triangle x, ..., x_0+[N-1]\triangle x)$ to yield a sequence of values

$$\{f(x_0), f(x_0+\triangle x), f(x_0+2\triangle x), ..., f(x_0+[N-1]\triangle x)\} \tag{3.4}$$

The function $f(x)$ is then defined at discrete locations such that

$$f(x) \triangleq f(x_0 + x\triangle x) \tag{3.5}$$

where $x = 0, 1, 2, ..., N-1$ [25].

Similarly the frequency domain function $F(u)$ can be evaluated at the discrete points $(0, \triangle u, 2\triangle u, ..., [N-1]\triangle u)$, so we define the discrete function $F(u)$ in a similar manner, i.e.,

$$F(u) \triangleq F(u\triangle u) \tag{3.6}$$

for $u = 0, \triangle u, 2\triangle u, ..., [N-1]\triangle u$.

Therefore, the discrete Fourier transform (DFT) pair for sampled functions is

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \tag{3.7}$$

for $u = 0, 1, 2, ..., N-1$, and

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{j2\pi ux/N} \tag{3.8}$$

for $x = 0, 1, 2, ..., N-1$.

The terms $\triangle u$ and $\triangle x$ have the following relation [25]

$$\triangle u = \frac{1}{N}\left(\frac{1}{\triangle x}\right). \tag{3.9}$$

In the spatial domain,

$$\triangle x = \frac{L}{N} \tag{3.10}$$

where L is the total length of x-axis [26]. In the frequency domain, rewriting Equation (3.9), we have the following relations,

$$\triangle u = \frac{1}{L}, \quad N\triangle u = \frac{1}{\triangle x}. \tag{3.11}$$

The Fast Fourier Transform

The fast Fourier transform (FFT) is an efficient algorithm for the computation of the Fourier transform. The FFT algorithm is a decomposition procedure that has an operation count for multiplication and addition proportional to $N \log_2 N$ [26]. The algorithm has the form

$$\mathbb{FFT}\{f(x)\} = F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) {W_N}^{ux} \tag{3.12}$$

where $u = 0, 1, \cdots, N-1$, and

$$W_N = e^{-j\frac{2\pi}{N}}. \tag{3.13}$$

It can be shown that if N is a power of 2, then $F(u)$ can be decomposed into two parts and significant computational efficiencies can be obtained.

Compared to Equation (3.7), the FFT algorithm significantly reduces the computational effort. In Equation (3.7), the summation requires N complex multiplications of $f(x)$, $x = 0, 1, 2, \cdots, N-1$, by $e^{-j\frac{2\pi}{N}}$ and followed by $N-1$ additions. Particularly for large N, the FFT has a significant computational advantage over the DFT. For example, using FFT for $N = 512$ is 56.89 times as fast as using the DFT [26]. FFT can only be applied when N is a power of 2 (i.e., $N = 2, 4, 8, 16, ...$).

3.3.3 The Shannon Sampling Theorem

In the experimental data, there are valid data and noise, due to the acquisition system and previous precessing steps. The use of the Shannon sampling theorem allows a valid representation of the valid data to be obtained from samples of the experimental data.

In signal processing and image processing, the Shannon theorem is important in signal sampling and reconstruction, in order to prevent aliasing. Aliasing is a phenomenon where additional frequency components, or aliased frequencies, are introduced into the sampled signal, and the sampled signal is corrupted [25]. It typically happens if the sampling rate is too low. The sampling rate, $\mathcal{F}_s$, is the number of samples taken within a unit distance, when taking samples from a continuous signal.

The Shannon sampling theorem states that if the sampling rate is at least twice the bandwidth (the width of a frequency range), then uniformly-spaced discrete samples will give a complete representation of the signal. In the case of a low-pass filter (refer to Section 3.3.4), which removes high frequencies, the bandwidth, B_{width}, is the width between lower and upper cutoff frequencies. The term cutoff frequency (described in Section 3.3.4) indicates the threshold frequency. Frequencies outside the range of the lower and upper cutoff frequencies are removed, or cut off. The condition that is sufficient to reconstruct the exact signal from samples at a uniform $\mathcal{F}_s$ is:

$$\mathcal{F}_s \geq 2B_{width} \tag{3.14}$$

If $B_{width} > \mathcal{F}_s/2$, the signal is undersampled, and aliasing would occur.

3.3.4 The Gaussian Low-Pass Filter (GLPF)

In signal processing, filtering is used to remove or reduce certain unwanted frequencies in a signal. The GLPF is a filtering technique that can be applied either in the frequency domain or in the spatial domain. A low-pass filter is a filter that passes low frequencies well, but attenuates (or reduces) frequencies higher than a certain cutoff frequency associated with the filter.

The idea is that a given signal has an outline, or overall fluctuation, as well as details, or smaller waves. Smoothing is often used to remove details, while keeping the primary characteristics of the signal. However, this task may not be straightforward using spatial domain filtering techniques, because it may be difficult to separate overall behavior from the details. The Fourier transform converts the spatial signal to the frequency domain, so that high frequency values, representing the details, can be removed easily and accurately.

After the input spatial signal $f(x)$ is transformed to the frequency domain, it becomes a frequency domain function, $F(u)$. Filtering in the frequency domain can be done through multiplying by a filtering function $H(u)$ [25]. Then, the inverse Fourier transform of the above filtered frequency function is computed in order to compute the filtered result in the spatial domain [25]. Below we discuss GLPF functions in the spatial and frequency domains for both 1D and 2D.

1D GLPF

The 1D GLPF can be applied to a smooth 1D signal. The Fourier transform of the filtered signal, $G(u)$, is given by

$$G(u) = H(u)F(u) \tag{3.15}$$

where $H(u)$ is the Fourier transform of the filter (to be discussed shortly), and $F(u)$ is the Fourier transform of the original signal. The filtered result in the spatial domain, $g(x)$, is obtained by taking the inverse Fourier transform of $G(u)$.

The Gaussian low-pass filter function is given by the following two equations. In the frequency domain,

$$H(u) = Ae^{\frac{-u^2}{2\sigma^2}}, \tag{3.16}$$

where σ is the standard deviation of the Gaussian curve. A is the gain factor of the GLPF. It controls the height of the filter curve. $A = 1$ means that the filter height is not changed. In the spatial domain, the corresponding filter function, $h(x)$, is

$$h(x) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2x^2}. \tag{3.17}$$

Multiplication of two functions in the frequency domain corresponds to convolution in the spatial domain. Hence the spatial domain equivalent of Equation (3.18) is

$$g(u) = h(u) * f(u) \tag{3.18}$$

where $*$ indicates the convolution operation.

2D GLPF

The 2D GLPF can be used for 2D signal denoising, and in particular in image filtering. The method can reduce noise in an image or a 2D signal, by multiplying the image signal by the filtering function in the frequency domain.

The 2D filtering equation is

$$G(u,v) = H(u,v)F(u,v) \tag{3.19}$$

where $F(u,v)$ is the Fourier transform of the image before filtering, $H(u,v)$ is the frequency domain filtering function and $G(u,v)$ is the Fourier transform of the image after filtering.

The 2D Gaussian low-pass filter in the frequency domain is

$$H(u,v) = Ae^{\frac{-(u^2+v^2)}{2\sigma^2}}, \tag{3.20}$$

while in the spatial domain it is

$$h(x) = 2\pi\sigma^2 Ae^{-2\pi^2\sigma^2(x^2+y^2)}. \tag{3.21}$$

Power Spectrum, Cutoff Frequency and Sigma

When applying the GLPF to attenuate the noise, the actual amount of attenuation for each frequency depends on the filter design. The key element is the σ value, which determines the width of the GLPF curve, affecting how much high frequency information will be leave alone.

As σ is the standard deviation of the Gaussian function curve, one approach is to define the shape of the GLPF curve according to the Fourier transform of the signal, $F(u)$, and then compute the σ value based on $F(u)$. This way, the GLPF based on

this σ value would match the width of $F(u)$ and have reasonable filtering results.

There is a relationship between a suitable σ value of the GLPF and the shape of the Fourier spectrum of a given function which is discussed below. However, it appears that there may not be a simple formula for the determination of σ. In order to work out such a relationship, the definition of the power spectrum of a signal in the frequency domain needs to be introduced.

The Fourier transform uses complex variables. The spectrum or magnitude of the Fourier transform is very important for filtering in the frequency domain [25]. The Fourier spectrum $|F(u)|$ is given by:

$$|F(u)| = \sqrt{[R^2(u) + I^2(u)]}, \tag{3.22}$$

where $R(u)$ and $I(u)$ are, respectively, the real and imaginary parts of $F(u)$.

The power spectrum, $P(u)$, of a signal is the square of the magnitude of the Fourier transform. It describes how the power of a signal is distributed with respect to frequency [25]. $P(u)$ is given by

$$P(u) = |F(u)|^2 = R^2(u) + I^2(u). \tag{3.23}$$

Summing the components calculated in Equation (3.23) for $u = 0, 1, 2, \cdots, N-1$, gives the total 1D signal power, P_T,

$$P_T = \sum_{u=0}^{N-1} P(u). \tag{3.24}$$

Gonzalez [25] described one way to use the summed power spectrum of an image in the frequency domain to compute the cutoff frequency. The approach works similarly for a 1D signal, using the signal power to compute the cutoff frequency.

As mentioned earlier, the GLPF removes high frequency components for the

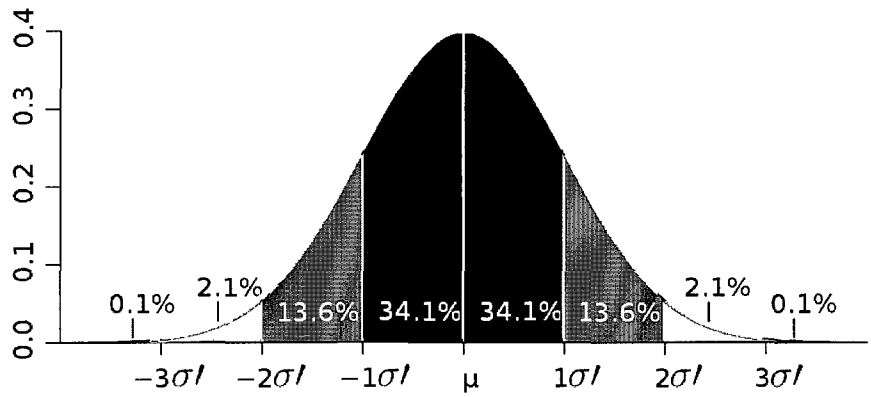

Figure 3.1: Standard deviation, confidence intervals and three-σ rule. For the normal distribution, one standard deviation σ' away from the mean encloses about 68% of values; about 95% of the values are within the two σ' range; and about 99.7% lie within the three σ' range.

Fourier transform. The ideal low-pass filter (ILPF) is a much simpler filter; it has the following definition

$$H(u) = \begin{cases} 1, & \text{if } D(u) \leq D_0 \\ 0, & \text{if } D(u) > D_0 \end{cases} \tag{3.25}$$

where $D(u)$ is the distance between u and the origin center of the transform, and D_0 is a specified nonnegative distance from the origin. The ILPF "cuts off" all components with a distance greater than D_0 from the origin of the transform, which is called the cutoff frequency, $\mathcal{F}_{cutoff}$.

The D_0 distance from the origin of the transform encloses a percentage, α, of the signal power. The rest of the signal power needs to be filtered. The three-σ rule [64], shown in Figure 3.1, is useful to determine how much signal power to cut off. As D_0 of ILPF decreases, more power is removed, which results in more smoothing. σ' in the three-σ rule is the standard deviation of the normal distribution. This is different from σ in the GLPF, which is a parameter in Equations (3.16) and (3.17).

The ILPF is useful for understanding basic filtering concepts and it can be easily

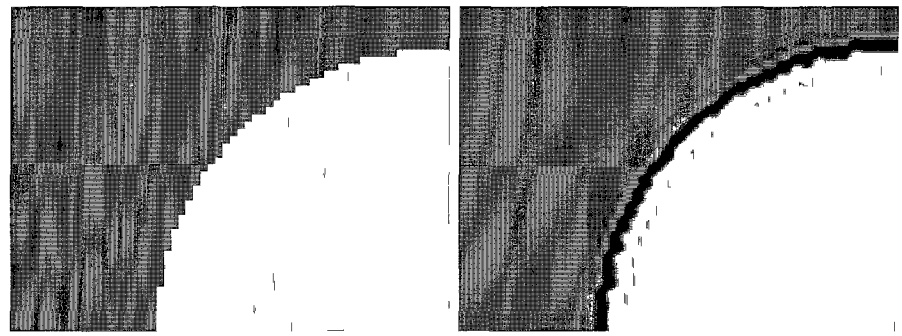

Figure 3.2: An example of the ringing effect. From left to right (a-b): a. an image without the ringing effect; b. the same image showing the ringing effect.

implemented, but it is not practical and causes aliasing. The "ringing" effect can be seen in an image that has been filtered by the ILPF. Figure 3.2 shows an example of the ringing effect, which appears as bands, or "echos" near the edge.

Applying a ILPF in the frequency domain generates multiple peaks in the spatial domain. A GLPF function does not have this issue. It does not have a sharp cutoff and thus does not exhibit ringing in the spatial domain.

Finally, since σ of the GLPF determines the spread of the curve, σ can be set to D_0 [25], and Equation (3.16) can be expressed as

$$H(u) = e^{\frac{-D^2(u)}{2D_0^2}} \tag{3.26}$$

The gain factor A in Equation (3.16) is set to 1, so that the peak of the GLPF is 1. The peak is at the origin of the GLPF where multiplying the GLPF by a signal does not change the signal. Using $D(u)$ to replace u in Equation (3.16) results in normalization of the GLPF. The GLPF is symmetrical, with the origin at $D(u) = 0$. When $D(u) = D_0$ (or $D(u) = \sigma$), the filter has a value that is 0.607 of its maximum, since $H(u) = e^{\frac{-1}{2}} = 0.607$. Interestingly, such a $D(u)$ value arises when u equals σ

In conclusion, the signal power can be analyzed using the power spectrum. The three-σ rule allows us to estimate how much power to cutoff, so that D_0 can be com-

puted. The ILPF illustrates the idea of using a low-pass filter to remove frequencies higher than D_0, giving the cutoff frequency, $\mathcal{F}_{cutoff}$. The GLPF uses the same idea as the ILPF, but does not cause aliasing while cutting off high frequencies.

3.4 Interpolation

Interpolation techniques are important in numerical analysis and have been used in a variety of applications in image processing. Piecewise linear interpolation (refer to Section 3.4.1), is a simple and standard method provided by a variety of commercial software packages. Better interpolants can be obtained by interpolation with higher degree polynomials. Our goal is to find a suitable polynomial interpolation method to use on image data to generate missing slices in histological sections data. For this purpose, this section introduces linear, polynomial, cubic spline, and monotonic piecewise cubic interpolation. It also describes an adaptive interpolation algorithm to ensure accuracy while optimizing data usage.

3.4.1 Nearest Neighbor and Linear Interpolation

Nearest neighbor and linear interpolation techniques are relatively simple. They are often applied in fast, coarse image processing. Nearest neighbor interpolation finds the nearest sample point to a desired point and assigns its value to the desired point [25]. This can also be described as piecewise-constant interpolation.

Piecewise linear interpolation assumes a straight line segment between two given data points. In this case, given two points (x_0, y_0) and (x_1, y_1), for an x value in the interval (x_0, x_1), the corresponding y value is given by

$$y = (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} + y_0. \tag{3.27}$$

Piecewise bilinear interpolation is an extension of piecewise linear interpolation for interpolating functions in two independent variables [25] by performing linear interpolation in both dimensions.

3.4.2 Polynomial Interpolation

Polynomial interpolation is the interpolation of a given dataset by a polynomial. In other words, given a set of data points, the aim is to find a polynomial that goes exactly through these points. Given $n+1$ data points, assume a polynomial of degree n of the form:

$$y = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n, \tag{3.28}$$

where $a_0, a_1, ..., a_n$ are $n+1$ real constants. Since $n+1$ values of y are given at $n+1$ values of x, one can write $n+1$ equations obtained by requiring that the value of the polynomial agree with the given y value. For the data points (x_i, y_i), we require

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + ... + a_n x_i^n. \tag{3.29}$$

Then the $n+1$ constants, $a_0, a_1, ..., a_n$, can be found by solving the system of $n+1$ simultaneous linear equations.

More advanced techniques such as Lagrange interpolation or Newton interpolation can also be used to obtained the polynomial interpolants [38]. A potential issue is that the resulting polynomial may have oscillatory behavior when n is large. Consequently, spline interpolation can be a more appropriate choice when n is large since the resulting interpolant tends to have values closer to the data points.

3.4.3 Cubic Spline Interpolation

A spline is a piecewise polynomial function [13]. Cubic splines are piecewise cubic polynomials. Between each pair of data points a unique cubic polynomial is generated such that the curve is continuous and smooth, and interpolates the data points.

The piecewise function has the form:

$$V(x) = \begin{cases} v_1(x), & \text{if } x_1 \leq x \leq x_2 \\ v_2(x), & \text{if } x_2 \leq x \leq x_3 \\ & \vdots \\ v_{n-1}(x), & \text{if } x_{n-1} \leq x \leq x_n \end{cases} \tag{3.30}$$

where $v_i(x)$ is a third degree polynomial defined by

$$v_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{3.31}$$

for $i = 1, 2, ..., n - 1$. The first and second derivatives of $V(x)$ are fundamental to this process. We have

$$v_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \tag{3.32}$$

$$v_i''(x) = 6a_i(x - x_i) + 2b_i \tag{3.33}$$

for $i = 1, 2, ..., n - 1$.

Each segment of the cubic spline is defined by three constraints [72], which are:

1. $V(x)$ will interpolate all data points. It follows that $V(x)$ will be continuous on the interval $[x_1, x_n]$.

2. The first derivative of $V(x)$ will be continuous on the interval $[x_1, x_n]$

3. The second derivative of $V(x)$ will be continuous on the interval $[x_1, x_n]$

This leaves two degrees of freedom in the definition of $V(x)$; a variety of techniques have been developed to specify the two remaining degrees of freedom [72].

3.4.4 Monotonic Piecewise Cubic Interpolation

A variant of cubic spline interpolation is monotonic piecewise cubic interpolation. Monotonic piecewise cubic interpolation yields an interpolant that preserves the monotonicity of the interpolated dataset. A monotonic piecewise cubic interpolant has a continuous first derivative, unlike a cubic spline interpolant that has two continuous derivatives.

The main reason that monotonic piecewise cubic interpolation is considered is that the direction of the curve between data points is controlled to be either monotonically increasing or decreasing. Monotonically increasing means that the function value is strictly increasing from the left to the right, and the next value cannot be less than the previous value. Monotonically decreasing is the opposite. In many cases, such as medical and histological data, to fit the physical data better, monotonicity of the interpolant is a useful property. The coefficients of the cubic on each subinterval are determined to interpolate the associated data points, to provide continuity of the first derivative, and to preserve monotonicity.

Two examples were constructed using Matlab to compare the interpolated values of a monotonic piecewise cubic with those of a cubic spline. For monotonically increasing data, in Figure 3.3 the monotonic piecewise cubic interpolant preserves the monotonicity, but the cubic spline does not. For non-monotonic data in Figure 3.4, the monotonic piecewise cubic deals better with the intervals in which the data changes direction.

Cubic spline interpolation and monotonic piecewise cubic interpolation have dif-

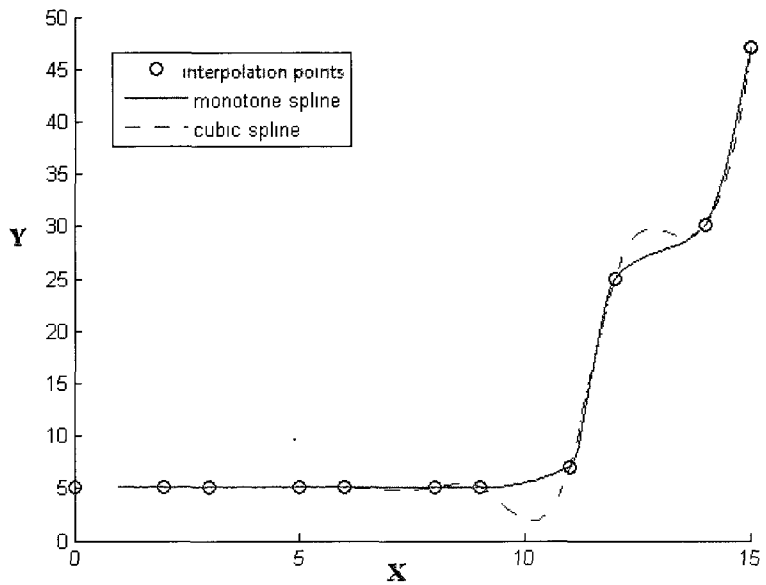

Figure 3.3: Cubic spline and monotonic piecewise cubic interpolation plotting on monotonic data

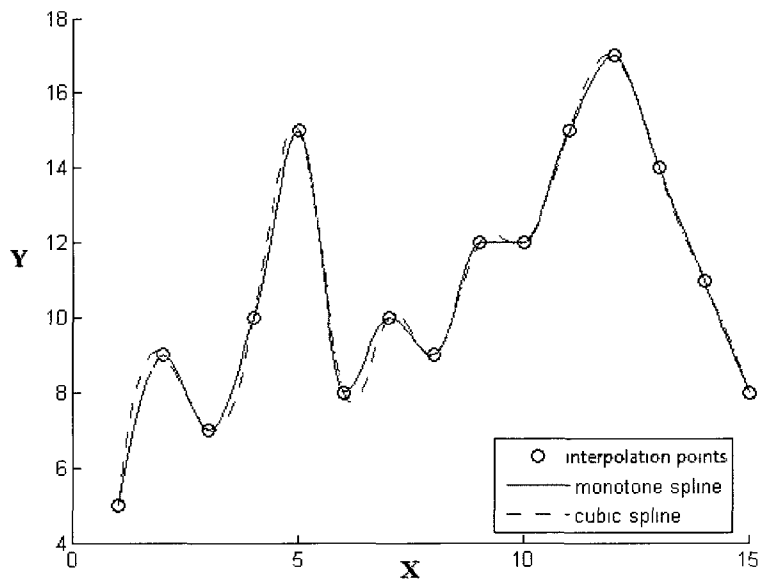

Figure 3.4: Cubic spline and monotonic piecewise cubic interpolation based on non-monotonic data

ferent features. Cubic spline algorithms are very popular. However, many examples show that cubic spline interpolation can give a poor fit, particularly if there are rapid changes in a small interval in the dataset.

Fritsch and Carlson discuss a "visually pleasing" monotonic cubic interpolant [38]. A software package for the implementation of monotonic piecewise cubic, PCHIP [23], will be introduced in Section 4.8.2.

3.4.5 Adaptive Interpolation

The idea of subdividing a task where needed is generally known as divide and conquer. Here we use this refinement idea to control the interpolation process.

The basic idea is to use an adaptive algorithm to minimize the number of subintervals over which the piecewise polynomial interpolant is defined. The algorithm starts by interpolating at small number of data points, and then estimates the error of the interpolant. A new interpolant using more data points where there are large estimated errors is then constructed. The process is iterated until each piece of the interpolant has an acceptable error or there are no more sample points available. The idea is to dynamically refine the pieces on which the piecewise polynomial is based. The algorithm uses fewer data points in regions where the estimated error is small and more data points in regions where the estimated error is large.

For histological sections with some slices missing, an algorithm is needed to generate missing data. For CT images, all the data are available to use. However, it will be useful to develop an interpolation algorithm and test it on the full set of CT data. Once this task is completed, the algorithm can then be applied to histological sections to approximate missing data.

There is a question of how good is "good enough" for each piece of the interpolant. A tolerance needs to be applied to the interpolation error estimate. The tolerance is

related to the accuracy of the data being interpolated. The interpolation process is considered to be successful if its errors are smaller than the errors introduced by the data collection process.

Interpolation errors can be estimated by comparing the ith and $(i+1)$th interpolants. Let $Interpolant_i$ be the piecewise polynomial interpolant generated by the adaptive interpolation algorithm during its ith iteration. The relative error estimate for the ith interpolant at location z is

$$\frac{|Interpolant_{i+1}(z) - Interpolant_i(z)|}{|Interpolant_i(z)|}. \tag{3.34}$$

If the difference between two successive interpolants is sufficiently small, then we will assume that the generated interpolant represents the data reasonably well.

The overall idea of the adaptive interpolation algorithm is explained here. A much more detailed algorithm description will be given in Section 4.8.3.

1. Choose $X_1 < X_2$, where X_1 and X_2 are integers smaller than the total number of data points.

2. Build interpolant1 using every X_1th data point.

3. Build interpolant2 using every X_2th data point.

4. Compare interpolant1 and interpolant2 on each piece of interpolant2 as in Equation (3.34) and as described in Section 4.8.3.

5. If the estimated error is smaller than the tolerance on certain pieces, the number of data values employed in such pieces is considered to be sufficient. If the estimated error is larger than the tolerance on a given piece, we introduce a new sample point chosen from the middle of the set of data points associated with the given piece of the interpolant. Once this new set of interpolation points

is identified over the entire domain, interpolant3 is constructed to interpolate this new dataset.

6. Iterate the above process until the estimated error is less than the tolerance on every piece, or there are no more available data points.

Monotonic piecewise cubic interpolant is chosen as the underlying interpolation technique. The non-smooth data regions where rapid changes occur use more data points than the smooth regions. Then, an overall satisfactory approximation with acceptable accuracy can be achieved using a different density of interpolation data points over various regions.

3.5 3D Reconstruction

In the context of this project, 3D reconstruction is the process of reconstructing a 3D model from an image dataset or image sequence of the model. The result is a 3D visualization of the 3D model, which is better and easier for analysis than a 2D sequence of images.

One goal of this project is to investigate methods to improve 3D reconstruction for histological sections. The methods are based on adaptive interpolation methods, from which intermediate slices can be generated to augment the dataset. With more data to work with, the resulting 3D model can be built more accurately, giving a surface that is smoother and has more detail.

Although improving 3D reconstruction is a goal of the research, 3D reconstruction is not the focus of our algorithm development. The improvements that we obtain for 3D reconstruction result from filtering and interpolation methods. 3D models generated with and without interpolation will be compared to evaluate the interpolation method. The hypothesis is that the 3D reconstruction process will be improved by

the use of interpolation methods.

Chapter 4

Methods and Algorithm Design

This Chapter presents the methods based on the theories that are described in Chapter 3. Table 4.1 presents an overview of the methods and algorithms we consider. It also indicates whether each step has been applied to CT images or histological sections. The CT images required fewer processing steps than histological sections because they were segmented before being provided to us. Also, CT images did not need to be registered, because they were not displaced during the CT scanning procedure. Also no median filtering was required for the CT scans.

We present methods and algorithms to 1) perform image segmentation; 2) obtain an equal number of contour points (data points representing ROI contours) on every segmented ROI; 3) filter control points using automatically developed GLPFs; 4) analyze errors in above steps and pass error results to the adaptive interpolation algorithm; 5) perform adaptive interpolation using monotonic piecewise cubics to generate missing structures; and 6) perform 3D reconstruction. The result is a complete, smooth and detailed 3D model. As well, data usage information for contour points is produced.

The algorithm has a streamlined and connected design, with methods grouped into four categories. The core methods are Gaussian low-pass filtering and adaptive

Number	Methods	CT Scans	Histological Sections
1	Images acquired from the EAR-Lab [17]	Yes	Yes
2	Image preprocessing - convert images to JPG	No	Yes
3	Rigid registration using Amira	No	Yes
4	Image renaming - depth information captured	Yes	Yes
5	Segmentation - adaptive region growing	Yes	Yes
6	Median filtering	No	Yes
7	Segmentation - edge detection	Yes	Yes
8	Contour finding - smooth contour	Yes	Yes
9	Gaussian low-pass filtering on slices	Yes	Yes
10	Gaussian low-pass filtering in the z direction	Yes	Yes
11	Calculate filtering error	Yes	Yes
12	Apply monotonic adaptive spline interpolation and generate intermediate slices, based on error limit	Yes	Yes
13	3D reconstruction using all the intermediate generated slices	Yes	Yes

Table 4.1: Overview of methods and steps

interpolation (developed in C#). The secondary methods are image segmentation (using C and CVlab) and contour finding algorithms (using C#). CVLab is a Linux based computer vision software tool. Other software that was used includes Amira (for image registration and 3D reconstruction), and stand-alone programs developed to allow interaction, connectivity and automation. These include region selection, image renaming and conversion tools (developed in C#). Information on these programs and algorithms is listed in Table 4.2.

Name	Platform	in	Purpose of the program
Patchgrow	Linux	C	Adaptive region growing
Median	Linux	C	Median filtering
Cannycontour	Linux	C	Canny edge detection
Kmeanss	Linux	C	Kmeans image segmentation
Zeropad	Linux	C	Image zero padding
Rename1	Windows	C#	Rename and convert Amira result images
Rename2	Windows	C#	Rename and convert CVlab result images
Selectobject	Windows	C#	Select objects and write CVlab scripts for segmentation
Reconstruction	Windows	C#	Load segmentation results; contour finding, Gaussian filtering, error analysis and adaptive interpolation; save result images

Table 4.2: List of software developed as part of the thesis research

4.1 Experimental Datasets

This section introduces the experimental datasets. Both histological sections and CT images were supplied by the EAR-Lab [17]. They were originally acquired from the Temporal Bone Foundation, in Boston, USA. There are a number of bones in the images, as described in Figure 2.6. The incus and the malleus are two ROIs studied in this thesis. These were introduced in Section 2.1.2.

4.1.1 Histological Sections

The histological section images provided to us by the EAR-Lab [17] were in PSD format. There are six sets of color images, identified in the Table 4.3. The sections are usually 20 microns thick. Every 5th or 10th (10 x 20 micron) section has been digitized and provided to us, giving the distance between two neighboring data slices as 100 or 200 microns, respectively. The slide number represents the z coordinate of the slice and indicates if every 5th or 10th section has been used. With this z coordinate, plus the (x, y) coordinates in the image plane, the 3D space of the image set is defined. The physical dimensions of the actual slides are 25 mm by 18 mm.

Case number	Case name	Number of Images
1	8486i	69
2	8486d	41
3	8655d	84
4	8655i	59
5	87213d	137
6	8913i	101

Table 4.3: Histological section datasets

Some digital images were larger than this because the canvas size was increased to accommodate rotation and translation during image alignment.

Unlike the CT dataset which has over 700 images, the histological section sets have between 40 and 140 images per set. Although they are provided at every 5th or 10th slice on average, if a slice that should be picked is broken or of low quality, a nearby slice may be provided instead. For example, Case 8486i has images with numbers 1, 6, 11, 13, 15, 21... Obviously not every 5th data value is provided. Therefore, the images are not uniformly spaced.

4.1.2 CT Images

One CT dataset was provided, as specified in Table 4.4. These images are obtained using a micro CT scanner (SkyScan 1072 Desktop x-ray Microtomograph) and are saved in BMP and JPG formats. The CT image resolution is 600 by 600 pixels. The CT images did not come in DICOM format (this is a standard medical image format). The physical dimensions of one image slice are 15.5 mm by 15.5 mm by 0.015 mm (i.e., one slice is 0.015 mm thick). The specifications for the incus and the malleus datasets are summarized in Table 4.5.

The original CT images were of poor quality. Therefore, manual segmentation was performed by the Research Associate at the EAR-Lab [17] using a priori knowledge. In this manner segmented ROIs were provided for testing the filtering and adaptive

Case number	Case name	Number of Images
1	Gastro1	751

Table 4.4: CT image set

Region	Start Image#	End Image#	# of Images	Width	Height	Depth
Incus	36	224	189	15.5	15.5	5.67
Malleus	22	245	224	15.5	15.5	6.72

Table 4.5: Physical dimensions of the incus and the malleus CT dataset in mm

interpolation algorithms designed in this thesis. Both the original CT images and segmentation results were provided to us. Segmentation was applied to every second image by the Research Associate at EAR-Lab, because that would provide enough image data.

4.2 Image Registration for Histological Sections

Image registration is needed only for histological sections; CT scans do not need to be registered, because no displacement was introduced from CT scanning. Rigid registration for histological sections was performed using Amira. Pre-processing, renaming and format conversion was done before and after registration using Photoshop and C# programs we developed.

4.2.1 Image Pre-processing

The objective of this step is to convert source images to a dimension and format suitable for registration in Amira. The original and converted format and resolutions are listed in the Table 4.6. The table shows that both the format and resolutions of the histological sections are changed, so that they can be used in Amira and later in CVlab. The CT images are not changed before being sent to Amira, but are changed

Image	Original Format	Original Resolution	Converted Format	Converted Resolution
CT	JPG	600 by 600	GIF, JPG	600 by 600
Histological Sections	PSD (Photoshop format)	varies between 2400 by 1800 and 3200 by 2400, e.g., 2038 by 2868	GIF, JPG	800 by 600

Table 4.6: Summary of Image Format Information

later for processing by CVlab.

Issues associated with the original histological sections are:

1. The PSD format can be only loaded by certain types of software, e.g., Photoshop, so images in PSD format need to be converted to a more common format.

2. PSD files are too large to manipulate easily. One PSD file is around 9M.

3. The original resolution is too high. Very high resolution images cannot be loaded into the Amira software.

Another issue is non-uniform slice spacing, due to the fact that some slices are completely corrupted and cannot be used. This issue is handled in later steps by using the file names of original images to indicate the slice positions in 3D space. For example, No. 41 and No. 46 images in Figure 2.5 have file names 41.jpg and 46.jpg, which indicates that they are in relative positions corresponding to $z = 41$ and $z = 46$.

PSD files are converted to JPG files through the following steps:

1. Apply zero padding and let all images have the same resolution: 3200 by 2400, (since the original resolutions are not always the same).

2. Resize to 800 pixels by 600 pixels

3. Convert to JPG images

4. Save them with the original file name, except use 001.jpg instead of 1.jpg

For histological sections, all images are resized only once, to 800 by 600, after zero padding, so that they all have the same image size. Early in the research, this resolution was set as a requirement by the Research Associate at the EAR-Lab [17]. The requirement was based on the 1G RAM in the lab computers and the need to load all the images at one time.

We complied with this image size requirement by reducing the resolution of the histological sections and then using the reduced resolution images throughout the process.

4.2.2 Registration Methods

Image registration of the image set is performed using the Amira software's alignment and rigid transformation modules. The AlignSlices module in Amira with least squares as the alignment algorithm is applied to correct the displacement of the images; rigid transformations, translations and rotations, are applied in space.

This step also changes the color images to gray scale images, since we do not need color images for our research. Gray scale images contain sufficient information.

A demonstration of the registration process and sample registration results done by Amira are shown in Section 5.1.1.

4.2.3 Image Renaming

The goal of this step is to attach 3D depth information to registered images. There is an issue introduced by the Amira software in the image registration step. Amira only saves images with names numbered 0, 1, 2, and so on, instead of using the original names that contain 3D depth information.

To rename registered images to preserve original depth information, we developed a C# program, Rename1. It renames a list of images to GIF images using the original names, as in the PSD files. The Rename1 program is introduced in Appendix C.2. It takes registration result images from Amira and outputs images for image segmentation using CVlab.

4.3 Segmentation Algorithms for Histological Sections

The segmentation framework was designed to work on histological sections. An object selection program was developed to select ROIs. Thresholding, clustering, region growing, and Canny edge detection techniques were implemented, tested, and customized to optimize performance. K-means clustering was not applied because it could not extract a single region of interest (ROI) from an image. The algorithm, test results for k-means clustering are presented in Appendix B.1. Methods of patch growing, median filtering, and Canny edge detection were combined in a segmentation algorithm. The algorithm starts with a stand-alone selection program, followed by a CVlab script which automatically goes through all methods and saves the results. The object selection is a semi-automatic process, and all other methods are fully automatic.

4.3.1 Object Selection

We developed an object selection program, SelectObject, which produces the input for the automatic segmentation algorithm for an image set. It is implemented in C# under Microsoft Windows. It allows users to select an object in the first image, and only select it again if the object location changes. It automatically generates the

patch coordinates and segmentation commands for a CVlab script for region growing and Canny edge detection. As the script is run in CVlab, the final segmentation is performed. CVlab has many low level image processing algorithms that ease the implementation of a segmentation algorithm.

The detailed steps for the use of the SelectObject program and its advantages are described in Appendix C.4.

The result script from this step and the image set can be loaded into CVlab for segmentation.

4.3.2 Adaptive Region Growing

Region growing is a segmentation method that we experimented with in this thesis. Extending the "traditional" region growing algorithm (refer to Appendix B.2), this section describes two modified and improved methods: two-threshold region growing and a patch growing algorithm. The patch growing algorithm is the one we finally chose to implement as our segmentation algorithm. It can segment and extract only the target region, without growing and finding other similar regions.

Two-threshold Region Growing Algorithm

After the "traditional" region growing algorithm (Appendix B.2) was tested, a modified method, two-threshold region growing, was designed. It improved upon the region growing method. The following is the algorithm description:

1. Ask the user to input a threshold value (T1) and apply thresholding (introduced in 3.2.1). Different from "traditional" region growing, this method is started by picking a threshold empirically instead of selecting a patch. Pixels in the resulting region of thresholding are the seeds.

2. Use the result from step 1 as the starting region for this step. Calculate the

mean and standard deviation of the pixel values of the seed region from Step 1. Use the standard deviation as another threshold value (T2).

3. For every pixel in the starting region, the algorithm checks the eight neighboring pixels. There is a binary image that has one bit for every pixel of the original image. If the difference between the gray level of the pixel and its neighbors is less than T2, the position in the binary image corresponding to the neighboring pixel is set to 1.

4. Do not check a neighboring pixel, if the corresponding position in the binary image has value of 1. That pixel is either inside the T1 thresholding region or has been checked already.

5. Repeat steps 2, 3 and 4 until no more neighboring pixels are added.

The resulting image is represented by a binary image with pixel values 0 for black and 1 for white.

Results obtained from applying "traditional" region growing and two-threshold region growing are given in Section 5.1.2. The "traditional" region growing method exhibits some drawbacks, as observed in our test results. Its patch selection process is only semi-automatic. The modified two-threshold region growing method is relatively efficient in our particular case. Firstly, it is a fully automatic segmentation process. Secondly, it only asks the user for one parameter, the threshold (T1). Thirdly, Figure 5.5.b obtained from two-threshold region growing exhibits better segmentation results. Basically, region growing only segments regions with fairly similar gray values, but two-threshold region growing segments the image by continuing to grow until a strong edge (gradient difference) is detected.

Nevertheless, the two-threshold region growing method requires the input of a threshold value for every image and this may be difficult to find. The images have

ROIs as well as unwanted regions. Only the ROIs are required for 3D reconstruction. The two-threshold region growing method still needs a suitable scheme for selecting a segmented ROI for each image.

Patch Growing Algorithm

Both the "traditional" region growing and two-threshold region growing methods segment the image globally. A better method is needed for segmenting a certain ROI locally, when there are many objects in the image. The patch growing method is designed to improve the two-threshold region growing method. It allows easy identification of one ROI, followed by segmentation of that ROI.

The patch growing algorithm description is as follows:

1. Select a rectangular patch completely inside the ROI which needs to be segmented. This is similar to "traditional" region growing, but the patch is applied differently.

2. The pixels within the patch are considered to be seeds. Their positions in the original image are recorded in a binary image that has one bit position for every pixel in the original image. The bits in the binary image that correspond to pixels inside the patch are set to 1.

3. Calculate the mean value, avg, and standard deviation, σ, of the selected patch.

4. For every position in the binary image with a value of 1, the algorithm checks the eight neighboring pixels in the original image. If the difference between the gray level of these surrounding pixels and avg is less than σ, the position in the binary image corresponding to the patch of the neighboring pixel in the original image is set to 1.

5. Each time a new pixel position is added to the binary image, the mean and standard deviation of the pixel values that have been recorded in the binary image are recalculated.

6. Repeat the above two steps until no more neighboring pixel is added.

The result is the ROI that grows from the selected patch. The algorithm has the following features:

1. The seeds are all the pixels in the selected patch inside the ROI, which is different from using the mean value or thresholding regions as seeds in region growing and two-threshold methods.

2. Picking an ROI and inputting a patch can be done at the same time. When a ROI is selected, xy coordinates of two points of the patch (the upper left and bottom right corners) are the only input needed. This requires only one mouse click by the user; the software stores the mouse-down and mouse-up positions as the input for the determination of the patch.

3. The process is highly automatic. Selection of all patches for each image is done using the SelectObject program. The user does not need to wait for the processing of each image, but can select all ROIs first and then let all the segmentation processes run automatically.

After experimenting on several image datasets, we found that the patch growing algorithm was the most suitable for our application.

4.3.3 Median Filtering and Smoothing

Median filtering is a noise reduction technique that was implemented in CVlab using the following steps:

1. Choose a dimension for the filter window, for example, 3 pixel by 3 pixel or 5 pixel by 5 pixel. The window includes an odd number of pixel samples.

2. Sort the sample gray values in the image from smallest to largest within the window range.

3. Determine the median of the sample values and assign this value to the pixel at the center of the window, replacing the previous value

4. Repeat steps 2 and 3 and traverse the image, letting the filter window move from the top left corner to the bottom right corner.

Median filtering is applied to the image resulting from the application of the patch growing algorithm to remove noise and artifacts left in the image. In particular, some noise present in the image produced by the patch growing algorithm is salt and pepper noise, i.e., a few dots distributed throughout the image. Median filtering can reduce this type of noise efficiently. The next step in the segmentation process is Canny edge detection, and incorrect edges might be identified if the noise is not removed. The method needs a suitable choice for the filter width. For the processing of histological sections, a filter width of 5 pixels by 5 pixels was selected empirically.

4.3.4 Canny Edge Detection

This section will explain the methods called the Sobel operator, nonmaxima suppression and double thresholding. They are applied in the Canny edge detection algorithm in several steps:

1. Apply a 2D Gaussian low-pass filter (GLPF was introduced in Section 3.3.4) to smooth the image.

2. Use a Sobel operator to compute the gradient, magnitude and orientation using finite-difference approximations for the partial derivatives.

3. Apply nonmaxima suppression to the gradient magnitude to generate a thin line in the output image.

4. Detect and link edges using the double thresholding algorithm.

Sobel Operator

There are two parts to this step. The first step is to find the edge strength by estimating the gradient of the image. The Sobel operator computes a 2D spatial gradient approximation of an image. The Sobel operator estimates the gradient in the x-direction (columns) and y-direction (rows). G, the magnitude of the gradient is then approximated using the formula:

$$G = \sqrt{G_x^2 + G_y^2}, \tag{4.1}$$

where G_x is the gradient in the x direction and G_y is the gradient in the y direction.

The second step, finding the edge direction, is straightforward once the gradients in the x and y directions are known. The formula for finding the edge direction is:

$$\theta = arctan(G_y/G_x), \tag{4.2}$$

where θ is the angle of the edge direction. However, an error is generated whenever G_x is equal to zero. Whenever the magnitude of G_x is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the magnitude of G_y is. If $|G_y|$ has a value of zero, the edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. If both $|G_x|$ and $|G_y|$ have values of zero, there is no edge.

Degree	Direction
0	the horizontal direction
45	along the positive diagonal
90	the vertical direction
135	along the negative diagonal

Table 4.7: Edge directions for surrounding pixels

Nonmaximum Suppression

Assume a 5 by 5 image mask. When describing the surrounding pixels, there are only four possible directions listed in Table 4.7. For the pixels in the 180 to 360 degree range, we can subtract 180 degrees to apply the results in Table 4.7. The edge orientation has to be resolved into one of these four directions. This is done based on which direction the edge is closest to. The edge direction is set as follows:

$$\theta = \begin{cases} 0, & \text{if } 0 \leq \theta \leq 22.5 \text{ or } 157.5 \leq \theta \leq 180 \\ 45, & \text{if } 22.5 \leq \theta \leq 67.5 \\ 90, & \text{if } 67.5 \leq \theta \leq 112.5 \\ 135, & \text{if } 112.5 \leq \theta \leq 157.5 \end{cases} \tag{4.3}$$

After the edge directions are known, nonmaximum suppression is applied. Nonmaximum suppression traces along the edge in the edge direction and suppresses any pixel value (i.e., sets it equal to 0) that is not considered to be an edge. A thin line in the output image is generated.

Double Thresholding

Hysteresis is used to eliminate streaking. Hysteresis is the dependence of an edge that is currently detected on the edges already detected. Streaking is the breaking up of an edge when the Sobel operator outputs values fluctuating around the threshold

value. Why are two thresholds necessary? If a single threshold (T1) is applied to an image, and an edge has an average strength equal to T1, in some cases the edge will dip below the threshold because of noise. Similarly the edge will look like a dashed line if it extends above T1. To avoid this, two thresholds are used by hysteresis, a higher one (T1) and a lower one (T2). Any pixel is presumed to be an edge pixel and marked if its value is greater than T1. After that, any pixel which is connected to this edge pixel is also selected as an edge pixel if its value is greater than T2. In order to follow an edge, the gradient of T1 is needed to start. The process does not stop until a gradient below T2 is detected.

4.3.5 Image Loading and Saving

An image set and a CVlab script implementing the previous step needed to be uploaded into our CS server (Linux) in order to perform segmentation. In CVlab, the segmentation can be done, by running the script, in only one command.

The Rename2 program was developed to load segmentation results from CVlab, output images for the contour finding algorithm, and later for interpolation steps. It is a C# program similar to the Rename1 program. The Rename2 program is given in Appendix C.3. Advantages of these GUI programs are also discussed Appendix C.5.

4.4 The Segmentation Algorithm for CT Scans

The CT scans have already been segmented, but there are many objects in each image. The ROI still needs to be selected and the region and edge images generated.

1. Use ObjectSelect to select the object or ROI from all objects in the manual segmentation results, which were provided by the EAR-Lab [17].

2. Automatically generate a CVlab script.

3. Apply the software that implements the patch growing algorithm, to determine the regions within the image.

4. Run Canny edge detection program to identify the edges. There is no need to perform a median filtering step, because there are no outliers or noise inside the region.

5. Save region and edge image result.

4.5 The Contour Finding Algorithm for Histological Sections and CT Scans

The contour finding algorithm is expected to identify corresponding contour points (edge points representing the contour) on different images. It needs to obtain contour points in all images and create the same number of uniformly spaced angular contour points in all images using monotonic piecewise cubic interpolation. This step is intended to eliminate noise inside the ROI and identify points on the contour of the ROI. It also computes an angle attribute for each contour point. In 3D reconstruction, contour points contain 3D coordinate data, xyz. The angle attribute helps in the correlation of 3D data points, so that all 3D points that are to be interpolated have the same angle attribute.

The contour finding algorithm was designed for histological sections. It has the functionality to eliminate outliers inside segmented region images. For CT scans, the algorithm still works for region images without inside outliers. Therefore, this algorithm works on both histological sections and CT scans. It was designed and preliminarily tested on histological sections, but was also refined, improved and tested on CT scans.

4.5.1 Objective

The goal is to (i) interpolate the segmentation results, and (ii) use the augmented dataset, including given and generated data, to reconstruct a 3D model. Issues with segmentation results are: there are noise and unwanted structures inside ROIs and edge contours are rough and have different sizes, so interpolation is very challenging. Therefore, the algorithm needs to:

1. Remove noise and unwanted structures.
2. Have a closed and clear contour representing the ROI.
3. Identify points on the contour of an ROI that will be interpolation points for the interpolation algorithm.
4. Ensure that there are the same number of contour points for the ROI in each image.

There is related work on generating missing images [24]. It has a purpose similar to our research, which is to reconstruct and visualize the underlying 3D structure, and to facilitate analysis of datasets. Although this method generates images and histological sections, it still requires segmentation. Our method, on the other hand, automatically generates missing ROIs ready for 3D reconstruction.

We designed an algorithm to automatically obtain the closed contour or ROI. Edge detection methods can find all edges, but broken edges with pieces in the middle should be removed by this method. This algorithm has advantages over other edge detection methods because it generates a closed contour with points in the angular order. These contour points represent an edge in the 3D space.

In designing an algorithm that achieves this, we needed to calculate the center of the region and identify a rediating line from the center to every edge. If there are two or multiple edge points along one radiating line, only the farthest one from the ROI

center is marked as valid. Noise can also be eliminated during this process by finding the farthest point for contour points with the same angle. The noise and artifacts inside the region can be eliminated.

These contour points are used in the interpolation step to generate missing points with input z. For example, we can use contour points in 4 images (No. 1, 6, 13, and 20) to interpolate and get all points in 20 images (No. 1 to 20). Here z is the image number as well as the depth in 3D space.

4.5.2 Algorithm Description

The contour finding algorithm is described in the following.

Input: segmentation results for a dataset; these are two binary image sets; region and edge images.

Output: contour point list, saved in XML format. The list has contour points with xyz and angle values representing the ROI for each image in the data set.

Goal: Compute a closed and clear contour representing the ROI, eliminating noise and artifacts, or outliers.

Algorithm steps:

1. Load region and edge result images (output of segmentation algorithms); assign the image number to the z value of the image.

2. Calculate the geometric center for the ROI in the region image; obtain xy coordinates of all points in the edge image; compute the angle from these points to the ROI center.

3. Segmentation results are stored into a list of points with attributes x, y, z and angle.

4. For any points with the same angle, keep only the point with the maximum distance to the ROI center and remove the rest.

5. The resulting contours from Step 4 may have a different number of points for each image; this is a problem for interpolating points on different images. We decided that interpolation is needed for points on each image using uniformly spaced angle values, so that each angle has corresponding points in each image. Interpolation can be then applied to these corresponding points. Therefore, the angle increment ($\triangle\theta$) is required. It is found by averaging all angle differences (θ_{diff}) between each two neighboring points:

$$\triangle\theta = \frac{1}{Q_{max}} \sum_{i=0}^{Q_{max}-1} \theta_{diff,i}, \tag{4.4}$$

where Q_{max} is the number of points in the image contour that has the largest number of points. The resulting contour point number N is given by

$$N = \frac{360}{\triangle\theta}. \tag{4.5}$$

6. Using the $\triangle\theta$ value, the angle array for points in every slice is:

$$0, \triangle\theta, 2\triangle\theta, \cdots, (N-1)\triangle\theta \tag{4.6}$$

Apply monotonic piecewise cubic interpolation to resulting points from Step 4 on each image using this angle array; obtain resulting contour points.

The final result of the contour finding algorithm is the set of contour points. There are the same number of contour points for the ROI in each image. Later steps (filtering and interpolation) can be done on every set of these contour points with the same angle on different images in the dataset (this is also called the z direction).

This solves the issue of filtering and interpolation for the edges that vary in size and shape. The value of $\triangle\theta$ is calculated automatically by the algorithm. The contour point number is parametric and depends on the image set.

To summarize, the contour finding algorithm has a creative design. It generates the same number of contour points for each image using segmentation results, and creates a mapping of these points on different images correspondingly using the angle attribute. It provides connectivity between segmentation results and Gaussian filtering and interpolation methods.

4.6 A Gaussian Filtering Algorithm for CT Scans

This section describes the Gaussian filtering algorithm that is applied to smooth the contour points obtained from Section 4.5. It applies both the theory and methods introduced in Section 3.3 in order to use the FFT algorithm and build GLPFs with appropriate parameters. It presents an algorithm for building two 1D GLPFs and applying them to 3D contour points.

The Gaussian filtering algorithm is designed for CT scans. In future work, this algorithm can be adjusted to work on histological sections. The idea would be the same; however, the change needed would involve implementing the Fourier transform for non-uniformly spaced data from histological sections.

4.6.1 Gaussian Filtering Overview

Gaussian filtering for both image denoising and 1D signal smoothing was introduced in 3.3.4. The Gaussian filtering in this section is a 1D filter that is applied to smooth contour point data. The algorithm is designed to smooth 3D contour points using two 1D GLPFs, one of which smooths the contour points on each slice, while the other smooths data in the z direction.

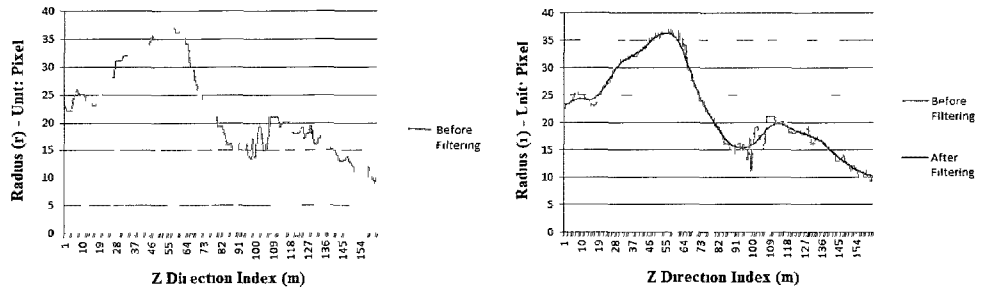

Figure 4.1: Example data smoothed using Gaussian filtering (CT Incus region, 162 slices, contour points at 180 degrees). From left to right (a-b): a. Before filtering data; b. Before and after filtering data

Filtering is an important step since interpolation works better on smooth data. In the initial experiments, simple filtering methods, e.g., mean filter and median filter, did not give satisfactory results. In theory, the signal processing methods in the frequency domain would work much better according to the characteristics of the data. When data are transformed into the frequency domain, the high frequency data represent details, small changes, or spikes, while the low frequency part of the data represents the outline shape and global changes. Then, a Gaussian low-pass filter with appropriate parameters can be applied to remove high frequency data, or spikes, so that the results are smoother. Figure 4.1 demonstrates the effect of filtering.

We need to examine the data and determine frequency spectrum, find the appropriate σ value of the GLPF, and apply the GLPF.

There are two equivalent ways to apply the GLPF:

1. Multiply the signal by the GLPF in the frequency domain and transform the filtered result back to the spatial domain.

2. Perform a convolution using the spatial form of the GLPF.

We chose the second option, which means that there was no need to use an inverse Fourier transform. In order to use this second option, we need to understand the relationship between the GLPFs in the frequency and spatial domains. More

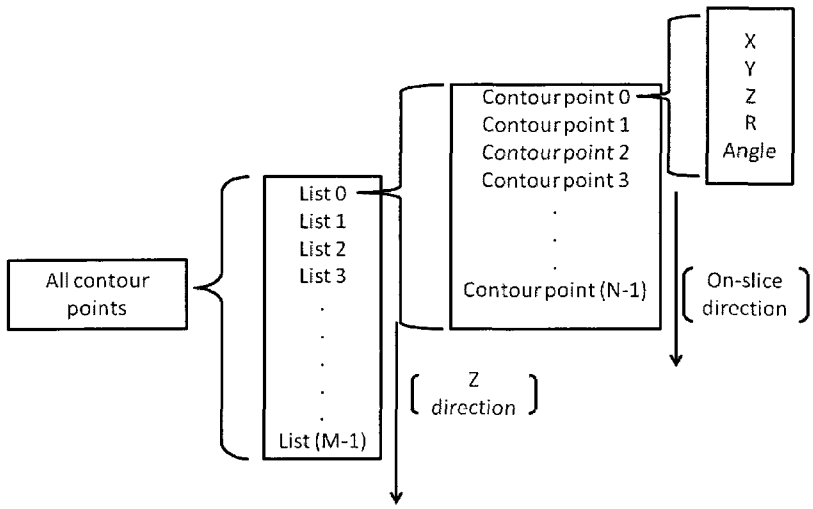

Figure 4.2: The structure of all contour points.

importantly, after determining the standard deviation (σ_f) in the frequency domain, the standard deviation (σ_s) in the spatial domain needs to be calculated.

4.6.2 The Gaussian Filtering Algorithm Description

This algorithm is designed to filter and smooth the contour points, and remove spikes and outliers. It takes a contour point list as input, and outputs a filtered list of contour points and filtering errors.

The structure of contour points is shown in Figure 4.2. They are represented by M lists. Each of these lists is obtained from a ROI of an image at a certain z location in the 3D space. For each ROI corresponding to a given z value, there are N contour points. Figure 4.3 shows a sketch of sample on-slice contour points. Figure 4.4 is a sketch of sample z-direction contour points.

The idea is to process the on-slice contour points in the frequency domain, by

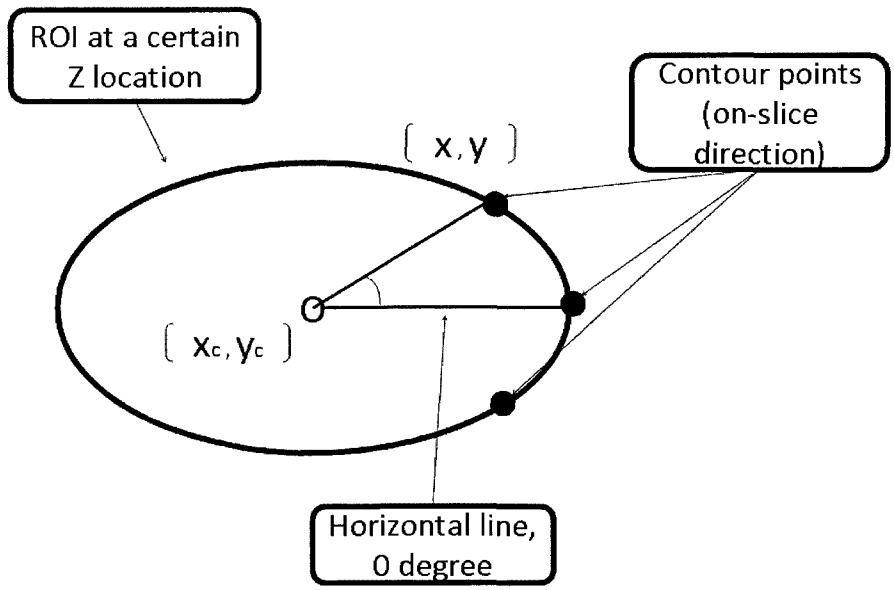

Figure 4.3: Schematic diagram of on-slice contour points.

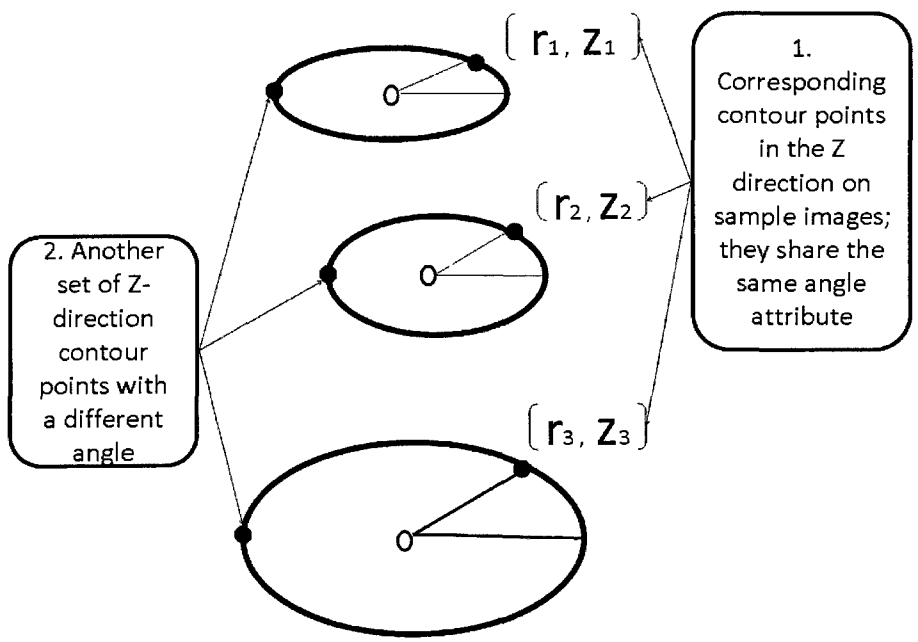

Figure 4.4: Schematic diagram of z-direction contour points.

building and applying a GLPF in the on-slice direction. Then, we use the same approach to develop another GLPF and filter z-direction contour points. This way the 3D model represented by 3D points with angle attributes can be smoothed automatically. The algorithm steps are as follows:

1. In on-slice filtering, contour points have the same z value. Each is represented using a radius (r) and an angle (θ). r is the distance from contour points to the contour center. The list of contour points on a given slice can be viewed as a 1D signal, $f(x)$.

2. Automatically develop a spatial GLPF for $f(x)$ using a frequency analysis and filtering method. As this step is the most important in the algorithm, it will be further detailed in Section 4.6.3, which is to define σ and the filter width to build the GLPF based on the average power spectrum of the signal in the frequency domain.

3. Apply convolution using the spatial GLPF to the radii values (r) (this does the actual smoothing). Smoothing the r values will change the xy coordinates.

4. Apply the same type of Gaussian filtering to the contour point list for the z direction.

5. Measure differences or errors generated from automatic Gaussian filtering both in on-slice and z-direction.

4.6.3 Gaussian Filtering Development

The above algorithm presents the overall logic and steps. As well, several key points are described, in particular how σ is computed, how GLPFs are built, and how the filter width is determined. The only input of the program is the percentage of the signal power to keep, e.g., $\alpha = 95\%$.

The theory described in Section 3.3.4 is applied in the subroutine to compute σ_f (σ in the frequency domain), and its related σ_s (σ in the spatial domain). The CT dataset and its incus ROI is used to walk through this subroutine. Two major and useful conclusions from the analysis and experiments in this work are:

$$D_0 = \mathcal{F}_{cutoff} = \sigma_f \tag{4.7}$$

$$\sigma_s = \sigma_f \frac{1}{N}. \tag{4.8}$$

The development steps of Gaussian filtering are as follows:

1. There are M slices in the dataset and N contour points on each slice. Contour points on each slice can be represented by xy coordinates or radii, r, and angle, θ. As $\triangle\theta$ was previously computed and fixed in Equation (4.4), there is:

$$\theta = \theta_0 + 0\triangle\theta, 1\triangle\theta, 2\triangle\theta, \cdots, (N-1)\triangle\theta \tag{4.9}$$

Equation (4.9) is equivalent to:

$$\theta = \theta_0 + n\triangle\theta \tag{4.10}$$

where $n = 0, 1, 2, \cdots, N-1$, and θ_0 is set to 0 degrees. The value N is the number of contour points on each slice, defined by Equation (4.5). These N contour points can be simplified as a 1D signal with discrete r values, which are represented by function $f_s(n)$:

$$r_n = f_s(n) = f_s(\theta_0 + n\triangle\theta) \tag{4.11}$$

The value of $f_s(n)$ is the r value at a certain n location on slice.

2. Contour points in one slice have corresponding contour points in all other slices, with the same θ value. That is, each θ angle has M points in the z direction. Contour points at each θ can be represented by radii r and distances z. The function of z can be expressed as

$$z = z_0 + 0\triangle z, 1\triangle z, 2\triangle z, \cdots, (M-1)\triangle z. \tag{4.12}$$

Equation (4.12) is equivalent to

$$z = z_0 + m\triangle z, \tag{4.13}$$

where $m = 0, 1, 2, \cdots, M-1$, z_0 is the label number of the first image, and M is the number of images. $\triangle z$ is a fixed physical distance between two neighbor points and depends on the data. These M contour points can be simplified as another 1D signal with discrete r values, which are represented by function $f_z(m)$:

$$r_m = f_z(m) = f_z(z_0 + m\triangle z). \tag{4.14}$$

Similar to the on slice function, $\triangle z$ is the distance between each two neighbor contour points in the z direction or the distance between two neighbor images. The value of $f_z(m)$ is the r value at a certain m location in the z direction. Obviously, Equations (4.11) and (4.14) have the same form. Gaussian functions developed for both directions would share the same method in the following.

3. There are M $f_s(n)$ functions and N $f_z(m)$ functions. These functions are analyzed and filtered in two steps. On slice filtering is done at first.

4. M $f_s(n)$ functions are normalized, which is to find the mean of the function and subtract it from each value so that the function has a mean of zero.

5. FFT is applied to each function $f_s(n)$ using Equation (3.12):

$$F_s(u) = \mathbb{FFT}\{f_s(n)\}. \tag{4.15}$$

Each is an array of complex numbers with real and imaginary parts. The FFT routine was implemented by modifying a 100-line code (approximately) that was found in pudn.com, a Chinese programming source community. It was tested and compared with FFT in Matlab. Its results and Matlab results match completed, which shows that the FFT routine works correctly.

6. The cumulative spectrum and percentage of total power under the cumulative spectrum is calculated. A cutoff size between 95% and 99.7% needs to be chosen.

7. A power spectrum is computed using Equation (3.23) for each $F_s(n)$:

$$P_s(u) = |F_s(u)|^2 = R_s^2(u) + I_s^2(u). \tag{4.16}$$

For M slices, contour points on each slice would have a power spectrum $P_s(u)$.

8. The on-slice averaged power spectrum $P_{s,avg}(u)$ is then computed by averaging M power spectrums, $P_{s,m}(u)$:

$$P_{s,avg}(u) = (\sum_{m=0}^{M-1} P_{s,m}(u))/M. \tag{4.17}$$

For each frequency u, an average power value is calculated. This average is calculated across all slices (in the z direction).

9. The total on-slice signal power $P_{s,T}$ is computed by summing all the average

power values $P_{s,avg}(u)$, across all the frequencies, u:

$$P_{s,T} = \sum_{u=0}^{N-1} P_{s,avg}(u). \tag{4.18}$$

10. The $P_{s,avg}(u)$ function is then shifted by moving the zero-frequency component to the center of the function. Because $P_{s,avg}(u)$ is symmetric, dividing it into two equal intervals, left and right, the total signal power of each of the left and right portion is $P_{s,T}/2$.

11. The suitable cutoff frequency, $\mathcal{F}_{cutoff}$, is a distance D_0 [25] from the Fourier origin. It is located by considering the fraction (α) of the total power spectrum, $P_{s,T}$, that we wish to maintain. Using the three-σ rule (Section 3.3.4), about 95% of the power is contained within two standard deviations ($2\sigma'$) of the mean, and 99.7% of the power is contained within three standard deviations ($3\sigma'$). Empirically it was determined that maintaining 95% of the power spectrum ($2\sigma'$) causes too much smoothing/blurring, and maintaining 98% does satisfactory smoothing. Therefore, 98% was used. For example, when summing power from the center to the right of $P_{s,avg}(u)$, if the at $u = 5$, the summed power hits α of the $P_{s,T}/2$, $\mathcal{F}_{cutoff}$ is 5.

12. The GLPF in the frequency domain can be built using Equation (3.26) and the relationship that was worked out using theory in Section 3.3.4:

$$D_0 = \mathcal{F}_{cutoff} = \sigma_f. \tag{4.19}$$

which is a key point that connects pieces of theory to the method, and gives this algorithm a seamless design. It can be interpreted as: The D_0 value determined by computing α of the signal power is useful to define the cutoff frequency,

which is also the suitable σ input to build a GLPF in the frequency domain to smooth the particular signal.

13. Based on Shannon sampling theorem in Section 3.3.3, the sampling rate, $\mathcal{F}_s$, needs to be checked and verified using:

$$\mathcal{F}_s \geq 2\mathcal{F}_{cutoff}. \tag{4.20}$$

According to Equation (4.19), it is also indicated that σ_f is valid if it is less than half of the sampling rate:

$$\sigma_f < \mathcal{F}_s/2. \tag{4.21}$$

14. Applying the frequency filter would require the inverse Fourier transform to convert the filtered results back to the spatial domain. Its drawback is that the inverse Fourier transform needs a lot of computing time for a large amount of data. The alternative is to build the equivalent GLPF in the spatial domain, and apply convolution on the spatial 1D signal, $f_s(n)$.

15. In theory, building the equivalent GLPF using its frequency filter is possible, but not an easy problem. As in Equation (3.17), σ (specifically σ_s) needs to be determined. The problem is to compute σ_s using σ_f.

16. σ_s and σ_f are related, but an equation between them needs to be figured out. They are corresponding variables in the spatial and frequency domains. The relationship between samples in these two domains was introduced in Equations (3.9), (3.10) and (3.11).

17. σ_f is already known, which represents a length in $F(u)$. Working out the actual length in $f(x)$ would give the σ_s value. Then, the length σ_s is σ_f units or

increments of $F(u)$ in u-axis. The increment is $\triangle u$. This analysis concludes the following:

$$\sigma_s = \sigma_f \triangle u = \sigma_f \frac{1}{N \triangle x}. \tag{4.22}$$

In the FFT routine applied, the unit of $\triangle u$ is "radians per N samples". $\triangle x$ is the x-axis increment in $f(x)$, so $\triangle x = 1$. Therefore,

$$\sigma_s = \sigma_f \frac{1}{N}. \tag{4.23}$$

18. This seems too simple to be true. As well, the spatial GLPF built this way in the experiments seems too wide. In order to confirm this statement logically, various experiments were done. One way to examine and verify Equation (4.23) is to see if the GLPFs built using σ_f and its corresponding σ_s match.

19. The GLPF function pair in the test is:

$$H_{Gaussian}(u) = Ae^{\frac{-u^2}{2\sigma_f^2}} \cdots \text{(Frequency domain form)} \tag{4.24}$$

$$h_{Gaussian}(x) = \sqrt{2\pi}\sigma_s A e^{-2\pi^2\sigma_s^2 x^2} \cdots \text{(Spatial domain form)} \tag{4.25}$$

Applying FFT to $h_{Gaussian}(x)$ would give a $H'_{Gaussian}(u)$ function:

$$H'_{Gaussian}(u) = \mathbb{FFT}\{h_{Gaussian}(x)\}. \tag{4.26}$$

This experiment shows:

$$H'_{Gaussian}(u) = H_{Gaussian}(u), \tag{4.27}$$

where values in both functions are equal. A complete CT dataset was used in this experiment. For the on-slice direction, all GLPFs in this test are plotted in

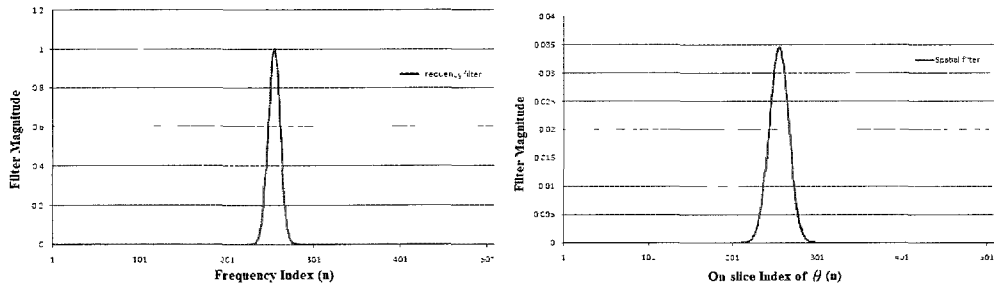

Figure 4.5: The relationship between σ_f and σ_s (on-slice direction). From left to right (a-b): a. The frequency (Gaussian) filter is built using σ_f; b. The spatial filter is built using σ_s based on Equation (4.23).

Figures 4.5 and 4.6. The z direction GLPFs are shown in Figures 4.7 and 4.8.

20. $h_{Gaussian}(x)$ is the GLPF built for applying convolution on the contour points. The filter width of the spatial GLPF (on-slice), W_s, needs to be defined, because the function needs to be cropped to a certain width range. (Similarly, the filter width of a spatial GLPF in the z direction is noted as W_z.) If some values in the GLPF $h_{Gaussian}(x)$ are too small to be effective, there is no need to keep them, although in theory every point of the Gaussian function is non-zero. Therefore, a rule should be used to remove effectively-zero values from the GLPF. Then, the remaining number of values would be the width. In practice, values outside the 3-σ' range of $H_{Gaussian}(u)$ can be considered zero, according to the three-σ rule of Gaussian distribution. To achieve the 3-σ' range is to keep 99.7% of the values from the GLPF center/origin.

21. As σ_s and W_s are determined, the spatial GLPF can be properly built. This filter is for on-slice smoothing.

The development method of the spatial GLPF for filtering on-slice contour points was described. The number of samples, N, is how many angles in the contour points. The method of developing the spatial GLPF for z-direction filtering is similar. The input list of contour point data are transformed by Fast Fourier Transform. The

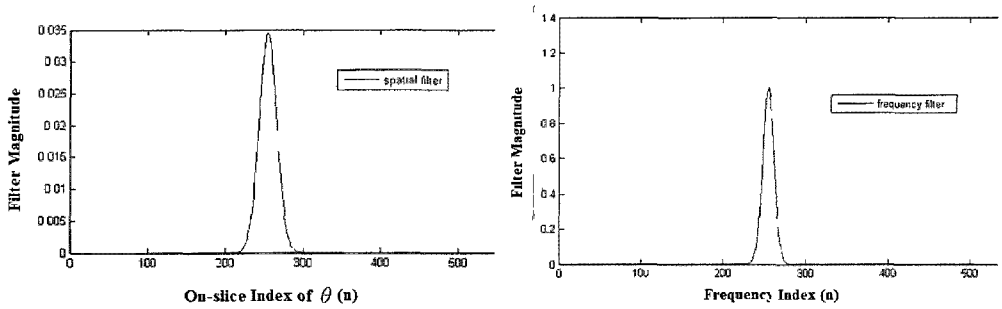

Figure 4.6: The Matlab test confirms σ_f and σ_s relationships (on-slice direction). From left to right (a-b): a. The spatial filter in 4.5.b is imported into Matlab; b. Such spatial filter in (a) is then transformed to the frequency domain using Matlab's FFT, which shows the resulting frequency filter is the same as Figure 4.5.a.

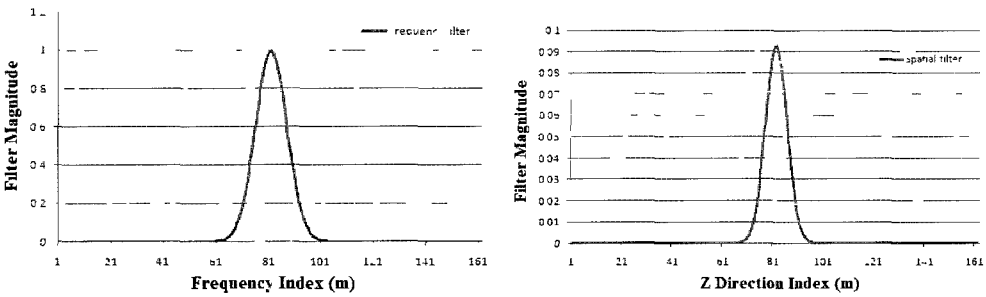

Figure 4.7: The relationship between σ_f and σ_s (z direction). From left to right (a-b): a. The frequency (Gaussian) filter is built using σ_f; b. The spatial filter is built using σ_s based on Equation (4.23).

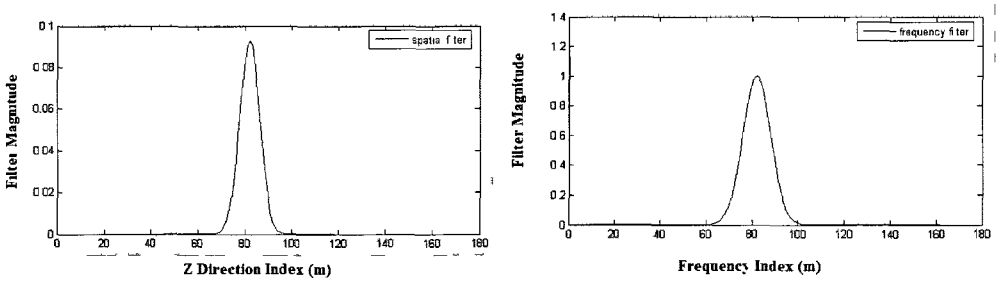

Figure 4.8: The Matlab test confirms σ_f and σ_s relationships (z direction). From left to right (a-b): a. The spatial filter in 4.7.b is imported into Matlab; b. Such spatial filter in (a) is then transformed to the frequency domain using Matlab's FFT, which shows the resulting frequency filter is the same as 4.7.a.

power spectrum, α of signal power, and the cutoff frequency are computed with three-σ rule being applied. As σ_s and W_s is determined, the GLPF can be fully defined in z-direction. In this case, M is the number of images in the dataset.

4.7 Error Analysis for CT Scans

The sources of error are analyzed in this section for CT scans. The goal of this section is to determine the approximate data error E_{approx}. Such data error serves as important input for the adaptive interpolation algorithm (refer to Section 4.8). The data points are not perfect due to errors introduced in data collection and data processing steps. The errors of these data points need to be analyzed so that the performance of the adaptive interpolation algorithm can be measured. They need to be estimated, defined, and provided as basis for the adaptive interpolation algorithm.

This section presents error analysis methods for segmentation, contour finding and filtering methods. It starts with an overview and ends with error combination. For all three errors, it uses the $1 - S$ (one minus similarity index) method. The filtering error is measured by one more method, which measures changes of radii between contour points and the ROI center. This method is called the $\triangle r$ method.

4.7.1 Errors Overview

Table 4.8 describes what the main errors are. It shows how these errors can be estimated or accurately measured. Figure 4.9 shows the overview diagram of the error analysis. The analysis methods of these errors are described in Sections 4.7.2, 4.7.3, 4.7.4, and 4.7.5. The results of error analysis are presented in Section 5.2.2.

Aside from these main errors in data acquisition, CT scan is relatively accurate. Pixel size is given in CT header and is known, but this information was not available because the original CT scans in DICOM were not provided. Instead, JPG images

Number	Method name	CT error calculable
1	Image segmentation	Not measurable but can be estimated. The expert's manual segmentation results are compared with 3 manual tests using 1 minus similarity index ($1 - S$ method), in Equation (4.29), resulting $E_{segment}$.
2	Contour finding	Measurable using $1 - S$ on before and after images, resulting $E_{contourFind}$.
3	Gaussian filtering	Can be measured in two methods, $1 - S$ and $\triangle r$; the $1 - S$ method measure the difference in the regions, resulting $E_{filtering1}$, while the $\triangle r$ method accurately measures the radius changes of contour points, resulting $E_{filtering2}$.

Table 4.8: Summary of error categories and analysis methods

are given. Due to the digitizing process, the image data have integer pixel locations (rounding generates certain error); the error is estimated to be less than 0.5 pixels. CT images do not need to be registered, so there is no need to analyze registration errors for CT.

4.7.2 Segmentation Error Analysis

The segmentation algorithm can be measured by similarity index, S. It measures how similar two regions, ROI_1 and ROI_2 are:

$$S = 2 * \frac{|ROI_1 \cap ROI_2|}{|ROI_1| + |ROI_2|} \tag{4.28}$$

If ROI_1 and ROI_2 are identical, S is 1. The error then generated from the algorithm is how much the two regions are dissimilar. The error E_m on one image at

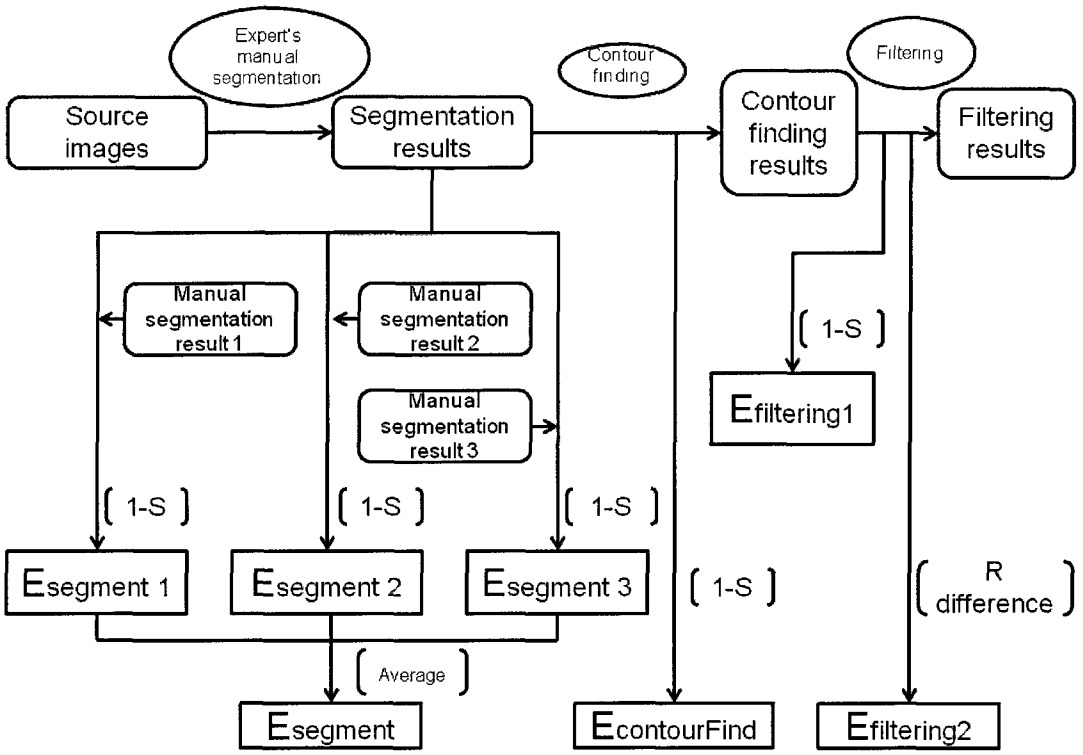

Figure 4.9: Error analysis overview diagram.

location z ($z = z_0 + m\triangle z$; index of m) is given by:

$$E_m = 1 - S_m = 1 - 2 * \frac{|ROI_{m,1} \cap ROI_{m,2}|}{|ROI_{m,1}| + |ROI_{m,2}|} \tag{4.29}$$

where S_m, $ROI_{m,1}$ and $ROI_{m,2}$ are the similarity index and two regions at the location index of m respectively. The error averaged over all z locations is given by:

$$E = \frac{1}{M} \sum_{m=0}^{M-1} E_m. \tag{4.30}$$

For CT images, segmentation results were done manually by the Research Associate at the EAR-Lab [17]. These segmentations, completed by an expert, were considered the gold standard for the purposes of this research. In order to have rough estimates of segmentation errors using statistical methods, we decided to perform manual segmentation using Amira on the CT complete dataset three times: twice by the author of this thesis, and once by a third party tester. The segmentation error is estimated by comparing those three sets of segmentation results to the results of the expert, as shown in Figure 4.9. From these comparisons and using Equations (4.29) and (4.30), three lists/arrays were obtained; $E_{segment1}$, $E_{segment2}$ and $E_{segment3}$. The error result data for both the incus and the malleus regions in CT images are presented in Section 5.2.2.

4.7.3 Contour Finding Error Analysis

The success of the contour finding algorithm can be measured by comparing regions obtained before and after this step. On one image at the location index of m, the before region is the segmentation result, $ROI_{m,segment}$, while the after region is the

contour finding result, $ROI_{m,contourFind}$. The error $E_{m,contourfind}$ is given by:

$$E_{m,contourfind} = 1 - S_{m,contourfind} = 1 - 2 * \frac{|ROI_{m,segment} \cap ROI_{m,contourFind}|}{|ROI_{m,segment}| + |ROI_{m,contourFind}|}. \quad (4.31)$$

The error averaged over all z locations is given by:

$$E_{contourfind} = \frac{1}{M} \sum_{m=0}^{M-1} E_{m,contourfind}. \quad (4.32)$$

Error data for the incus and the malleus regions in CT images are in Section 5.2.2.

4.7.4 Filtering Error Analysis

Filtering error can be computed in two ways, resulting in $E_{filtering1}$ and $E_{filtering2}$ respectively.

The first method is called $1 - S$ method, which stands for one minus similarity index. It measures the filtering error, $E_{filtering1}$, by comparing the regions obtained before and after this step. On one image at the location index of m, the before region is the contour finding result, $ROI_{m,contourFind}$, while the after region is the filtering result, $ROI_{m,filtering}$. The error $E_{m,filtering1}$ is given by:

$$E_{m,filtering1} = 1 - S_{m,filtering1} = 1 - 2 * \frac{|ROI_{m,contourFind} \cap ROI_{m,filtering}|}{|ROI_{m,contourFind}| + |ROI_{m,filtering}|} \quad (4.33)$$

The error averaged over all z locations is given by:

$$E_{filtering1} = \frac{1}{M} \sum_{m=0}^{M-1} E_{m,filtering1}. \quad (4.34)$$

The $\triangle r$ method for obtaining $E_{filtering2}$ is computed differently. The radii, r, of all contour points in all images are known before and after filtering. These contour points are on M images, each of which has N of them. Each contour point has a r

value. Filtering has two steps, on slice and in the z direction, the results of which are $r\prime$ and $r\prime\prime$ respectively. Therefore, the r of a contour point corresponds to an error on slice ($E_{filteringS}$), an error in the z direction ($E_{filteringZ}$), and one more overall error ($E_{filtering2}$). They are given by three equations:

$$E_{filteringS,m,n} = \frac{|r\prime_{m,n} - r_{m,n}|}{r_{m,n}} \tag{4.35}$$

$$E_{filteringZ,m,n} = \frac{|r\prime\prime_{m,n} - r\prime_{m,n}|}{r\prime_{m,n}} \tag{4.36}$$

$$E_{filtering2,m,n} = \frac{|r\prime\prime_{m,n} - r_{m,n}|}{r_{m,n}} \tag{4.37}$$

where $n = 0, 1, 2, \cdots, N-1$, and $m = 0, 1, 2, \cdots, M-1$, according to Section 4.6.3. The structure of $E_{filtering2,m,n}$ and $r_{m,n}$ are 2D arrays, with $M \times N$ values, much the same as there are $M \times N$ contour points. Each contour point corresponds to a $r_{m,n}$ value, as well as a $E_{filtering2,m,n}$ value.

$E_{filtering2}$ is the average of $E_{filtering2,m,n}$ and is given by:

$$E_{filtering2} = \frac{1}{NM} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} E_{filtering2,m,n} \tag{4.38}$$

$E_{filteringS}$ and $E_{filteringZ}$ are averages computed using an equation of the same form as Equation (4.38).

The sample error data for both the incus and the malleus regions in CT images for both $1 - S$ and $\triangle r$ methods are in Section 5.2.2.

Errors of contour finding and filtering can be measured separately, noted as $E_{contourfind}$ and $E_{filtering1}$. They also can be measured using one equation, between $ROI_{m,segment}$ and $ROI_{m,filtering}$. The error generated in these two steps, $E_{m,contourfiltering}$, on one image at the location index of m is given by:

$$E_{m,contourfiltering} = 1 - S_{m,contourfiltering} = 1 - 2 * \frac{|ROI_{m,segment} \cap ROI_{m,filtering}|}{|ROI_{m,segment}| + |ROI_{m,filtering}|} \tag{4.39}$$

The error averaged over all z locations is given by:

$$E_{contourfiltering} = \frac{1}{M} \sum_{m=0}^{M-1} E_{m,contourfiltering}. \tag{4.40}$$

Such results are given in Section 5.2.2.

4.7.5 Error Combination

Each step in the algorithm results in a different error measurement. We use the term "error" in this thesis to refer to lack of consistency. A key point is how the errors are combined to obtain an overall error, E_{approx} , that can be used as input to the adaptive interpolation algorithm. In determining how to combine the several sources of error, three methods were considered. Assuming that there are three error sources, E_1, E_2 and E_3, the three methods considered are described here.

1. The absolute values of the individual errors can be added to form an overall error [80].

$$E_{overall} = |E_1| + |E_2| + |E_3| \tag{4.41}$$

2. It is unlikely that all errors are in the same direction, although it is possible. The overall error of multiple errors is usually larger than any of the individual error, and smaller than their sum. Therefore, a statistical treatment can give a more conservative estimate [80] [10]:

$$E_{overall} = \sqrt{E_1^2 + E_2^2 + E_3^2} \tag{4.42}$$

3. If one error has a magnitude far larger than all other errors, it is considered the dominant error. Then, $E_{overall}$ could be the maximum of all errors. The dominant errors need to be found in the experiment and should be reduced as much as possible [10].

Equation (4.42) was applied to the CT error analysis to obtain an overall error measure, $E_{overall}$, with the three error sources being $E_{filtering1}$, $E_{segment}$ and $E_{contourfind}$:

$$E_{overall} = \sqrt{E_{segment}^2 + E_{contourfind}^2 + E_{filtering1}^2}. \tag{4.43}$$

Errors analyzed for CT incus and malleus are summarized in Table 5.3. As discussed at the end of Section 5.2.2, the filtering error $E_{filtering2}$ is assigned to E_{approx} and passed to the adaptive interpolation algorithm. The error basis for the adaptive interpolation algorithm has been defined as

$$E_{approx} = E_{filtering2}. \tag{4.44}$$

4.8 Adaptive Interpolation Algorithm for CT Scans

This section presents experimental adaptive interpolation algorithms for CT scans. An important question is which interpolation technique is most suitable for our application, and how should it be applied to produce accurate results, while using a small amount of data. We describe why monotonic piecewise cubics were chosen and how they are applied.

4.8.1 Spline Interpolation Experiments

This section presents experiments on spline interpolation techniques that were performed in order to choose a suitable interpolation method for our application. Our original investigation considered linear splines, as well as natural and clamped cubic splines. However we found that none of the above gave satisfactory results. We found that monotonic piecewise cubic interpolation led to smooth and accurate results when applied to the test data. Our research thus focused on monotonic piecewise cubic interpolation.

Linear interpolation involves the use of line segments between every two data points. However, bone structures in our image data usually have smooth surfaces. Thus, interpolation with polynomial of higher degree should be more suitable. Polynomial and spline interpolation (introduced in Section 3.4) are two other options. However, polynomial interpolation can have oscillatory behavior for large numbers of data points. Cubic splines are piecewise cubic polynomials. Because cubic splines are continuous and smooth, we felt they might be reasonable candidates and we therefore conducted some experiments using them.

The CT image dataset had uniformly spaced data points, so we implemented spline interpolation that assumes uniformly spaced data points. However, since histological sections have non-uniformly spaced data images, the algorithm needs a more general form can handle non-uniformly spaced points.

In the implementation of cubic spline interpolation, a number of code resources were found. A C++ implementation, by Moreau [51], was ported to our C# program. Moreau's code was based on Fortran 77 source from Tuan Dang Trong's Numath Library and Forsythe's book [19].

Both natural cubic splines and clamped cubic splines were tested. A natural cubic spline has a starting and ending slope set to be 0, while the clamped cubic spline sets

the starting and ending slopes. In the experiment, natural cubic splines resulted in good approximations in most data intervals, but did not work well in certain other intervals. Clamped cubic splines generated a better fitting function, especially at the beginning and the end of the interval, where the slopes are controlled, but oscillations still existed in the small intervals where there were rapid changes in the data.

A summary of our results on cubic splines will be presented in Section 5.2.3. We drew the conclusion that cubic splines are not suitable for our research. A better and more suitable approach, monotonic piecewise cubic interpolation, was then considered. Monotonic piecewise cubics are very similar to cubic splines, but on each interval the cubic is either monotonically increasing or decreasing between each pair of data points.

According to the theory in Section 3.4.4, monotonic piecewise cubic interpolation can produce smooth and visually pleasing interpolants. Figure 5.28 gives an example where cubic splines and monotonic piecewise cubics were compared using actual CT sample data (refer to Section 5.2.3). It shows that the monotonic piecewise cubic deals better with the intervals in which there is a significant change in the data values. In our experiments, we observed that cubic splines work well where the data change slowly, but worse than monotonic piecewise cubics when the data changes abruptly.

4.8.2 Monotonic Piecewise Cubic Implementation

The theory behind monotonic piecewise cubics is described in Section 3.4.4. Its implementation is available in a software package, Piecewise Cubic Hermite Interpolation Package (PCHIP), designed by Fritsch and Carlson [23]. A routine, PCHEZ, generates a smooth piecewise cubic interpolant that is visually pleasing. Another routine, PCHEV, evaluates the interpolant and its derivative.

Unlike cubic splines for which there are a large number of available implemen-

tations, it was difficult to find suitable implementation of the monotonic piecewise cubic. PCHIP is available in Matlab and in Fortran in Kahamer's book [38], both of which are difficult to port to C#. Some other C++, python and Fortran programs were analyzed as listed in Table D.1 in Appendix D. In the end, a C++ version in an open source program, called Hugin (hugin.sourceforge.net), was ported successfully into our "Reconstruction" program in Windows/C# .NET (refer to Table 4.2). Hugin is a panorama photo stitcher, a popular and large software application.

The C++ PCHIP source was compiled to be dynamic-link libraries (DLL), which are shared libraries of functions in executable files, using Visual Studio .NET so that the routines can be called in C#. Unit tests for the program verified that the program was correctly implemented, and that the interpolation results were accurate. An actual dataset sample (CT images, incus, No. 36 to No. 66) was tested and results are presented in Appendix D. Both Figure D.1 and Table D.2 show that interpolated values of Matlab and PCHIP DLLs in the C# "Reconstruction" program match well.

4.8.3 The Adaptive Interpolation Algorithm Description

Section 3.4.5 introduced the adaptive algorithm interpolation, why it is needed, a brief description of the algorithm steps, and Equation (3.34). In this section,we describe how the theory and equations in Section 3.4.5 are applied and implemented. The adaptive interpolation algorithm is implemented in C#.

Input: filtered contour point list and E_{approx} (results of Sections 4.6 and 4.7.

Output: Interpolants, interpolated images (entire set) and data for analysis

Goal: Implement an adaptive interpolation algorithm based on monotonic piecewise cubics. Perform a smooth interpolation using monotonic piecewise cubics, apply optimization to use a small number of data points to obtain good approximations, and generate intermediate images for the 3D reconstruction of a virtual model.

The structure of contour points was introduced in Figure 4.2. Figures 4.3 and 4.4 illustrate on-slice and z-direction contour points. The adaptive interpolation algorithm is applied on contour points in the z direction.

Each of the contour points is represented by an (x, y) pair. These N contour points share the same ROI center (x_c, y_c), and z value. The angle θ and radius r from each contour point to the center can be computed:

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{4.45}$$

$$\theta = \arccos\left(\frac{x - x_c}{r}\right) \tag{4.46}$$

Then, each contour point has attributes, x, y, z, θ, r, and is associated with the ROI center (x_c, y_c).

The approximate data error, E_{approx} (calculated in Equation (4.44)), is represented by a similar structure except that each entry is an error value associated with the contour point. In each angle from the ROI center, E_{approx} is an array with filtering error values in the z direction, perpendicular to the image plane. Figure 5.18 shows an example; the E_{approx} values are larger in some places, but smaller in other places.

The interpolation function of monotonic piecewise cubics is applied along the z direction, as in Figure 4.10. For a certain θ angle, there are M corresponding contour points, one from each image ROI. The r values of these points are used to build an interpolant. Then, new r values are generated from the resulting interpolant at desired z values for a given angle θ. With θ, z and r defined, the other two attributes of the contour point, x and y, can be calculated:

$$x = x_c + r\cos(\theta) \tag{4.47}$$

$$y = y_c + r\sin(\theta) \tag{4.48}$$

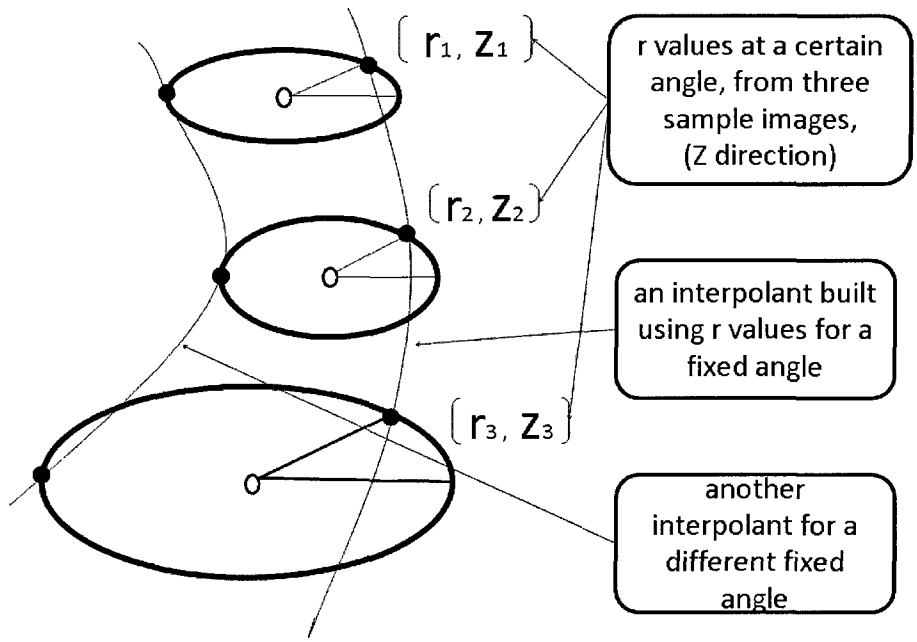

Figure 4.10: The structure of all contour points and interpolants.

One interpolant is required for each of the N angles. The interpolation method traverses all θ values, so that the corresponding contour points with the same θ are interpolated in the z direction.

Each interpolation function uses some fraction of the available M contour points in the z direction. The approximate data error, E_{approx}, is used to guide the interpolation error. It is better if the error array of E_{approx} for each angle can be applied locally so that the adaptive interpolation algorithm allows smaller errors in places where the data error is smaller, and allows larger errors for places with larger data error. Therefore, the array E_{approx} for each angle was averaged in each interval, instead of being averaged globally.

Two terms used for the adaptive algorithm are:

1. Knot: a contour point that is provided to the interpolation algorithm.

2. Interval: the region between two adjacent knots.

For each θ, the adaptive algorithm performs the following steps (based on Section 3.4.5). X_1 is a variable, but is set to 16 for this explanation.

1. Build a piecewise monotonic $Interpolant_i$ starting with coarse data, where i represents the interpolant level and starts at $i = 1$. Every X_1th (e.g., $X_1 = 16$) data point is used.

2. Build another piecewise monotonic $Interpolant_{i+1}$ with twice as many as data points of the previous interpolant (or every 8th point), $Interpolant_i$.

3. Compare $Interpolant_i$ and $Interpolant_{i+1}$ on each interval of $Interpolant_{i+1}$. In each interval, we would like to have the maximum of the interpolation errors be no larger than some fraction, β, of the average of the corresponding E_{approx} values. The comparison that is performed on each interval is given by:

$$\max_{t=1}^{T_k-1}\left(\frac{|Interpolant_{i+1}(z) - Interpolant_i(z)|}{|Interpolant_i(z)|}\right) \leq \frac{\beta}{T_k - 1}\sum_{t=1}^{T_k-1} E_{approx}(z) \tag{4.49}$$

where i represents the interpolant level, t is the index of the contour point in the current interval, k is the interval index, T_k is the number of contour points in the current interval. The value of $Interpolant_i(z)$ is the radius (r) at a particular θ and z. The z value can be expressed as

$$z = z_{k,0} + t, \tag{4.50}$$

where $z_{k,0}$ is the z value at the beginning of the kth interval. The interpolants can be compared in all z locations within the interval.

4. For the first two interpolants constructed by this algorithm, the data points used to build $Interpolant_i$ and $Interpolant_{i+1}$ are saved within a dictionary/mapping structure. The dictionary stores the z value at the beginning of each interval

under the index of the interval. The result of the above comparison is an array of boolean variables (one entry per subinterval) indicating whether Equation (4.49) is true or false. When we move to the next interpolation level, i is increased by 1, and the process repeats.

5. The next step is to search the above boolean array for any intervals for which Equation (4.49) was false. We then subdivide each such interval using an intermediate data point (at the interval center). This gives us a new list of data points upon which to build $Interpolant_{i+1}$ ($i = 2$ at the interpolant level 3). The new list includes all the data points used to build $Interpolant_i$ as well as all the new intermediate data points introduced in this step. We then compare $Interpolant_i$ and $Interpolant_{i+1}$ using Equation (4.49).

6. In the experiments, the coefficient β is set to 0.5 in Equation (4.49). The goal is to let the adaptive interpolation algorithm stop when the interpolation errors are no larger than half of the approximate data error, E_{approx}. Such results will be presented in Section 5.2.3.

7. A suitable X_1 value needs to be defined. If X_1 is power of 2, it is convenient to subdivide in the middle of each interval. In our test data of CT incus and malleus, the total image numbers are 189 and 224 respectively. Therefore, for the CT data, the minimum of the X_1 value is 4, while its maximum is 128. If X_1 is 4, the algorithm has only 3 levels to adapt. $X_1 = 128$ provides three points in the first level for both the incus and the malleus regions of CT images. Therefore, possible X_1 values, 8, 16, 32, 64, or 128, were tested.

8. The above process was iterated until Equation (4.49) is true for all intervals, or until all the contour points are included in a given interval. The latter case represents a failure in that the adaptive interpolation algorithm was not able

to obtain an interpolant for which the estimated interpolation error is less than the E_{approx} values associated with the interval.

9. The algorithm is repeated for each of the N angles, which is illustrated in Figure 4.10. N interpolants are computed. They can be evaluated at a number of locations. Then, the new list of contour points with a structure described in Figure 4.2 is saved as XML files.

As the algorithm goes to the next level, as more points are used for one piece, the error for neighboring intervals needs to be rechecked. Changing the data on one piece/interval could cause small changes on neighboring intervals.

Although two interpolants are both monotonic in an interval, their slopes may be different, and thus their interpolated values may change differently. The interval maximum does not always occur at the end point, so all points in the interval are checked to find the maximum.

4.8.4 Interpolation Analysis Method

One approach to analyzing the interpolation results is using the $1-S$ method, based on a similarity index. This method was used in the error analysis in Sections 4.7.4, 4.7.2 and 4.7.3.

The $1-S$ method is used to compare the interpolation results with the filtering results. Two filtering errors, $E_{filtering1}$ and $E_{filtering2}$, were computed in Section 4.7.4, using the $1-S$ method and $\triangle r$ method respectively. $E_{filtering2}$ was set as E_{approx} which is used as the tolerance for the adaptive interpolation algorithm, as in Equation (4.49). The interpolation error measured by the $1-S$ method should be relatively low; the ratio of the interpolation error to $E_{filtering1}$ should be close to β.

All contour points resulting from the adaptive interpolation algorithm can be saved as ROI images, $ROI_{m,interpolate}$ (at index of m), using the region filling algorithm in

Section 4.9.1. These images are compared with the filtering result, $ROI_{m,filtering}$, produced in Section 4.7.4. The overall interpolation error, $E_{m,interpolate}$, is given by:

$$E_{m,interpolate} = 1 - S_{m,interpolate} = 1 - 2 * \frac{|ROI_{m,filtering} \cap ROI_{m,interpolate}|}{|ROI_{m,filtering}| + |ROI_{m,interpolate}|} \quad (4.51)$$

The error averaged over all z locations is given by:

$$E_{interpolate} = \frac{1}{M} \sum_{m=0}^{M-1} E_{m,interpolate}. \quad (4.52)$$

We examined the interpolating error for the incus and the malleus regions in CT images and they are given in Table 5.13 in Section 5.2.3.

Another method to evaluate the success of the interpolation method is to measure the difference (in percentage) between the last interpolant and the filtered data, $r_{filtered}(z)$. At a location z, this difference $E_{interpolant}(z)$ is given by

$$E_{interpolant}(z) = \frac{|Interpolant_{last}(z) - r_{filtered}(z)|}{r_{filtered}(z)}. \quad (4.53)$$

To measure the averaged error between interpolation results and filtered data, the average of $E_{interpolant}(z)$, noted as $E_{interpolant,avg}$, is given by

$$E_{interpolant,avg} = (\sum_{m=0}^{M-1} E_{interpolant}(z_0 + m\triangle z))/M. \quad (4.54)$$

4.9 3D Reconstruction for CT Scans

As reviewed in Section 3.5, 3D reconstruction is a necessary step to present the interpolation result. The Amira software is applied in the 3D reconstruction step. It loads interpolation result images, does surface generation and displays the 3D model, which can be manipulated in 3D, e.g., dragging, panning, and zooming.

There are several steps in 3D reconstruction:

1. Apply a region filling algorithm and generate solid ROI images using interpolation results.

2. Load images into Amira.

3. Generate a 3D model using the SurfaceGen module.

4. View the 3D model result using the SurfaceView module.

5. Perform volume calculation and comparison of different 3D models.

The final 3D models can be measured by visual comparison and volume calculation as a quantitative analysis method.

4.9.1 Region Filling

The region filling algorithm is a preliminary step in 3D reconstruction. It is to fill inside a closed edge contour with a pixel value, so that the region becomes solid and stands out from the background. On a source image, the edge contour is white, while the rest is black. Region filling starts from one seed inside the ROI defined by the edge contour and grows and fills the ROI with the pixel value of 255, which is white. It is a similar but simplified version of region growing algorithms introduced in Section 4.3.2.

Region filling needs to be applied to the 3D contour points obtained from the adaptive interpolation step, to generate a sequence of black and white images with a solid region on each of them. Then, these images can be loaded into Amira for 3D reconstruction, since Amira only takes an image set as input.

Region filling in most image manipulation programs is an interactive process, which takes mouse clicking events from a user to know where to fill. We need an

automatic method to avoid mouse clicking on multiple images. The hypothesis is that automatically filling in a set of edge contour images can be done. However, in the experiment, there are issues and challenges in this step.

The ROI center was considered as the seed location inside the ROI. However, the ROI center is a valid seed for most cases, but not all. The ROI center can land inside the region, outside the region, on the edge contour, or on an artifact. In the last three conditions, the region filling algorithm would fail on filling the target ROI. For a small percentage of images, it is difficult to correctly find the valid seed inside the ROI and fill automatically.

Region filling involves neighbor checking recursively. It is straightforward in C programming. However, in C# .NET, we found that our recursive code was too slow to be effective. As a result, we replaced our approach that used a seed with a faster and more accurate region filling method from the .NET C# library. In the Drawing2D library, GraphicsPath class can have edge lines defined to form a path. Using the path, a Region instance can be created. Then, a method, FillRegion in the Graphics class can fill the region object with a defined color. This approach automatically fills in a closed edge contour efficiently. For example, when applied to our application, 200 region images were filled and saved in 2 seconds.

4.9.2 Model Generation and Volume Calculation

Model generation is provided by the SurfaceGen module in Amira. This module generates a 3D model using an image set with ROIs. The produced surface is a triangular approximation of the all the interfaces of the image stack. The number of triangles depends on the resolution of the input slices.

There is a Volume Measurement function in Amira, by which the volume of the 3D model can be measured. The units in the volume column can be specified by the

user. For our test cases of inner ear bones, the volume can have mm^3 as the unit.

The module TissueStatistics can compute the volume for multiple target regions. It produces a list of statistical quantities, such as the number of regions or materials and their names, the volumes of each region and the number of triangles used to represent the 3D model.

The measurement of the volume and surface area provides useful quantitative data for analysis of 3D reconstruction. This data can be used to compare the 3D models built using different datasets, e.g. segmented, filtered and interpolated.

Chapter 5

Experimental Results

As outlined in Section 4.1, two types of experimental data sets were provided by the EAR-Lab [17]: six sets of histological section images and a set of CT images. The methods and algorithms were designed with both data sets in mind. This chapter presents the results and discussions when the developed algorithms were applied and tested on the appropriate dataset. Early stage experiments were done on histological sections. Sample results of histological sections after registration, segmentation and contour finding are presented. The second stage of our research focused on the CT image data provided. The results of Gaussian filtering, error analysis, adaptive interpolation and 3D reconstruction are presented.

5.1 Results for Histological Sections

Sample results of image registration on histological sections are given below with a demonstration of transformations. The segmentation algorithms were tested on histological sections, and sample results are shown.

Figure 5.1: Colored regions in two neighboring histological sections. From left to right (a-b): a. Colored image data No. 51; b. Colored image data No. 56 (Temporal Bone Foundation, Boston, USA.)

5.1.1 Results of Registration

Both rigid and non-rigid registration techniques were investigated for use on histological slices. Affine transformations including rotation, translation, scaling and shearing were implemented in C#. A program entitled Transform1 implemented rigid transformations and one entitled Transform2 implemented affine transformations.

Figure 5.1 shows two sample neighboring histological sections in the dataset (Case 8486i, slice No. 51 and No. 56). In these two images the goal is to align the three regions indicated by yellow, blue and green. Screen shots of sample results are shown in Figure 5.2. Figure 5.2.a shows overlaying two images before transformation, with the top one semi-transparent. In Figure 5.2.b, the three regions align after the rigid transformation including rotation and translation. This is done using the Transform1 program. As non-rigid transformations were not investigated further, a sample result from Transform2 is in Appendix C.1).

As there was no gold standard for success using non-rigid registration, we used rigid registration to prepare the histological sections for further processing. Least squares based rigid registration in Amira (as outline in Section 3.1.1) was used for this step. A sample result is shown in Figure 5.3. The results show that images are

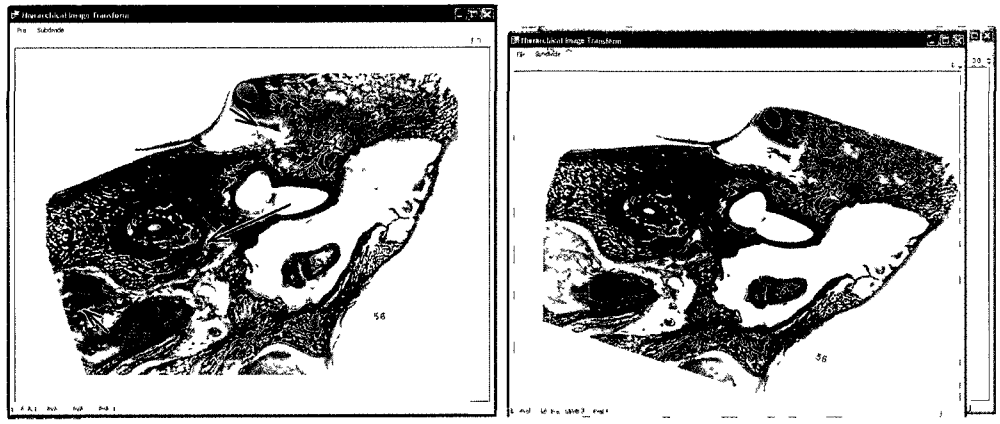

Figure 5.2: Screen shot of before and after transformation of two colored histological sections. From left to right (a-b): a. Before transformation; b. Aligned images after transformation. (Case 8486i, slice No. 51 and No. 56)

Figure 5.3: A sample registration result - (1). From left to right (a-b): a. Before registration; b. After registration. (Case 8486i, slice No. 11)

geometrically transformed so that the image set is aligned.

5.1.2 Results of Segmentation

This section presents preliminary results of algorithms of two-threshold region growing and "traditional" region growing. The two-threshold region growing algorithm provides better global segmentation. This section also shows sample results of the customized local segmentation algorithm, patch growing, as well as Canny edge detection. The overall sample results of our segmentation and contour finding algorithms

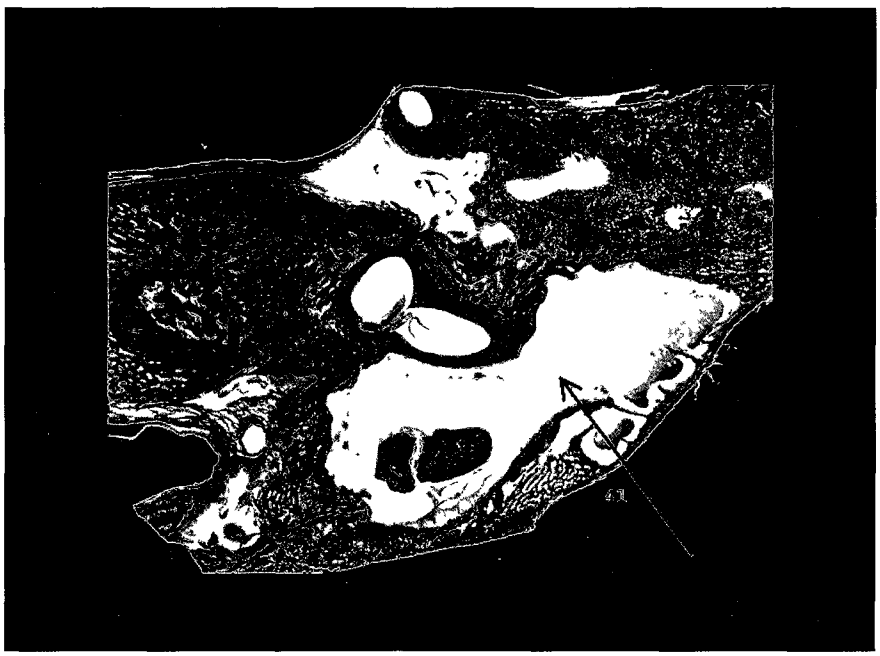

Figure 5.4: Original image for region growing with an arrow pointing to ROI. (Case 8486i, slice No. 41)

are presented.

Results of Two-threshold Region Growing

Preliminary results show that the two-threshold region growing algorithm works better than the "traditional" region growing algorithm in segmenting the test data globally.

Figure 5.4 is the original image, with an arrow pointing to the ROI. This particular image is used to demonstrate challenges in segmentation. This ROI is relatively large for clear viewing. Figure 5.5.a is the gray scale image converted from the original color image, as the source image for the program. A rectangular patch of the ROI completely inside the bright region is selected.

Figure 5.5.b shows the result image of region growing using the patch selected in Figure 5.5.a. The seed is the mean value of the patch selected in Figure 5.5.a. The

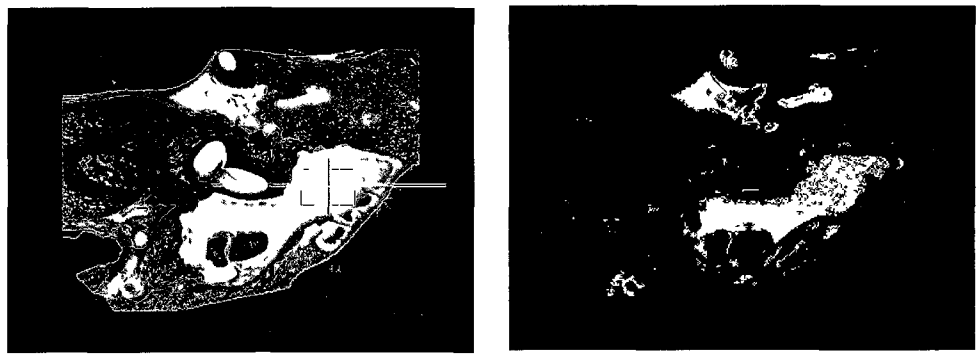

Figure 5.5: Selecting ROI and region growing. From left to right (a-b): a. Selecting ROI in the original image; b. Result image from the region growing method. (Case 8486i, slice No. 41)

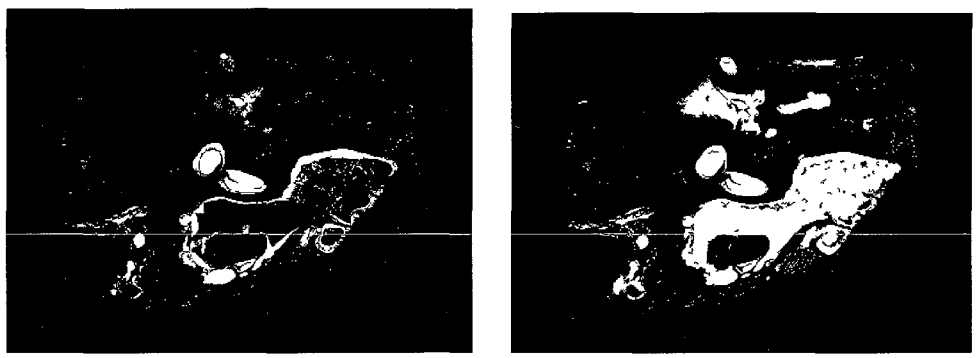

Figure 5.6: Applying two-threshold region growing. From left to right (a-b): a. Thresholding result as the first step; b. Result image after applying the two-threshold region growing method. (Case 8486i, slice No. 41)

problems with this method are (i) grouping stops without including the entire region, and (ii) failing to group all pixels in the selection.

To better deal with these problems. we designed a modified method, two-threshold region growing. The following images demonstrate the results. Figure 5.6.a is the thresholding result using a gray value of 240 (in range of [0, 255]). Figure 5.6.b shows the result image. It shows better segmentation results for the ROI than Figure 5.5.b.

Figure 5.7: Original image for Canny edge detection. (Case 8486i, slice No. 241)

Results of Canny Edge Detection

Figure 5.8 shows a result of applying the Canny edge detection algorithm directly on a sample histological section (as in Figure 5.7). It shows that the Canny edge detection algorithm has good detection, good localization, and low spurious response. However, the result also indicates that applying only Canny edge detection is not satisfactory to our purposes. There are a large number of artifacts in the original images. Therefore, a portion of detected edges show the contour of artifacts, which could be rather misleading in image analysis.

The Canny edge detection method is well suited if being combined with another method, patch growing. Clear edge results are obtained, when the Canny edge detection is applied to the result images from patch growing and median filtering.

Final Results of the Segmentation Framework

The final segmentation algorithm included patch growing, median filtering and Canny edge detection as outlined in Section 4.3. This algorithm was tested on three supplied datasets of histological sections, e.g., Case 8486i, 8486d and 8655d.

Sample results are shown in Figure 5.9, 5.10 and 5.11.

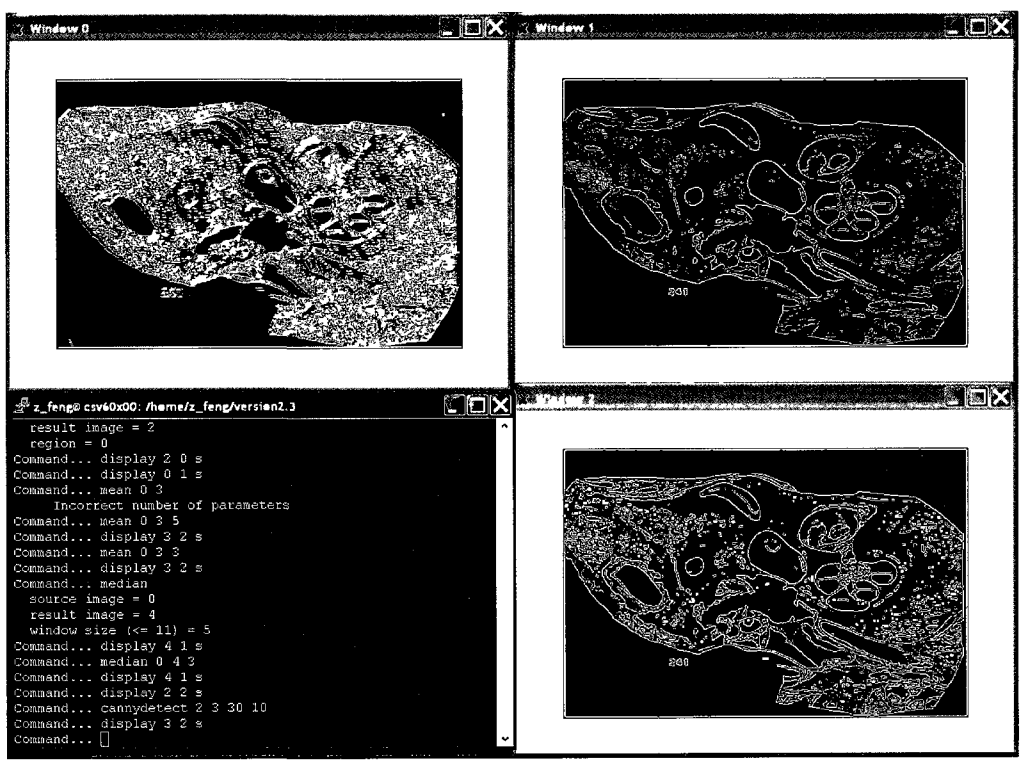

Figure 5.8: Screen shot of the Canny edge detection program. From top left to bottom right (a, b, c, d): a. The phase image; b. The magnitude image; c. The command field; d The result image (Case 84861, slice No. 241)

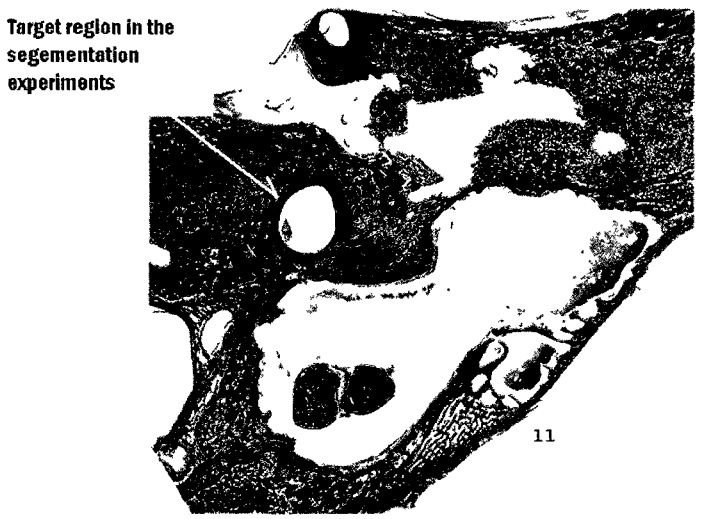

Figure 5.9: Original image showing target region. (Case 8486i, slice No. 11)

The ROI tested is the semicircular canal region which is part of the vestibular organ, shown in Figure 5.9. Test and result images are cropped in Figures 5.10 and 5.11. It is shown that the algorithms of patch growing, median filtering, and Canny edge detection produce clear region and edge image results.

The patch growing algorithm can segment one target region at a time. The region can be extracted from all other regions and noise nearby or with similar intensities. Canny edge detection can find clear edges from the patch growing results.

These segmentation methods are applied automatically on an image set, resulting in satisfactory region and edge images.

Preliminary Results of Contour Finding and Interpolation

Figure 5.12 shows a sample result of the contour finding and preliminary interpolation algorithms for two histological sections, No. 36 and No. 146. In Figure 5.12, starting from the left, the cropped original images show incus as the ROI for segmentation.

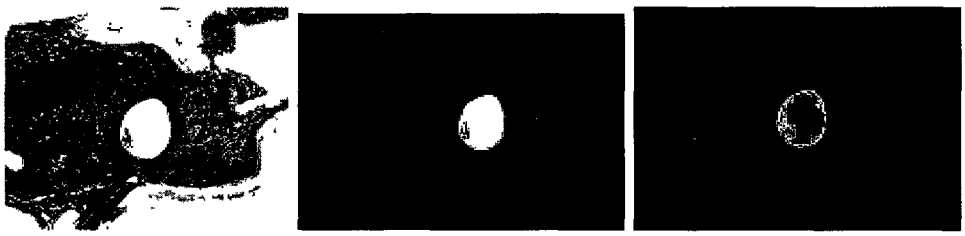

Figure 5.10: Applying patch growing, median filtering and Canny edge detection on a ROI. From left to right (a-c-b): a. Original image; b. Patch growing performed; c. Median filtering and Canny edge detection performed. (Case 8486i, slice No. 11)

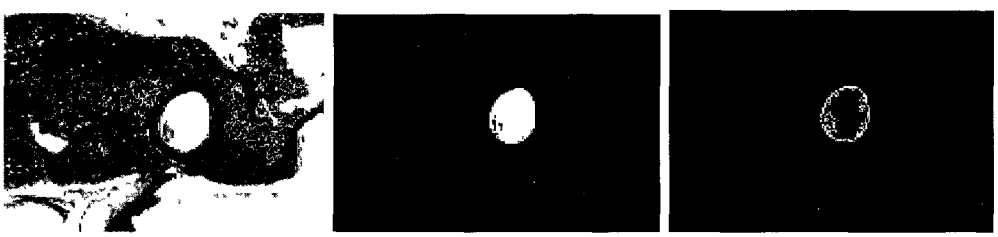

Figure 5.11: Applying above methods in a second image. From left to right (a-c-b): a. Original image: b. Patch growing performed; c. Median filtering and Canny edge detection performed. (Case 8486i, slice No. 15)

After applying the segmentation algorithm, which includes patch growing, median filtering, and Canny edge detection, the region and edge results are obtained. Despite visible noise and artifacts in the original images, the segmented results showed clear and satisfactory region and edge images. Then, the contour finding algorithm further eliminated a small circle artifact in the No. 36 image, to get the result of contour points. The list of contour points were then sent to the preliminary interpolation algorithm. Six generated neighboring images for No. 36 are shown in the upper right corner, and six images for No. 146 are in the bottom right corner.

In summary, the segmentation framework developed for histological sections is highly automatic and efficient. It allows semi-automatic selection of ROIs and applies median filtering, patch growing, and Canny edge detection automatically. Although segmenting histological sections is a challenging task (refer to Section 2.1.3), the results are satisfactory and have good quality. For a dataset with 69 histological

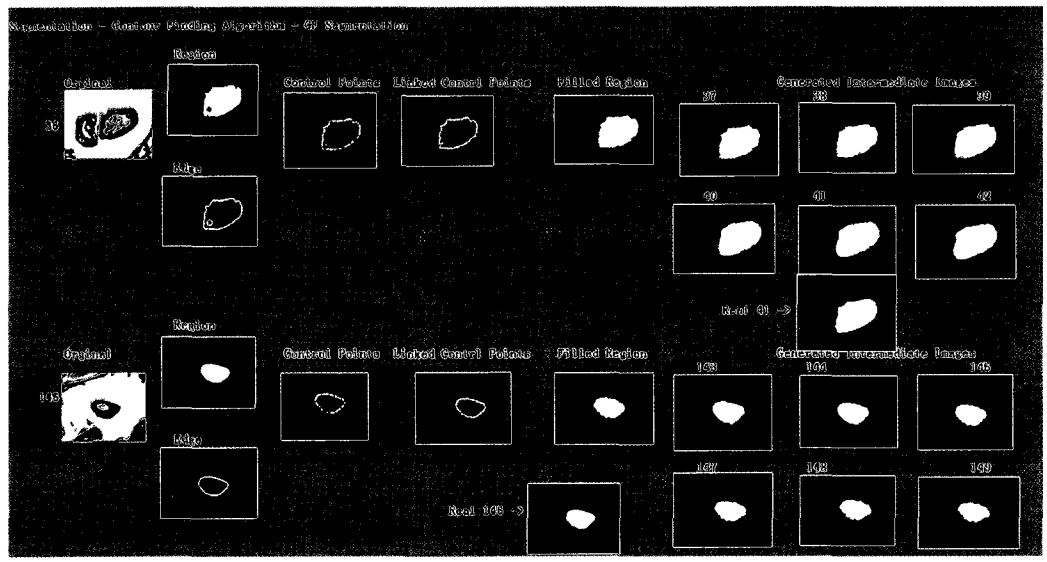

Figure 5.12: Sample results of segmentation, contour finding, and preliminary interpolation algorithms (histological sections, Case 8486i, slice No. 36 and No. 146).

sections, the processing time is less than five minutes on our lab computer (Duo CPU 3.16 GHz, 1.93 GB of RAM), plus one minute of manual interaction. The manual interaction only needs approximately 10 times of mouse clicking to check through 69 histological sections and select all incus or malleus regions. As an exploration into the segmentation techniques useful for histological sections, our segmentation algorithms have shown a large number of histological sections may be segmented rapidly and semi-automatically with a small amount of user interaction.

The contour finding algorithm uses the segmentation results to prepare the histological sections for the interpolation algorithms. Very preliminary work was completed on interpolation algorithms at this point. CT images were received and further investigations continued with the CT dataset (refer to Section 4.1.2).

5.2 Results for CT Images

The CT images were delivered to us in JPG format. Original images as well as images already manually segmented by an expert were received (as introduced in Section 4.1.2). The initial steps in our experiments involved applying the contour finding algorithm, followed by Gaussian filtering.

This section presents the CT test results obtained for Gaussian filtering, error analysis, interpolation and 3D reconstruction. For Gaussian filtering, it shows sample results for the incus in both on-slice and z direction. It also lists quantitative result data for both the incus and the malleus from the filtering performance evaluation and error analysis. This is followed by a section on error analysis results for segmentation, contour finding, and filtering. Interpolation results are presented as the emphasis of this chapter. The 3D reconstruction results are shown last.

5.2.1 Results of Gaussian Filtering

This section presents sample filtering results for the incus. In the Gaussian filtering experiment, 95% and 98% of the power spectrum were used to determine the filter parameters. After contour finding, there are 548 contour points in the on-slice direction at different angles from the ROI center in each image ROI. In total there are 189 images, and therefore 189 contour points in the z direction for each of 548 angles .

Two Gaussian low-pass filters (GLPF) were developed for the incus, one for smoothing contour points on the slice plane, and the other for the z direction. Figures 5.13 and 5.14 show important results before the filters are built. In the on-slice direction, each slice has a power spectrum computed using Equation (4.16). Then, 189 power spectrum functions are averaged using Equation (4.17) to obtain the averaged power spectrum in Figure 5.13. Similarly, 548 z-direction power spectrum functions are averaged to form the power spectrum in Figure 5.14.

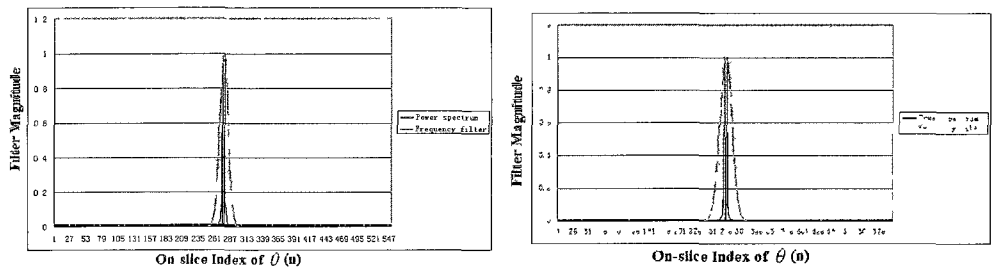

Figure 5.13: Averaged Power Spectrum and GLPFs (on-slice Direction). From left to right (a-b): a. Using 95% of Power Spectrum; b. Using 98% of Power Spectrum

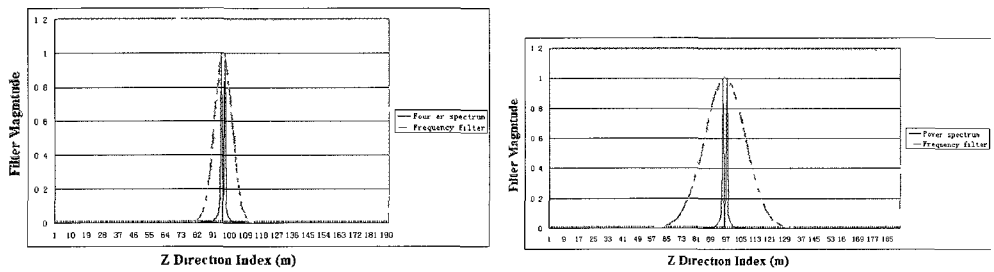

Figure 5.14: Averaged Power Spectrum and GLPFs (z direction). From left to right (a-b): a. Using 95% of Power Spectrum; b. Using 98% of Power Spectrum

Using 95% and 98% of the above power spectrum, Figure 5.15 shows the spatial GLPF developed for the on-slice direction, and Figure 5.16 presents the one in the z direction.

Figure 5.17 is an on-slice sample result from the 95th slice in the dataset. Figure 5.18 is a z-direction sample result from smoothing the list of contour points at 180 degrees. On the left of these two figures, 95% of power spectrum was applied and

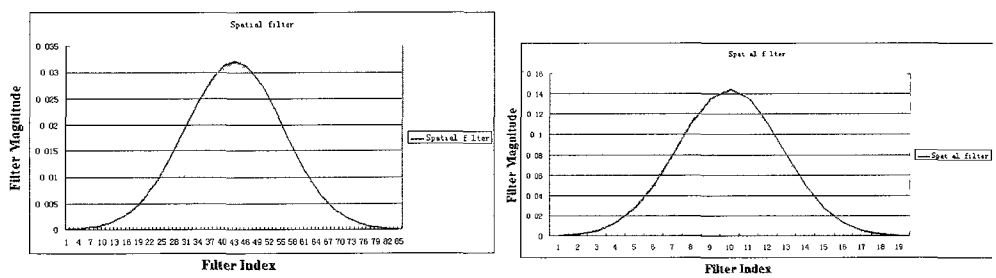

Figure 5.15: Spatial GLPF Developed and Applied (on-slice Direction). From left to right (a-b): a. Using 95% of Power Spectrum; b. Using 98% of Power Spectrum

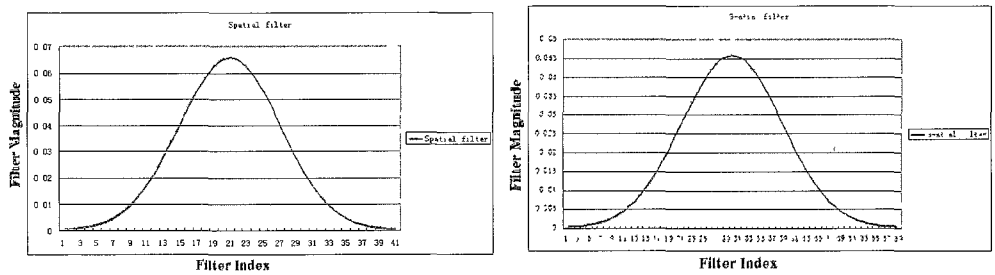

Figure 5.16: Spatial GLPF Developed and Applied (z direction). From left to right (a-b): a. Using 95% of Power Spectrum; b. Using 98% of Power Spectrum

98% was applied on the right for comparison.

Using 95% of power spectrum results in more smoothing than using 98%, but does not preserve the signal outline well. Applying 98% of power spectrum tends to preserve the signal outline better. All of these results show that the automatically developed GLPFs using our algorithms smooths the signal well. Particularly in the z direction, rapid changes near the middle of the curve are removed and the filtered data are very smooth while the overall shape of the curve is preserved. One important fact is that the filtering change is smaller in some places and larger in others.

There are 189 lists of on-slice filtering results in total. Figure 5.17 is one example. Similarly, there are 548 z-direction result lists of which Figure 5.18 is an example. These are 2D filtering results that contribute to smoothing the 3D structure. 3D models built with and without filtering on-slice and in the z direction are shown in Figures 5.39. A visual comparison shows the model on the right appears much smoother than the left model.

We used 98% of the power spectrum as input and applied filtering to the contour points. The error analysis of filtering using GLPFs is given in the following, Section 5.2.2.

Tables 5.1 and 5.2 list important parameters that were computed using 95% and 98% of the power spectrum. They are used to build GLPFs in on-slice and z directions respectively. Our method relates the percentage of power spectrum to the

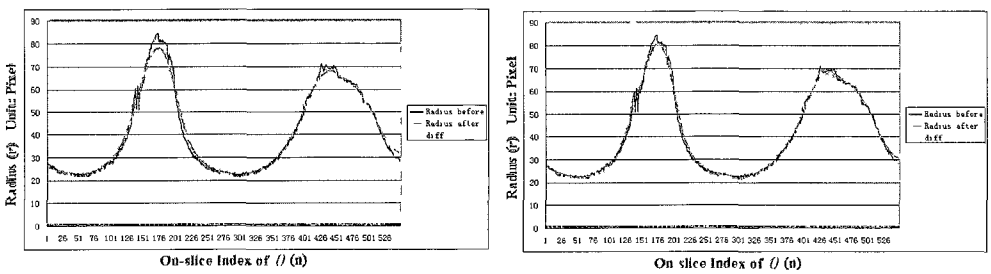

Figure 5.17: Gaussian Filtering Example Results (on-slice Direction - 95th Slice), CT incus data. From left to right (a-b): a. Using 95% of Power Spectrum; b. Using 98% of Power Spectrum

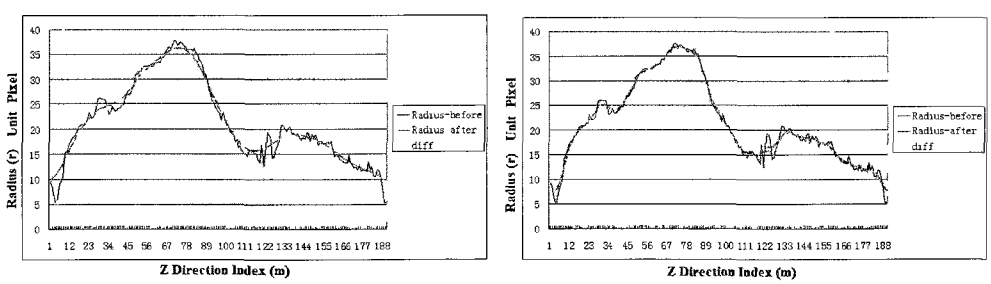

Figure 5.18: Gaussian Filtering Example Results (z direction - 180 degrees), CT incus data. From left to right (a-b): a. Using 95% of Power Spectrum; b. Using 98% of Power Spectrum

Power Spectrum (α)	σ_f	σ_s	W_s
95%	7	0.0128	85
98%	10	0.018	59

Table 5.1: On Slice Filter Parameters.

Power Spectrum (α)	σ_f	σ_s	W_z
95%	5	0.026	41
98%	11	0.058	19

Table 5.2: z direction Filter Parameters.

computation of σ_f and σ_s in a creative design. The parameter, the filter width of the spatial GLPF, W_s, is automatically computed (the method is introduced in Section 4.6.3). The spatial filter is fully defined with values of σ_s and W_s.

To summarize, our Gaussian filtering algorithm is an automatic and effective process. The algorithm achieves the goal of smoothing the contour points that represent a 3D model structure. As a contribution of this thesis, it fully defines the GLPF using a suitable value of α (the percentage of the power spectrum). Several aspects of the test results are presented. Sample results of on slice and z-direction filtering using developed GLPFs show that small fluctuations in the signal are successfully removed, while the outline of the signal is preserved. Quantitative filtering error analysis indicates how the data points are changed. 3D models built from filtering results display a smoother surface.

5.2.2 Results of Error Analysis

The CT dataset was tested, and the errors were computed using the methods in Section 4.7. The error analysis results for CT dataset are listed in the following sections: 1) segmentation error; 2) contour finding error; and 3) filtering error. A summary of error analysis results are presented at the end. It shows how different errors are combined to be an overall error, and how we chose the error to be applied

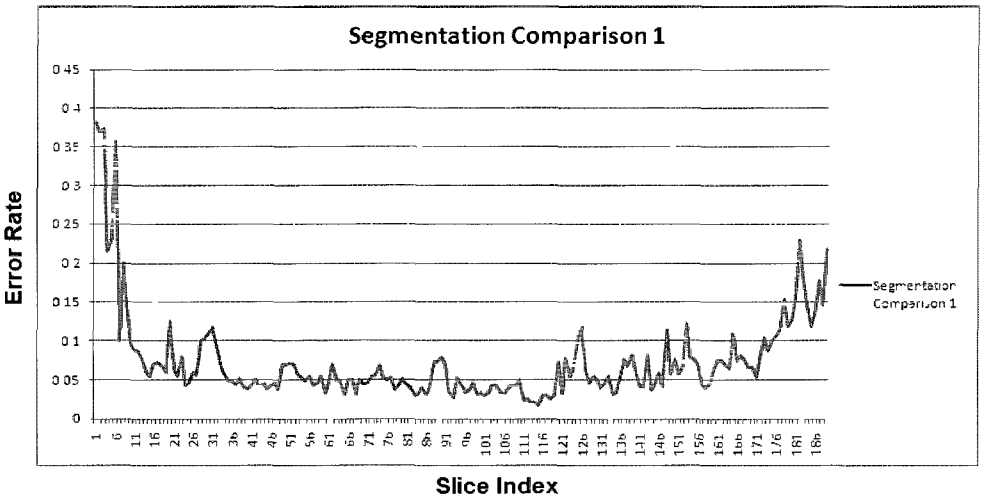

Figure 5.19: Segmentation comparison on sample CT data (1), measuring difference between results of Zhenfeng's first test and Research Associate at the EAR-Lab [17]. (A plot of $E_{segment1}$ array)

in the next step, interpolation.

Results of Segmentation Error

Using the methods in Section 4.7, three arrays of error data, $E_{segment1}$, $E_{segment2}$, and $E_{segment3}$ were calculated and the results are presented in Figures 5.19, 5.20 and 5.21. Their mean and standard deviation values are presented in the Table 5.3. These three lists of results are averaged to be one list, $E_{segment}$, to be applied as the overall segmentation errors.

The segmented images were provided to us by the EAR-Lab [17], and used as reference images. Each of the three new manual segmentations was compared to that of the expert. Variability in the segmentation results was large with the averaged segmentation error ranging from 7.3% to 7.6%. The largest errors occurred in the first and last images in the set and ranged from 15% to 35%. This is because the ROI in these images are very small so that the absolute error is very large relative to the size of the region.

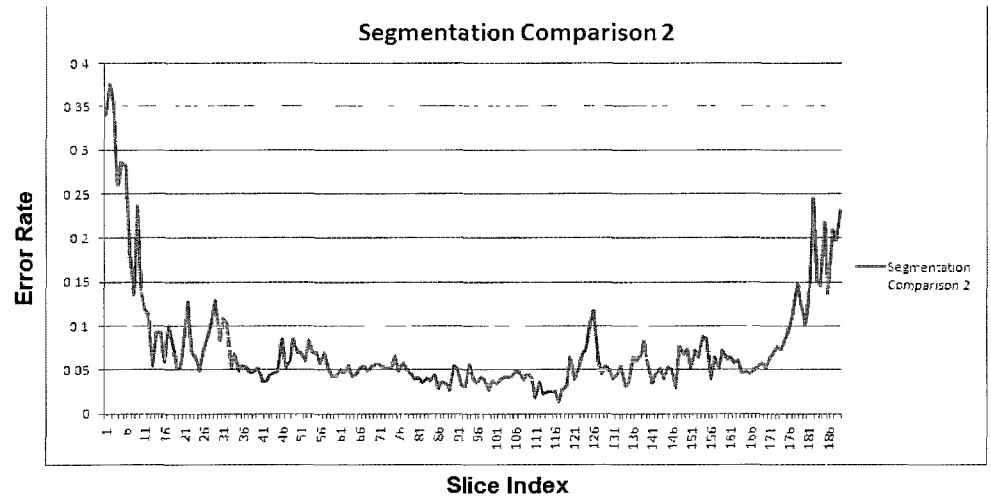

Figure 5.20: Segmentation comparison on sample CT data (2), measuring difference between results of Zhenfeng's second test and Research Associate at the EAR-Lab [17]. (A plot of $E_{segment2}$ array)

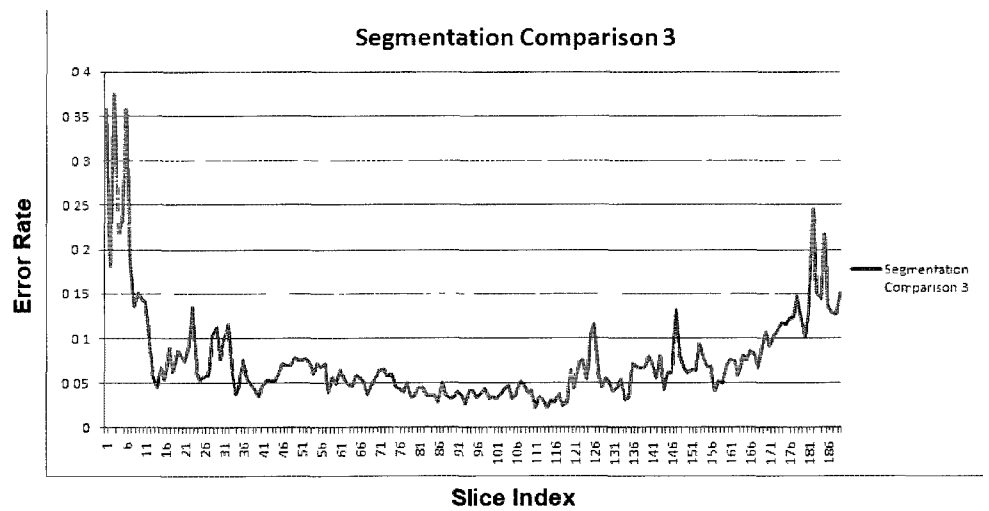

Figure 5.21: Segmentation comparison on sample CT data (3), measuring difference between results of third party tester and Research Associate at the EAR-Lab [17]. (A plot of $E_{segment3}$ array)

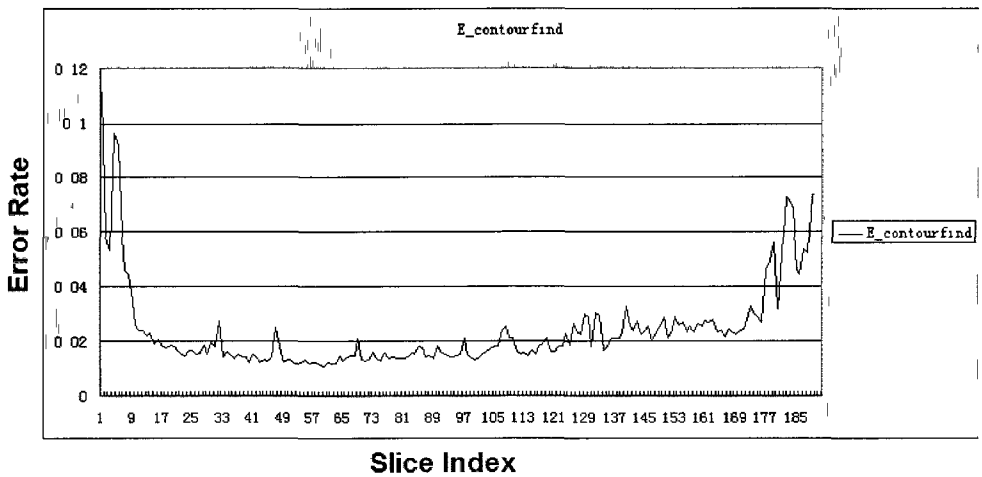

Figure 5.22: Contour finding algorithm errors on sample CT data. x-axis: 189 images; y-axis: error rate, between 0 and 1.

Results of Contour Finding Error

Applying the method in Section 4.7, contour finding errors were calculated for both the incus and the malleus. The errors are summarized in Table 5.3. Figure 5.22 shows a graph of the error for the incus region.

Results of Filtering Error

Figures 5.23, 5.24. and 5.25 are plots of the three types of filtering errors on each slice combined to be one curve for 189 images in the CT incus test, according to the method in Section 4.7.4. The average for each of these error values is summarized in Table 5.3.

Summary of Error Analysis Results

Error analysis results for CT incus and malleus are summarized in Table 5.3. It presents the average and standard deviation of errors in segmentation, contour finding, and filtering methods. It provides information on the magnitudes of various errors. They were calculated as discussed in Section 4.7 and used to obtain E_{approx} which is

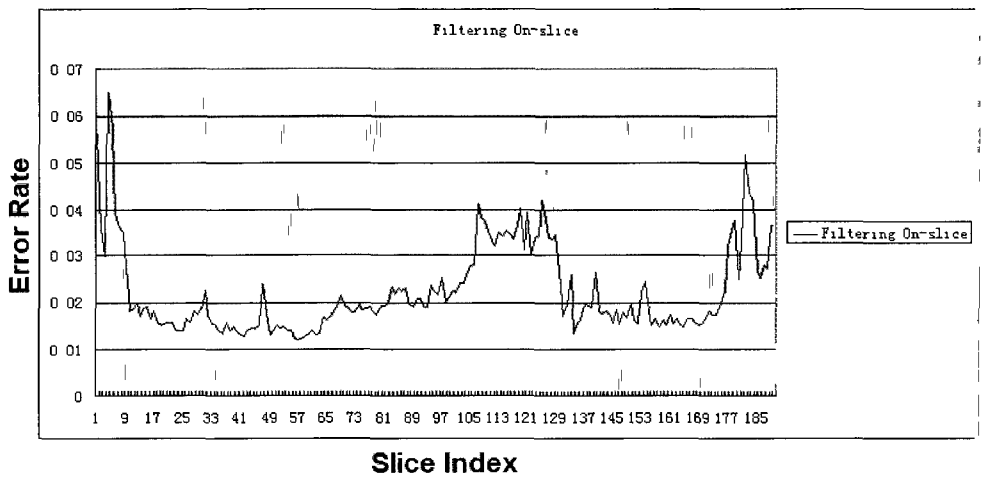

Figure 5.23: Filtering algorithm errors on sample CT data (on slice). x-axis: 189 images; y-axis: averaged error rate on 548 contour points in each image.

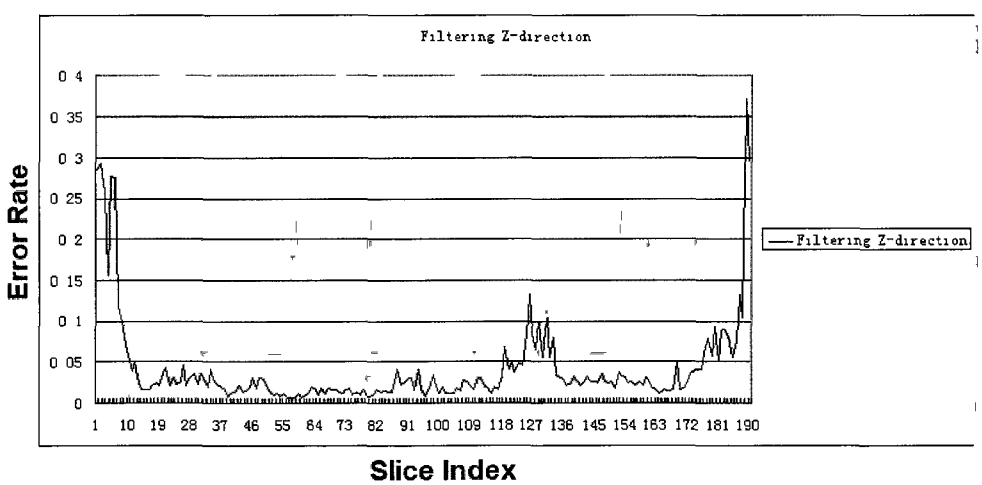

Figure 5.24: Filtering algorithm errors on sample CT data (z direction). x-axis: 189 images; y-axis: averaged error rate on 548 contour points in each image.

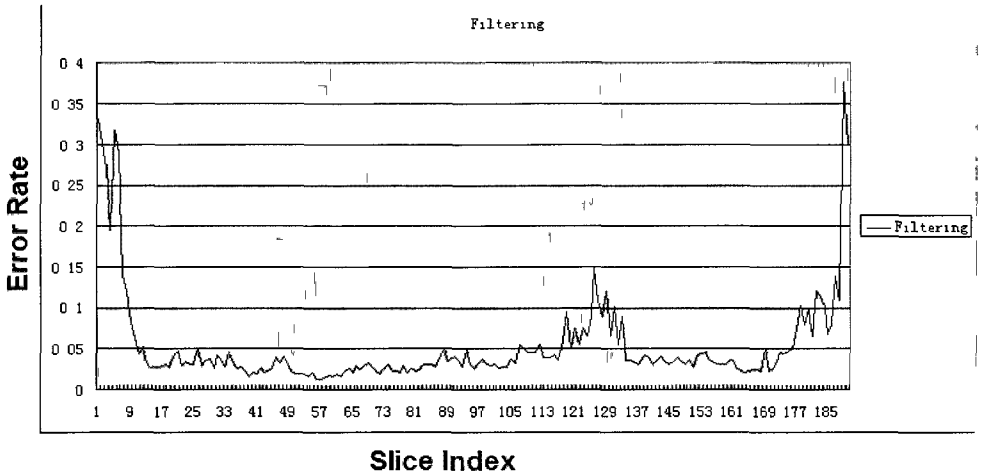

Figure 5.25: Filtering algorithm errors on sample CT data (overall). x-axis: 189 images; y-axis: averaged error rate on 548 contour points in each image.

the input of the adaptive interpolation algorithm.

The contour finding algorithm is a very complex algorithm that obtains all contour points from segmentation results, with a creative design. It provides connectivity between segmentation results and Gaussian filtering and interpolation methods.

Taking the incus as an example, Table 5.3 shows that the largest error is the segmentation error of 7.46% (0.0746). The next is the filtering error of 5.30% (0.0530), and the smallest error is the contour finding error of 2.38% (0.0238). These give a combined error, $E_{overall}$, of 9.23% (0.0923). The errors of the malleus are similar in magnitude to the incus with segmentation, filtering, contour finding and combined errors being 7.69%, 4.28%, 2.32% and 8.8% respectively.

The segmentation results show significant variability between manual segmentations. We concluded that it was extremely important to have a reliable, consistent and accurate automated segmentation algorithm. We developed a segmentation framework for histological sections, but were provided with segmentation results for the CT images. We proceeded with the provided segmentation results and used the filtering error when calculating the interpolation tolerance. As a result, the 2D array,

Error Name	Incus		Malleus	
	Average	Standard Deviation	Average	Standard Deviation
$E_{segment1}$	0.0734	0.0591	0.0752	0.0485
$E_{segment2}$	0.0747	0.0597	0.0764	0.0487
$E_{segment3}$	0.0757	0.0541	0.0769	0.0506
$E_{segment}$	0.0746	0.0576	0.0762	0.0493
$E_{contourfind}$	0.0238	0.0160	0.0232	0.0158
$E_{filteringS}$	0.0224	0.0262	0.0160	0.0171
$E_{filteringZ}$	0.0415	0.0325	0.0362	0.0285
$E_{filtering1}$	0.0489	0.0433	0.0374	0.0381
$E_{filtering2}$	0.0530	0.0471	0.0428	0.0371
$E_{contourfiltering}$	0.0554	0.0453	0.0437	0.0395
$E_{overall}$	0.0923		0.0880	

Table 5.3: Overall error analysis results (in ratio to 1) for CT data, both the incus and the malleus.

$E_{filtering2}$, is set to E_{approx} and passed to the adaptive interpolation algorithm as input. Reducing the segmentation and contour finding errors was left for "future work".

5.2.3 Results of Interpolation

This section presents the results of the adaptive interpolation algorithm in several subsections: 1) initial spline experiments that show that cubic splines do not produce satisfactory results; 2) a sample result for monotonic piecewise cubic interpolation that suggests that it is a suitable technique for our application; 3) results obtained with different X_1 values to determine an appropriate X_1 value for our algorithm; 4) tests to show that the adaptive algorithm is correctly implemented, and has satisfactory performance; 5) sample results for application to the incus; 6) sample results for application to the malleus; 7) since we observed several examples where the results showed larger but localized errors on small intervals in the incus and the malleus sample results, we experienced with different β values to explore the trade-off between treatment of these lager error and data usage; we show that these larger errors can

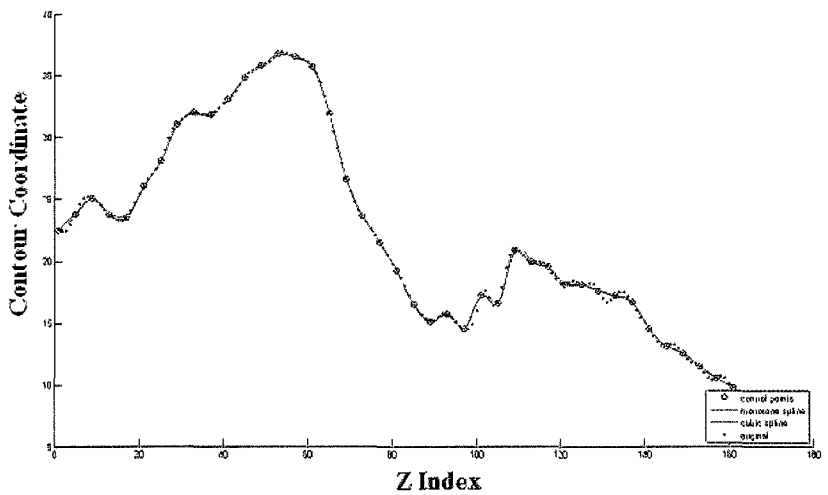

Figure 5.26: Example 1 of preliminary results. Cubic spline and monotonic piecewise cubic interpolation based on every 4th data point.

be completed eliminated using a smaller β value of 20%; and 8) we also provide over results using β equal to 50%.

Note that when applying Equation (4.49) to obtain results in the following subsections, unless it is specified, otherwise the coefficient β is set to 0.5 (50%).

1) Cubic Spline Result and Conclusion

In testing cubic splines on the medical images, the interpolant was found to be outside the reasonable range in some intervals. Examples of such results are shown in Figures 5.26 and 5.27, in which cubic spline interpolation is done on every 4th and every 8th sample point. Our preliminary experiments showed that a cubic spline was difficult to control and that interpolation errors were relatively large in some intervals.

We concluded that the cubic spline was not suitable for our research.

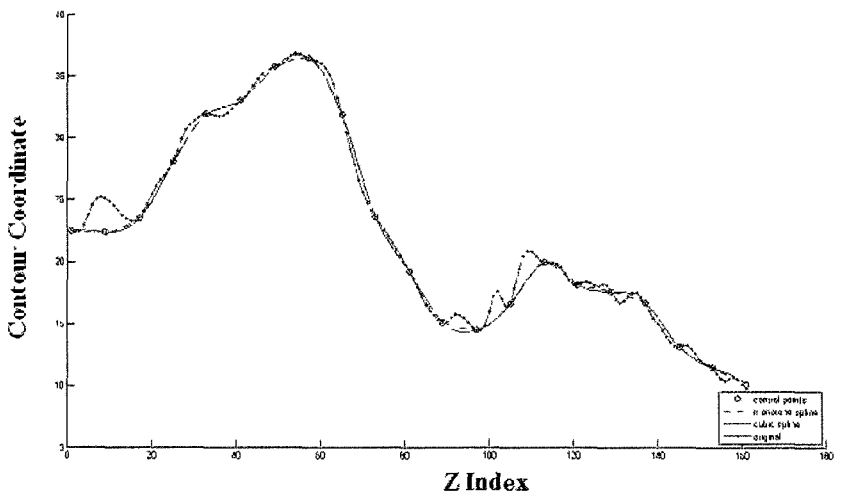

Figure 5.27: Example 2 of preliminary results. Cubic spline and monotonic piecewise cubic interpolation based on every 8th data point.

2) Monotonic Piecewise Cubic Sample Result

Monotonic piecewise cubics were found to be a suitable interpolation technique for our interpolation application. Figure 5.28 is an example. It shows the monotonic piecewise cubic deals better than cubic spline interpolation on intervals in which the data values change direction, using actual CT sample data. Because results of cubic splines are not satisfactory, we decided to proceed with monotonic piecewise cubics as the interpolation technique in our application.

3) Results for different X_1 values

The interval size at the first level of the adaptive interpolation algorithm, X_1, is an important parameter. It is useful to determine a suitable X_1 value for the adaptive interpolation algorithm, as discussed in Section 4.8.3. Table 5.4 shows possible X_1 values corresponding to different interval sizes; these interval sizes are represented by values of $X_1, X_2, X_3, \cdots, X_8$. Possible X_1 values, 8, 16, 32, 64, or 128, were tested. Different X_1 values result in different data usage that is shown in Table 5.5.

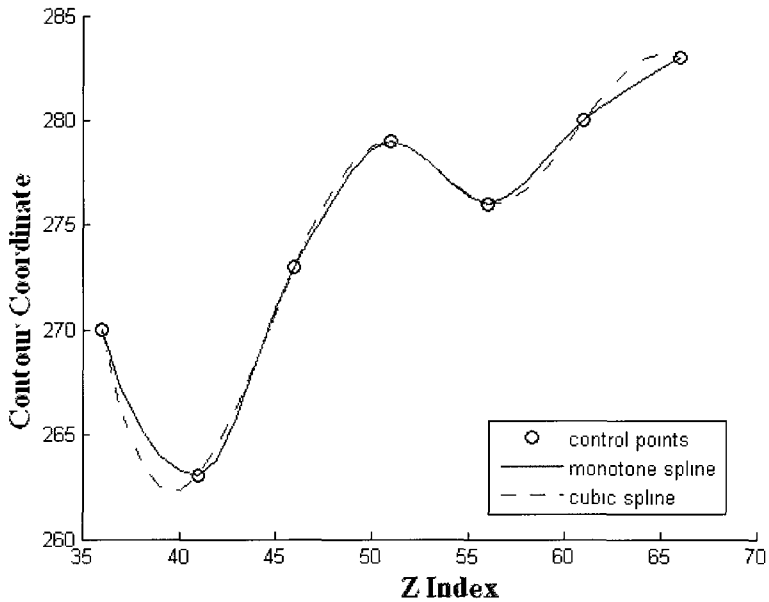

Figure 5.28: Cubic spline and monotonic piecewise cubic interpolation plotting on real sample data (CT images, incus, No. 36 to No. 66)

Also as shown in Table 5.5, as X_1 increases, the average number of data points used in every slice, $N_{used,avg}$, and the ratio between the number of points on average and the total point number available, R_{used}, generally become less, while the same accuracy is obtained. This shows that a greater X_1 value assists in optimizing data usage. As X_1 reaches 64 and 128, the difference in results is small. In theory, the algorithm can start its first level using three points, which means we set X_1 to 128. Hence, $X_1 = 128$ is applied to on-going testing to obtain CT experimental results.

4) Results of algorithm testing

Tests were done to see how close the two interpolants in the last adjacent levels in the adaptive interpolation algorithm were. Figure 5.29 shows the last two interpolants almost perfectly match for both the incus and the malleus at 0 degrees.

Besides the visual comparison of Figure 5.29, Equation (5.1) computes the dif-

	X_1=8	X_1=16	X_1=32	X_1=64	X_1=128
X_1	8	16	32	64	128
X_2	4	8	16	32	64
X_3	2	4	8	16	32
X_4	1	2	4	8	16
X_5	N/A	1	2	4	8
X_6	N/A	N/A	1	2	4
X_7	N/A	N/A	N/A	1	2
X_8	N/A	N/A	N/A	N/A	1

Table 5.4: The X_1 table for adaptive interpolation

X_1 **Value:**	**8**	**16**	**32**	**64**	**128**
N_{level}	4	5	6	7	8
N_{total}	189	189	189	189	189
$N_{used,avg}$	78	67	65	64	64
$N_{used,std}$	6.721	10.368	10.993	11.180	11.569
$N_{used,max}$	97	98	93	92	97
$N_{used,min}$	58	40	38	28	26
R_{used}	41.2%	35.5%	34.4%	33.9%	33.9%

Table 5.5. Tests of adaptive interpolation using different X_1 input, CT incus data. Summarized data are how many levels the algorithm reaches (N_{level}; its value is averaged and rounded to integers), the number of total data points available (N_{total}). For the number of data points used in every slice, it shows the average ($N_{used,avg}$), standard deviation ($N_{used,std}$), maximum ($N_{used,max}$), and minimum ($N_{used,min}$), and ratio (R_{used}) between the number of points used on average and the total point number available.

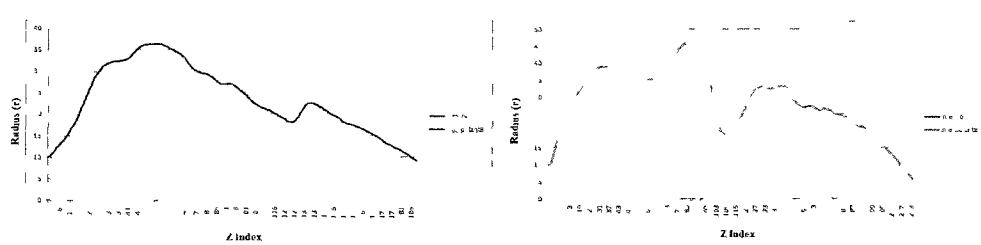

Figure 5.29: Sample test of interpolation results, the last two radius interpolants of CT incus 0 degrees. From left to right (a-b): a. CT incus 0 degrees; b. CT malleus 0 degrees.

$Diff_{interpolant}(z)$	**Degree 0**	**Degree 90**	**Degree 270**
Average (%)	3.66E-05	2.06E-04	4.071E-05
Minimum (%)	0	0	0
Maximum (%)	0.00277	0.00653	0.00156

Table 5.6: Statistics of differences between the last two interpolants for θ = 0, 90, and 270, CT incus data

$Diff_{interpolant}(z)$	**Degree 0**	**Degree 90**	**Degree 270**
Average (%)	0.000251	0.000277	0.000550
Minimum (%)	0	0	0
Maximum (%)	0.00425	0.0119	0.0213

Table 5.7: Statistics of differences between the last two interpolants for θ = 0, 90, and 270, CT malleus

ference between the last two interpolants by evaluating them at z locations. The difference is noted as $Diff_{interpolant}(z)$ in percentage.

$$Diff_{interpolant}(z) = \frac{|Interpolant_{last}(z) - Interpolant_{secondlast}(z)|}{Interpolant_{last}(z)} \tag{5.1}$$

Table 5.6 shows the average, minimum and maximum of the $Diff_{interpolant}(z)$ values for 189 z locations of the original 189 incus source images at degree 0, 90, and 270, as sample results. Table 5.7 presents the same information for the malleus in 224 original z locations. The results show that the interpolation error is much smaller than the overall average approximate data error, $E_{filtering2}$ (refer to Table 5.3), which is calculated using the same $\triangle r$ method (see Section 4.7). This means that the interpolation error is much smaller than the data error.

The tests show that the last two interpolants of the adaptive interpolation algorithm almost perfectly match and the difference between them is small enough, based on both visual and quantitative information. The resulting interpolation error is much smaller than the data error. It indicates that the adaptive interpolation algorithm is correctly implemented, and that it performs as designed.

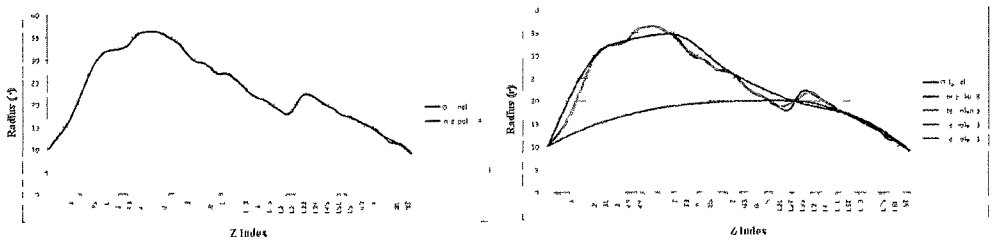

Figure 5.30: Sample interpolation results, radius interpolant for the CT incus, θ: 0 degrees From left to right (a-b)· a. Plots of the last (8th) interpolant and filtered data; b. Plots of the 1st, 3rd, 5th and 8th interpolants and filtered data

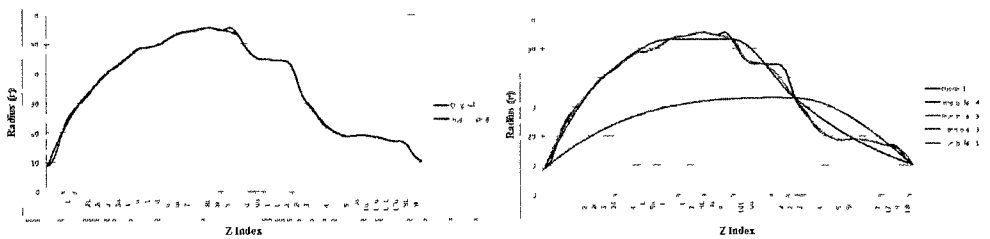

Figure 5.31: Sample interpolation results, radius interpolant for the CT incus, θ: 90 degrees. From left to right (a-b). a Plots of the last (8th) interpolant and filtered data; b. Plots of the 1st, 3rd, 5th and 8th interpolants and filtered data

5) Results for adaptive interpolation applied to the Incus

Adaptive interpolation results in z direction at three different angles, θ equal to 0, 90, 270 degrees, are shown below.

Figures 5.30, 5.31, and 5.32 are three sample interpolation results for the CT incus tests. Three plots on the left show how well the last interpolant matches the original data, where the original data are the filtered results. Three plots on the right show how the interpolants gradually adapt to approximate the data closer from the 1st, 3rd, 5th to 8th level of the adaptive algorithm.

Table 5.8 shows how many knots are used in each level of the adaptive interpolation and each interpolant. The algorithm is able to find knots needed increasingly one level after another. During this process, the interpolants fit closer and closer to the filtered data.

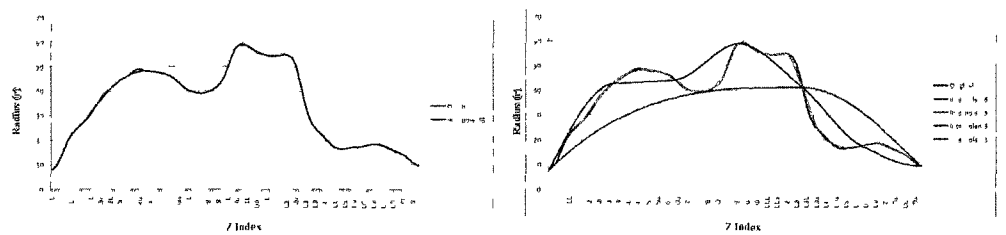

Figure 5.32: Sample interpolation results, radius interpolant for the CT incus, θ: 270 degrees. From left to right (a-b): a. Plots of the last (8th) interpolant and filtered data; b. Plots of the 1st, 3rd, 5th and 8th interpolants and filtered data

Interpolant #	**Degree 0**	**Degree 90**	**Degree 270**
Interpolant1	Count = 3	Count = 3	Count = 3
Interpolant2	Count = 4	Count = 4	Count = 4
Interpolant3	Count = 7	Count = 7	Count = 7
Interpolant4	Count = 13	Count = 13	Count = 13
Interpolant5	Count = 25	Count = 24	Count = 25
Interpolant6	Count = 45	Count = 33	Count = 46
Interpolant7	Count = 56	Count = 41	Count = 68
Interpolant8	Count = 60	Count = 46	Count = 73

Table 5.8: Knot number counts for adaptive interpolants for $\theta = 0$, 90, and 270, CT incus data

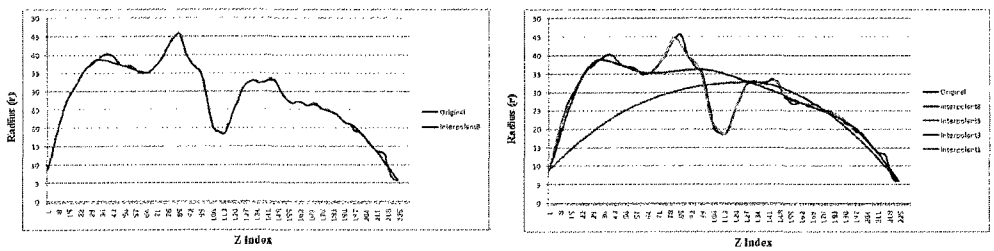

Figure 5.33: Sample interpolation results, radius interpolant for the CT malleus, θ: 0 degrees. From left to right (a-b): a. Plots of the last (8th) interpolant and filtered data; b. Plots of the 1st, 3rd, 5th and 8th interpolants and filtered data

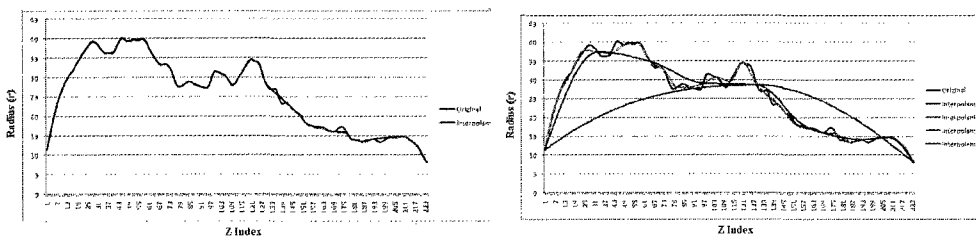

Figure 5.34: Sample interpolation results, radius interpolant for the CT malleus, θ: 90 degrees. From left to right (a-b): a. Plots of the last (8th) interpolant and filtered data; b. Plots of the 1st, 3rd, 5th and 8th interpolants and filtered data

6) Results for the adaptive interpolation algorithm applied to the malleus

The results for the malleus are presented in the same format as for the incus. Figures 5.33, 5.34, and 5.35 are three sample interpolation results of CT malleus tests. Similar to CT incus tests, these results show the interpolants adapt to the data points, as more data points are used. Table 5.9 shows how many knots are used for each interpolant.

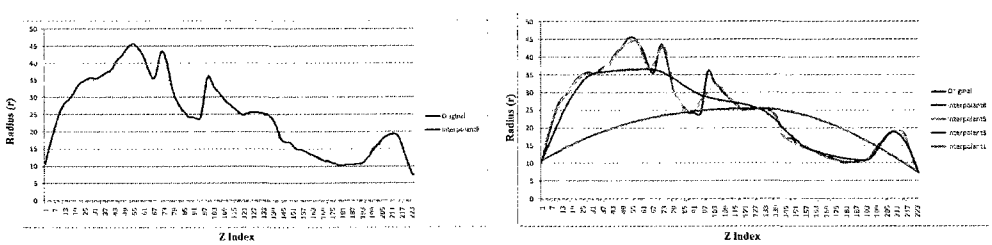

Figure 5.35: Sample interpolation results, radius interpolant for the CT malleus, θ: 270 degrees. From left to right (a-b): a. Plots of the last (8th) interpolant and filtered data; b. Plots of the 1st, 3rd, 5th and 8th interpolants and filtered data

Interpolant #	Degree 0	Degree 90	Degree 270
Interpolant1	Count = 3	Count = 3	Count = 3
Interpolant2	Count = 5	Count = 5	Count = 5
Interpolant3	Count = 9	Count = 9	Count = 9
Interpolant4	Count = 15	Count = 17	Count = 17
Interpolant5	Count = 25	Count = 31	Count = 32
Interpolant6	Count = 42	Count = 55	Count = 58
Interpolant7	Count = 69	Count = 88	Count = 93
Interpolant8	Count = 90	Count = 114	Count = 125

Table 5.9: Knot number counts for adaptive interpolants for $\theta = 0$, 90, and 270, CT malleus

7) Summary of the incus and the malleus results

In the above two subsections, sample results of the adaptive interpolation algorithm are presented for two regions, the incus and the malleus, separately. These results show the break-down steps of the algorithm and how this automatic process works in detail.

Results of both the incus and the malleus at degree 0, 90, and 270 show that the accuracy of interpolants increases while the level number becomes higher, from 1 to 8. It shows the change from using very coarse data (3 knots) to less coarse data (more and more knots).

There are a few places where the curve of $Interpolant_8$ does not match with the curve of the filtered data. These places can be called "error bumps"; the error bumps occur only in a small number of intervals. This is because not all of the filtered data are used in the comparison in Equation (4.49). The difference between two neighbor interpolants are compared instead. After the two neighbor interpolants match closely, and the difference between them gets small enough, or the algorithm has no more levels to adapt, the process would stop. In other words, the algorithm stops because the last two interpolants are nearly identical, although the last interpolant still varies slightly from the original/filtered data in small number of places.

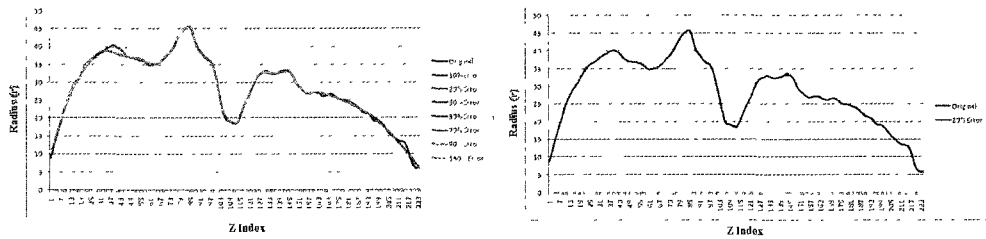

Figure 5.36: Comparison between original data points and final interpolants using different β values, CT malleus, $\theta = 0$ degrees. From left to right (a-b): a. Plots of filtered data and the last (8th) interpolants with the β values of 10%, 20%, 25%, and 30% of errors; b. final interpolant has a nearly perfect match to the filtered data at $\beta = 20\%$.

The following subsection show results with "error bumps" being reduced; this requires the use of more data points.

8) Results for different β values

Results in above sections applied the β value of 50%. This section investigates different β values. There are a few error bumps that occur in some figures, e.g., Figure 5.33. These error bumps happen in places where E_{approx} is large, which is reasonable. The value of E_{approx} fluctuates for each angle with some of the values being large and others small.

As Equation (3.34) indicated, there is no need to refine the approximation of interpolation to be more accurate than the original/filtered data. The data samples being interpolated have larger errors at places where the error bumps occur.

The β values of 150%, 90%, 70%, 50%, 30%, 20%, and 10% were used in the experiment, to see whether the error bumps associated with $Interpolant_8$, can be reduced.

Figure 5.36 presents plots of original data points (after filtering) and final interpolants ($Interpolant_8$) with different β values. Figure 5.33 shows that error bumps are substantially reduced when β is equal to 20%.

β	**150%**	**90%**	**70%**	**50%**	**30%**	**20%**	**10%**
Used points	37	56	66	89	116	169	195
Total points	224	224	224	224	224	224	224
Ratio	16.5%	25.0%	29.5%	39.7%	51.8%	75.4%	87.1%
$E_{interpolant,avg}$	1.523%	1.381%	1.355%	1.189%	0.399%	0.084%	0.020%

Table 5.10: Number of data points used for different β values, CT malleus, $\theta = 0$ degrees

Table 5.10 shows the number of data points used for the final interpolant, the total points available, and the ratio between used and total points, as β varies from 10% to 150%. The angle is $\theta = 0$ degrees. Values of $E_{interpolant,avg}$ are computed using Equations (4.53) and (4.54). This table shows that the error $E_{interpolant,avg}$ becomes smaller if the β value decreases, and that the number of used data points increases. Therefore, the selection of the β value is key to the success of the interpolation.

Table 5.11 shows the overall final results of interpolation for CT malleus. The tested β values are 150%, 90%, 70%, 50%, 30%, 20%, and 10%. It shows how the data usage is changed with different β values. For example, using $\beta = 20\%$, it shows that 74.2% of the data points on average are used in the algorithm. Compared to using 49.8% of the data points, which has error bumps, approximately 25% more data points on average are used. Therefore, as β decreases, the accuracy of interpolation becomes higher, but more data would be used. There is a trade-off between the data usage and the interpolation accuracy, where the value of β is the key element. This result is useful and relevant to indicate how many histological sections are needed for interpolating the incus and malleus data.

The results using different β values show how the interpolation accuracy increases with β values being decreased. Error bumps can be fully reduced using the β value of 20%. When the β value is 50%, the algorithm achieves a satisfactory approximation and uses a relatively small number of data points, with reasonable error bumps visible.

β	**150%**	**90%**	**70%**	**50%**	**30%**	**20%**	**10%**
X_1	128	128	128	128	128	128	128
N_{level}	8	8	8	8	8	8	8
N_{total}	224	224	224	224	224	224	224
$N_{used,avg}$	52.2	77.5	91.5	111.5	143.5	166.3	195.7
$N_{used,std}$	10.9	12.9	14.0	13.8	11.7	9.3	6.8
$N_{used,max}$	79	110	127	142	172	190	213
$N_{used,min}$	14	29	37	42	93	128	159
R_{used}	23.3%	34.6%	40.8%	49.8%	64.1%	74.2%	87.4%

Table 5.11: Overall results of adaptive interpolation, variable β, CT malleus. Symbols are the same as the caption of Table 5.5.

	Incus	**Malleus**
X_1	128	128
N_{level}	8	8
N_{total}	189	224
$N_{used,avg}$	64.1	111.5
R_{used}	33.9%	49.8%
$N_{used,std}$	11.6	13.8
$N_{used,max}$	97	142
$N_{used,min}$	26	42

Table 5.12: Overall results of adaptive interpolation, CT data, $\beta = 50\%$. Symbols are the same as the caption of Table 5.5.

9) Overall Results

Table 5.12 shows how many data points on average are enough for building the final interpolants for all different angles. Since the algorithm starts with the first level interval size, $X_1 = 128$, it has 8 levels in total. The interval size from level 1 to level 8 is 128, 64, 32, 16, 8, 4, 2, and 1. The table shows statistics and percentage of the averaged data points used.

Table 5.13 shows the interpolation error using the $1 - S$ method as in Equations (4.8.4) and (4.52). Compared to $E_{filtering1}$ which is presented in Table 5.3, it shows that the interpolation error, $E_{interpolate}$, is reasonable and small enough. Therefore, the goal to have the interpolation error not larger than 50% (the β value) of E_{approx}

	Incus		Malleus	
	Average	**Standard Deviation**	**Average**	**Standard Deviation**
$E_{interpolate}$	0.0227	0.0255	0.0180	0.0145
$E_{filtering1}$	0.0530	0.0472	0.0374	0.0383
Ratio	0.428 : 1	0.541 : 1	0.481 : 1	0.379 : 1

Table 5.13: Error of adaptive interpolation calculated using 1-S (similarity index), CT data (Unit of error: ratio from 1)

is achieved.

Combining Tables 5.12 and 5.13 shows an important conclusion. That is, the adaptive interpolation algorithm can use 33.9% of the data points for the incus to produce interpolated data points with the error rate of only 2.27% ($E_{interpolate}$ = 0.0227). It uses 49.8% for the malleus and has the error rate of 1.80% (0.0180).

To summarize, experimental results of the adaptive interpolation algorithm are presented in this section. Through spline experiments, we concluded that monotonic piecewise cubics are the most suitable for our application. Both visual comparison and quantitative results show that the algorithm has satisfactory performance as designed. In the experiments, the last two interpolants of the adaptive interpolation algorithm almost perfectly match and the difference between them is small enough. The resulting interpolation error is much smaller than the data error. Sample results of the incus and the malleus show the break-down steps of the algorithm, where the interpolants gradually adapt to the data points and their accuracy increases while at a higher level. Different values of two input parameters, X_1 and β, were tested, and the sample results are shown. The values of $X_1 = 128$ and $\beta = 50\%$ are suitable for our application, because the algorithm uses a relatively small number of data points to achieve a satisfactory approximation. Moreover, the interpolation accuracy can be increased further with smaller β values. Error bumps that occur at $\beta = 50\%$ can be fully eliminated using the β value of 20%. However, that would use more data points.

These results show that the goal of the adaptive interpolation algorithm is achieved.

The algorithm successfully allocates a small number of data points and produces highly accurate results in a fully automated process. For the incus, the adaptive interpolation algorithm uses 33.9% of the data points to obtain interpolation results with the relative error rate of only 2.27%. For the malleus, it uses 49.8% and has the error rate of 1.80%.

This is very good result, because the algorithm has intelligence to find and use approximately 1/2 to 1/3 of the data points to obtain satisfactory approximation. The error is in a reasonable range.

5.2.4 Results of 3D Reconstruction

The interpolation results were used to generate 3D virtual models using the Amira software. The 3D model can be compared visually with the bone photos, as in Figures 5.37 and 5.38. The bone photos on the left are provided by the EAR-Lab [17]. The screenshots on the right are our generated 3D model. The bone photos and the 3D models are not from the same piece of bone. It shows they appear alike, and that 3D reconstruction has done a good job. The difference is that the 3D model in Amira can be viewed from any angle in a 3D space, which is far more useful than a static bone photo.

Figures 5.39.a and 5.39.b show 3D models in Amira built from before filtering and filtered data points of the malleus. These models are tilted to such a viewing angle where the model surface on the right appears much smoother than the left model. It shows our filtering algorithm smooths the model effectively well.

Figure 5.40.a shows the 3D model in Amira built using filtered data points of the malleus. Figure 5.40.b overlaps 3D models using both filtered and interpolated data points. It shows the two models largely overlap, the overall shape is the same, and that the difference between them is very small.

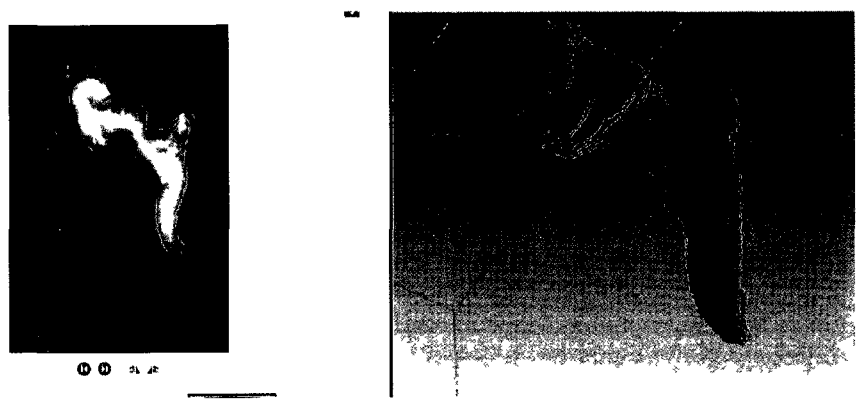

Figure 5.37: Visual comparison of 3D model and the bone photo of malleus (1). From left to right (a-b): a. Photo of the actual bone; b. 3D virtual model built using our method.

Figure 5.38: Visual comparison of 3D model and the bone photo of malleus (2). From left to right (a-b): a. Photo of the actual bone; b. 3D virtual model built using our method.

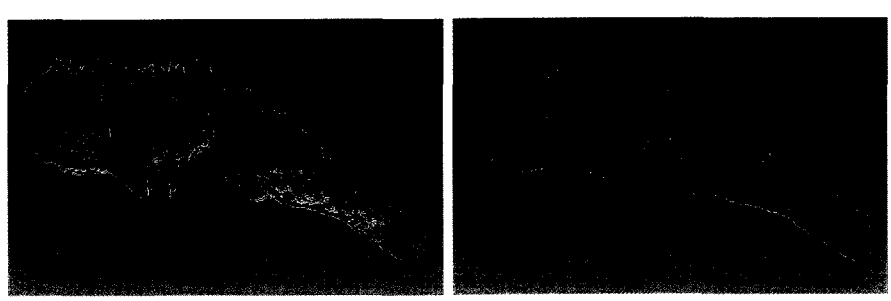

Figure 5.39: Visual comparison of 3D model of the malleus before and after filtering. From left to right (a-b): a. 3D model built using contour points before filtering; b. 3D model built using contour points after filtering.

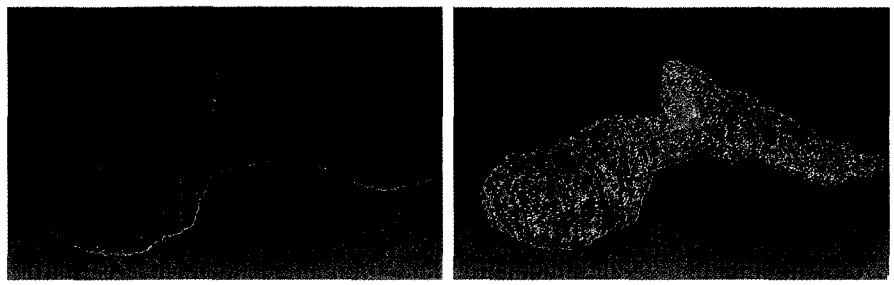

Figure 5.40: Visual comparison of 3D model built from filtered and interpolated data points of the malleus. From left to right (a-b): a. 3D model built using filtered contour points; b. 3D models built using both filtered and interpolated contour points.

Result Data	**Triangles**	**Surface Area** (mm^2)	**Volume** (mm^3)
1. Segmentation	173934	47.072	13.525
2. Filtering	163020	43.666	13.321
3. Interpolation	156584	43.095	13.669
Difference between 2 and 3		1.31%	2.61%

Table 5.14: Quantitative analysis of 3D reconstruction, CT incus; results of the number of triangles (in the polygon mesh), surface area and volume of the 3D model.

Besides visual comparison, volume calculation is a quantitative analysis method to find out the difference between two models Using the CT dataset for example, a complete dataset obtained by manual segmentation and the augmented dataset resulting from interpolation can be compared using 3D reconstruction.

Tables 5.14 and 5.15 shows quantitative analysis of 3D reconstruction, with three quantities measured for the 3D model: the number of triangles (in the polygon mesh), surface area, and volume. Table 5.14 are the results for the incus, and Table 5.15 shows similar results for the malleus. The dimensions and units were introduced in Section 4.1.2. Three types of datasets used for measuring the same model are segmentation results (without any processing work), filtering results (after contour finding and filtering), and interpolation results (after interpolation). The 3D model of filtering results uses the complete filtered dataset, which is considered the upper limit of the accuracy that the adaptive interpolation algorithm could achieve.

dataset	Triangles	Surface Area (mm^2)	Volume (mm^3)
1. Segmentation Results	168866	45.120	11.336
2. Filtering Results	160746	42.485	11.293
3. Interpolation Results	156826	41.707	11.266
Difference between 2 and 3		1.83%	0.24%

Table 5.15: Quantitative analysis of 3D reconstruction, CT malleus; results of the number of triangles (in the polygon mesh), surface area and volume of the 3D model.

ROI	Data Usage	Error of Interpolation	Difference of 3D Model Volume
Incus	33.9%	2.27%	2.61%
Malleus	49.8%	1.80%	0.24%

Table 5.16: Overall comparison between interpolation results and complete filtered data points for the incus and the malleus.

The 3D reconstruction results supports the fact that the interpolation results are satisfactory with high accuracy through visual and quantitative comparisons. The 3D model generated after our adaptive interpolation algorithm appear highly similar with the 3D model generated using filtered data points and the actual bone photos. Quantitatively, the difference between 3D models from filtered and interpolated data points is very small. The error of the area and volume of the incus model is 1.31% and 2.61%, and that of the malleus model is 1.83% and 0.24% respectively.

To conclude, the adaptive interpolation algorithm produces accurate results. Table 5.16 summarizes error measurements. Compared with using complete filtered data points (the upper limit), interpolation results using approximately 1/2 and 1/3 of filtered data points have very small error rates. The algorithm achieves acceptable accuracy in generating missing structure of the human temporal bone automatically.

Chapter 6

Conclusions and Future Work

6.1 Summary

The thesis described the methods applied to the CT data that were developed, with the goal of extending them to histological sections. It explored registration and segmentation techniques for histological sections and focused on contour finding, filtering and interpolation methods for the CT scans. Chapter 1 described the research problem and settings, and introduced the objective, which is to improve existing 3D model generation methods for medical researchers. Chapter 2 reviews the literature and relevant techniques in medical processing, including the topics of medical background, image registration, geometrical transformations, image segmentation, interpolation, and 3D reconstruction.

Chapter 3 described the theory behind the methods and algorithms in this thesis. They are: 1) image registration, 2) image segmentation, 3) noise and filtering, 4) interpolation, and 5) 3D reconstruction.

Chapter 4 presented algorithms and methods developed based on the techniques of Chapter 3. The methods applied are grouped into four categories. The core methods are the Gaussian filtering and adaptive interpolation algorithms (developed

in C#). They are fully automatic, robust and efficient. The secondary methods are image segmentation (using C and CVlab) and contour finding algorithms (using C#). Other software that was used includes stand-alone programs (developed using C# to allow interaction, connectivity and automation), and Amira (for image registration and 3D reconstruction). The stand-alone programs include region selection, image renaming and conversion tools.

Chapter 5 presented results for both histological sections and CT scans. It showed sample results of histological sections after registration, segmentation, and contour finding. It described CT results of Gaussian filtering, error analysis, adaptive interpolation, and 3D reconstruction.

6.2 Conclusions for the Experimental Results

Experimental results show that designed algorithms in this thesis improved existing 3D model generation methods. The proposed methods include useful contributions in segmentation, filtering and interpolation techniques.

Good quality results were obtained for segmenting of histological sections. The designed segmentation framework is highly automatic and efficient. It is a semi-automatic process to select ROIs and fully automatic in median filtering, patch growing, and Canny edge detection for an image set. Our segmentation algorithms have segmented a large number of histological sections rapidly and automatically with a small amount of user interaction.

The contour finding algorithm was designed for histological sections, and then applied to the CT dataset. It is a very complex algorithm that obtains all contour points from segmentation results. It creates a mapping of corresponding contour points on different images using angle attributes. It provides connectivity between segmentation results and Gaussian filtering and interpolation methods. This is a

necessary step, but introduces noise/error. The averaged error rate of this algorithm for the incus is 2.38% and malleus is 2.32%.

The Gaussian filtering algorithm is an automatic and effective process. It fully defines the GLPF using a suitable amount ($\alpha = 98\%$) of the signal power. Two GLPFs were developed and applied by traversing all on-slice and then z-direction lists of contour points, respectively. Sample results show that the algorithm successfully smoothed the signal. 3D models built from filtering results display a smoother surface. Therefore, the algorithm achieves the goal of smoothing the 3D model structure represented by contour points. The contour finding and Gaussian filtering algorithms output valid contour points for interpolation.

The error analysis results of segmentation, contour finding and filtering are computed in detail. This points out the contributions to the error and their magnitudes. The goal is to determine the approximate data error as input to the adaptive interpolation algorithm. Taking the incus for example, the filtering error is 5.30%. Its segmentation and contour finding errors are 7.46% and 2.38% respectively. The combined error is 9.23% (refer to Table 5.3). The malleus has errors similar in magnitude to the incus. We concluded that the filtering error was considered to be the dominant error, and it was used as input for the adaptive interpolation algorithm. We acknowledge that segmentation and contour finding errors are relatively high and the corresponding algorithms need improvements.

The adaptive interpolation algorithm successfully allocates a small number of data points and produces highly accurate interpolation results in a fully automated process. We decided that monotonic piecewise cubics were the most suitable to be applied. Both visual comparison and quantitative results show that the algorithm has satisfactory performance. With two parameters of the program, X_1 and β, being set, final interpolation results were obtained. The adaptive interpolation algorithm uses 33.9% of the incus data points to obtain results with the relative error rate of only 2.27%.

It uses 49.8% for the malleus and has the error rate of 1.80%. The algorithm uses approximately 1/2 to 1/3 of the data points to obtain satisfactory approximations. These results show that the goal of the adaptive interpolation algorithm is achieved.

Visual and quantitative comparisons of 3D reconstruction results indicate that 3D models built using interpolation results are satisfactory and accurate. The errors of the area and volume of 3D models are acceptable. Such errors are 1.31% and 2.61% for the incus, and 1.83% and 0.24% for the malleus respectively.

All these results show that the segmentation framework for histological sections has a good design; the contour finding and Gaussian filtering algorithms work as expected and designed; and results of the adaptive interpolation algorithm are very good. Compared with using the complete data set, interpolation results using approximately 1/2 of the data have acceptable errors; for the malleus, the interpolation error and difference of 3D model volume is 1.80% and 0.24%. The algorithm successfully and automatically generated the missing structure of the human temporal bone.

6.3 Applications of CT Results to Histological Sections

Histological sections were first provided to us. Because there are severe distortion and displacement issues, there is no gold standard to evaluate methods developed for histological sections. Therefore, a different dataset consisting of CT scans was provided to assist in the development of methods and algorithms. Because the CT dataset is complete, the performance and accuracy of the methods can be measured. Interpolation and 3D reconstruction results for the CT dataset can reach a target accuracy using only a portion of available data. This gives suggestions for how the methods and algorithms tested on the CT dataset can be applied to work on histological sections.

The results of CT scans can be related to histological sections. The adaptive interpolation results of the CT dataset indicate how many images are required, using the adaptive interpolation algorithm, in order to obtain a satisfactory and accurate generation of missing structure. The results also provide location information which shows that some regions of the z dimension need more histological cross-sections and some other places need less, because the incus and the malleus have different levels of details in the z direction.

Non-uniformly spaced histological sections would cause an issue for the Fourier transform algorithm since it requires uniformly spaced points. Another issue is that the image source dataset for histological sections is smaller than that for CT images. The Case 8486d has only 41 slices to choose from, which would mean that the adaptive algorithm would run out of data.

6.4 Future Work

This thesis has suggestions and valuable information for future 3D reconstruction using histological sections. 3D models of the incus and the malleus, reconstructed using the CT dataset, may serve as a basis of the model. In future research on 3D reconstruction using histological sections, our 3D models may work as reference models. For example, non-rigid registration for deformed histological sections can use 3D models from the CT dataset as a reference. The data usage information is valuable for 3D reconstruction of histological sections to indicate which locations need more data and which locations need less data. There may be a match between 3D models from the CT dataset and the future models from histological sections. Furthermore, templates of the incus and the malleus can be made using our results. In summary, 3D models from the CT dataset and the data usage information should provide improvement on registration and 3D reconstruction for histological sections.

Future work related to methods and algorithms of this thesis is:

1. The accuracy of segmentation and registration for both CT datasets and histological sections should be improved.

2. Refinement of the contour finding algorithm should be done. It is difficult to find contour points. The contour finding algorithm works relatively accurately and connects well to filtering and interpolation algorithms, but the method needs further improvement. Assigning contour points based on the center of the ROI and angles from the center was one option. Other alternatives can be studied. The error rate may be reduced by a better method.

3. There is possible refinement of Gaussian filtering. We realized that Gaussian filtering could be a very large task, and it alone can be a research topic. The development of GLPFs could be refined. For example, a 2D GLPF could be built for filtering both on slice and z direction contour points at the same time.

4. The error checking in the adaptive interpolation algorithm can be improved. The approximate data errors on slice and in the z direction can be studied further.

5. For interpolation and 3D reconstruction of each model, the locations that need more data points can be analyzed and studied. The studied result would provide useful information to create templates for models for the incus and the malleus for example. It also can provide information for histological section acquisition, such as the need for a small/larger number of slices in certain places of the specimen. The algorithm may be able to find out how many histological sections are enough to be interpolated for 3D reconstruction.

6. CT images have uniformly spaced and complete datasets. However, the issue with histological sections is that the datasets are non-uniformly spaced. For

future application on histological sections, the algorithm needs a more robust form that could encompass such an issue.

Our research has explored registration, segmentation, contour finding, filtering, interpolation and 3D reconstruction. The research is a proof of concept using a CT dataset. In future, our programs need to be tested on other CT datasets. Refinement needs to be done to make the programs solid, robust, accurate, and efficient. Then, the methods could be applied to histological sections with modifications and adjustments.

Bibliography

[1] Akerkar, R., Lingras, P., 2008, *"Building an intelligent web: Theory and practice: Chapter 5 - Clustering"*, Jones and Bartlett Publishers, pp. 177-204.

[2] Amira Software, Visage Imaging, Inc., Berlin, Germany. Website: http://www.amira.com/.

[3] Auer, M., Regitnig, P., Holzapfel, G.A., 2005, *"An automatic nonrigid registration for stained histological sections"*, IEEE Transactions on Medical Imaging, vol. 14, no. 4, pp. 478-486.

[4] Baggio, D.L., 2007, *"GPGPU Based Image Segmentation Livewire Algorithm Implementation"*, Thesis of Master in Science, Technological Institute of Aeronautics, Sao Jose dos Campos, Brazil.

[5] Bao, P., Zhang, L., Wu, X., 2005, *"Canny edge detection enhancement by scale multiplication"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 9, pp. 1485 - 1490.

[6] Black, M., Sapiro, G., Marimont, D., Heeger, D., 1998, *"Robust anisotropic diffusion"*, IEEE Transactions on Image Processing, vol. 7, pp. 421-432.

[7] Bookstein, F. L., 1989, *"Principal warps: thin-plate splines and the decomposition of deformations"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 6, pp. 567-585.

[8] Braumann, U.D., Kuska, J.P., Einenkel, J., Horn, L.C., Loffler, M., Hockel, M., 2005, *"Three-dimensional reconstruction and quantification of cervical carcinoma invasion fronts from histological serial sections"*, IEEE Transactions on Medical Imaging, vol. 24, no. 10, pp. 1286-1307.

[9] Brown, L.G., 1992, *"A survey of image registration techniques"*, ACM Computing Surveys, vol. 24, no. 4, pp. 325-376.

[10] Butner, H., Fovargue, A., Giovanetti, K., Lucatorto, L., Niculescu, G., ONeill, T., Utter, B., 2009, *"Physics 140L Laboratory Manual"*, James Madison University, Harrisonburg, VA.

[11] Canny, J., 1986, *"A computational approach to edge detection"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, pp. 679-698.

[12] Dawant, B.M., 2002, *"Non-rigid registration of medical images: purpose and methods, a short survey"*, Proceedings of IEEE International Symposium on Biomedical Imaging, pp. 465-468.

[13] De Boor, C., 1978, *"A representation for piecewise polynomial functions"*, in: A practical guide to splines, New York: Springer-Verlag, pp. 85-96.

[14] Demigny, D., 2002, *"On optimal linear filtering for edge detection"*, IEEE Transactions on Image Processing, vol. 11, pp. 728-1220.

[15] Dijkstra, E.W., 1959, *"A note on two problems in connexion with graphs"*, Numerische Mathematik, vol. 1, pp. 269-271.

[16] Diniz,P.R.B.,Murta-Junior,L.O.,Brum,D.G.,De Arajo,D.B., Santos,A.C., 2010, *"Brain tissue segmentation using q-entropy in multiple sclerosis magnetic resonance images"*, Brazilian Journal of Medical and Biological Research. , vol. 43, no. 1, pp. 77-84.

[17] Ear & Auditory Research Laboratory (EAR-Lab), Department of Anatomy & Neurobiology Surgery, Dalhousie University, Halifax, Nova Scotia, Canada. Website: http://ear-lab.medicine.dal.ca/.

[18] Fitch, A., Kadyrov, A., Christmas, W., Kittler, J., 2005, *"Fast robust correlation"*, IEEE Transactions on Image Processing, vol. 14, no. 8, pp. 1063-1073.

[19] Forsythe G.E., Malcolm M.A., Moler C.B., *"Computer methods for mathematical computations"*, Prentice Hall, pp. 19-77.

[20] Friedland, G., Jantz, K., Rojas, R., 2005, *"SIOX: Simple interactive object extraction in still images"*, Proceedings of the IEEE International Symposium on Multimedia (ISM2005), Irvine, California, USA, pp. 253-259.

[21] Friedland, G., Jantz, K., Lenz, T., Rojas, R., 2006, *"Extending the SIOX algorithm: Alternative clustering methods, sub-pixel accurate object extraction from still Images, and generic video segmentation"*, Technical Report B-06-06, Department of Computer Science, Free University of Berlin.

[22] Friedland, G., Jantz, K., Lenz, T., Wiesel, F., Rojas, R., 2007, *"Object cut and paste in images and videos"*, International Journal of Semantic Computing, World Scientific, USA, vol. 1, no 2, pp 221-247

[23] Fritsch, F. N.; Carlson, R. E., 1980, *"Monotone piecewise cubic interpolation"*, SIAM Journal on Numerical Analysis, vol. 17, pp. 238-246.

[24] Gaffling, S., Jager, F., Daum, V.. Tauchi, M., Lutjen-drecoll, E., 2009, *"Interpolation of Histological Slices by Means of Non-rigid Registration"*, Conference: Bildverarbeitung fur die Medizin, pp. 267-271.

[25] Gonzalez, R.C., Woods, R.E., 2002, *"Digital Image Processing"*, Second Edition, Prentice Hall.

[26] Gonzalez, R.C., Woods, R.E., 1993, *"Digital Image Processing"*, Prentice Hall.

[27] Goshtasby, A., 1988, *"Registration of image with geometric distortion"*, IEEE Transactions on Geoscience and Remote Sensing, vol. 26, no. 1, pp. 60-64.

[28] Guimond, A., Roche, A., Ayache, N.and Meunier,J., 2001 *"Three-dimensional multimodal brain warping using the demons algorithm and adaptive intensity corrections"*, IEEE Transactions on Medical Imaging, vol. 20, no. 1, pp. 58-69.

[29] Harder, R. L., Desmarais, R. N., 1972, *"Interpolation using surface splines"*, Aircraft, vol. 9, no. 2, pp. 189-191, 1972.

[30] Hellier, P., Barillot, C., 2003, *"Coupling dense and landmark-based approaches for nonrigid registration"*, IEEE Transactions on Medical Imaging, vol. 22, no. 2, pp. 217-227.

[31] Hou, H.S., Andrews, H.C., 1978, *"Cubic splines for image interpolation and digital filtering"*, IEEE Transaction on Acoustics, Speech and Signal Processing, vol. 26, no. 6, pp. 508-517.

[32] Hsu, L.Y., Loew, M.H., 2001, *"Fully automatic 3D feature-based registration of multi-modality medical images"*, Image and Vision Computing, vol. 19, pp. 75-85.

[33] Jeong H., Kim, C.I., 1992, *"Adaptive determination of filter scales for edge detection"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 5, pp. 579-585.

[34] Ji, J.X., Pan, H., Liang, Z., 2003, *"Further analysis of interpolation effects in mutual information-based image registration"*, IEEE Transactions on Medical Imaging, vol. 22, no. 9, pp. 1131-1140.

[35] Johnson, H.J., Christensen, G.E., 2002, *"Consistent landmark and intensity-based image registration"*, IEEE Transactions on Medical Imaging, vol. 21, no. 5, pp. 450-461.

[36] Jones, C., Sun, Q., Gan, R. Z., 2002, *"Computer-aided 3-dimensional modeling of human ear"*, Engineering in Medicine and Biology, 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society, vol. 1, pp. 266.

[37] Ju, T., Warren, J., Carson, J., Bello, M., Kakadiaris, I., Chiu, W., Thaller, C., Eichele, G., 2006, *"3D volume reconstruction of a mouse brain from histological sections using warp filtering"*, Journal of Neuroscience Methods, vol. 156, no. 1-2, pp. 84-100.

[38] Kahaner D., Moler C., Nash S., 1989, *"Numerical Methods and Software"*, Prentice Hall Series in Computational Mathematics.

[39] Kennedy,D.N., Filipek, P.A., Caviness, V.R., 1989, *"Anatomic segmentation and volumetric calculations in nuclear magnetic resonance imaging"*, IEEE Transaction on Medical Imaging, vol. 8, no. 1, pp. 1-7.

[40] Koplowitz, J., Greco, V., 1994, *"On the edge location error for local maximum and zero-crossing edge detectors"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 12, pp. 1207-1212.

[41] Lazebnik, R.S., Lancaster, T.L., Breen, M.S., Lewin, J.S., Wilson, D.L., 2003, *"Volume registration using needle paths and point landmarks for evaluation of interventional MRI treatments"*, IEEE Transactions on Medical Imaging, vol. 22, no. 5, pp. 653-660.

[42] Lehmann, T.M., Gonner, C., Spitzer, K., 1999, *"Survey: Interpolation methods in medical image processing"*, IEEE Transaction on Medical Imaging, vol. 18, no. 11, pp. 1049-1075.

[43] Li, Q., 2009, *"Rhythmic analysis of motion signals for music retrieval"*, Master of Applied Science Thesis, Saint Marys University, Halifax, Canada.

[44] Likar, B., Pernus, F., 2001, *"A hierarchical approach to elastic registration based on mutual information"*, Image and Vision Computing, vol. 19, pp. 33-44.

[45] Luo, F., 2007, *"Wavelet-based registration and segmentation framework for the quantitative evaluation of hydrocephalu"*, Master of Applied Science Thesis, Saint Marys University, Halifax, Canada.

[46] MacQueen, J.B., 1967, *"Some methods for classification and analysis of multivariate observations"*, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, vol. 1, pp. 281-297.

[47] Maes F., Collignon, A., Vandermeulen, D., Marchal, G., Suetens, P., 1997, *"Multimodality image registration by maximization of mutual information"*, IEEE Transactions on Medical Imaging, vol. 16, no. 2, pp. 187-198.

[48] Maintz, J.B.A., Viergever, M.A., 1998, *"A survey of medical image registration"*, Medical Image Analysis, vol. 1, pp. 1-36.

[49] Marr, D., Hildreth, E., 1980, *"Theory of edge detection"*, Proceedings of the Royal Society, vol. 207, pp. 187-217.

[50] McInerney, T., Terzopoulos, D., 1996, *"Deformable models in medical image analysis"*, Medical Image Analysis, vol. 1, no. 2, pp. 91-108.

[51] Jean-Pierre Moreau's Website of Numerical Analysis Sources. Website: http://jean-pierre.moreau.pagesperso-orange.fr/.

[52] Mortensen, E.N., Barrett, W.A., 1995, *"Intelligent scissors for Image Composition"*, SIGGRAPH 95. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, New York, NY, USA: ACM Press, pp. 191-198.

[53] Mortensen, E.N., Barrett, W.A., 1998, *"Interactive segmentation with intelligent scissors"*, Graphical Models and Image Processing, vol. 60, no. 5, pp. 349-384.

[54] Noblet, V., Heinrich, C., Heitz, F., Armspach, J.P., 2006, *"Retrospective evaluation of a topology preserving non-rigid registration method"*, Medical Image Analysis, vol. 10, no. 3, pp. 366-384.

[55] Orchard, J. , 2005, *"Efficient global weighted least-squares translation registration in the frequency domain"*, Image Analysis and Recognition (ICIAR), vol. 3656, pp. 116-124.

[56] Orchard, J., 2007, *"Efficient least squares multimodal registration with a globally exhaustive alignment search"*, IEEE Transactions on Image Processing, vol. 16, no. 10, pp. 2526-2534.

[57] Wong, A., Bishop, W., Orchard, J., 2006, *"Efficient multi-modal least-squares alignment of medical images using quasi-orientation maps"*, Proceedings of international conference on image processing, computer vision, and pattern recognition, pp. 66-73.

[58] Penney, G.P., Weese, J., Little, J.A., Desmedt, P., Hill, D.L.G., Hawkes, D.J., 1998, *"A comparison of similarity measures for use in 2-D-3-D medical image registration"*, IEEE Transactions on Medical Imaging, vol. 17, no. 4, pp. 586-595.

[59] Periaswamy, S.,and Farid,H., 2003, *"Elastic registration in the presence of intensity variations"*, IEEE Transactions on Medical Imaging, vol. 22, no. 7, pp. 865-874.

[60] Pham, N., Morrison, A., Schwock, J., 2007, *"Quantitative image analysis of immunohistochemical stains using a CMYK color model"*, Diagnostic Pathology, vol. 2, no. 8, pp. 1-10.

[61] Pluim, J.P.W., Maintz, J.B.A., Viergever, M.A., 2000, *"Image registration by maximization of combined mutual information and gradient information"*, IEEE Transactions on Medical Imaging, vol. 19, no. 8, pp. 809-814.

[62] Pluim, J.P.W., Fitzpatrick, J.M., 2003, *"Image registration"*, IEEE Transactions on Medical Imaging, vol. 22, no. 11, pp. 1341-1343.

[63] Pluim, J.P.W., Maintz, J.B.A., Viergever, M.A., 2003, *"Mutual information based registration of medical images: a survey"*, IEEE Transactions on Medical Imaging, vol. 22, no. 8, pp. 986-1004.

[64] Pukelsheim, F., 1994, *"The three sigma rule"*, The American Statistician, vol. 48, no. 2, pp. 88-91.

[65] Rusinkiewicz, S., Hall-Holt, O., Levoy, M., 2002, *"Real-time 3D model acquisition"*, ACM Transactions on Graphics, Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, vol. 21, no. 3.

[66] Sankar, P.V., Ferrari, L.A., 1998, *"Simple algorithms and architecture for B-spline interpolation"*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 10, pp. 271-276.

[67] Schnabel, J.A., Tanner, C., Castellano-Smith, A.D., Degenhard, A., Leach, M.O., Hose, D.R., Hill, D.L.G., Hawkes, D.J., 2003, *"Validation of nonrigid im-*

age registration using finite-element methods: application to breast MR images", IEEE Transactions on Medical Imaging, vol. 22, no. 2, pp. 238-247

[68] Shannon, C.E., 1948, *"A mathematical theory of communication"*, Bell System Technical Journal, vol 27, pp. 379-423, 623-656.

[69] Shapiro, L.G., Stockman, G.C., 2002. *"Computer Vision"*, Prentice Hall.

[70] Shen, S., Szameitat, A J., Sterr A., 2008, *"Detection of infarct lesions from single MRI modality using inconsistency between voxel intensity and spatial location– a 3-D automatic approach"*, IEEE Transactions on Information Technology in Biomedicine, vol. 12, no. 4, pp. 532-540 .

[71] Shin, M.C., Goldgof, D.B., Bowyer, K.W., Nikiforou, S., 2001, *"Comparison of edge detection algorithms using a structure from motion task"*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 31, pp. 589-601.

[72] Shirley, P., 2005, *"Fundamentals of Computer Graphics"*, Second Edition, A K Peters, Wellesley, Massachusetts.

[73] Skouson, M. B., Guo, Q., Liang, Z. P., 2001, *"A bound on mutual information for image registration"*, IEEE Transactions on Medical Image, vol. 20, no. 8, pp. 843-846.

[74] Studholme, C., Hill, D.L.G., Hawkes, D.J., 1999, *"An overlap invariant entropy measure of 3D medical image alignment"*, IEEE Transactions on Medical Imaging, vol. 17, no. 4, pp. 586-595.

[75] Sun, Z.Y., 2005, *"Using computer vision techniques on CT scans to measure changes in ventricular volume to aid in the diagnosis of hydrocephalus"*, Master of Applied Science Thesis, Saint Mary's University, Halifax, Canada.

[76] Tagare, H.D., DeFigueiredo, R.J.P., 1990, *"On the localization performance measure and optimal edge detection"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 12, pp. 1186-1190.

[77] Takagi, A., Sando, I., 1988, *"Computer-aided three-dimensional reconstruction and measurement of the vestibular end-organs"*, Otolaryngol Head Neck Surg, no. 98, pp. 195-202.

[78] Takagi, A , Sando, I., Takahashi, H., 1989, *"Computer-aided three-dimensional reconstruction and measurement of semicircular canals and their cristae in man"*, Acta Otolaryngol, no.107, pp. 362-365.

[79] Takahashi, H., Sando, I., 1992, *"Stereophotography of computer-aided three dimensional reconstructions of the temporal bone structures"*, Otolaryngol Head Neck Surg, no. 106, pp. 110-113.

[80] Taylor, J.R., 1997, *"An introduction to error analysis: the study of uncertainties in physical measurements"*, 2nd Edition, University Science Books.

[81] Thevenaz, P., Ruttimann, U.E., Unser, M., 1998, *"A pyramid approach to subpixel registration based on intensity"*, IEEE Transaction on Image Processing, vol. 24, no. 1, pp. 27-41.

[82] Unser, M., Aldroubi, A., Eden, M., 1993, *"B-spline signal processing: Part II-Efficient design and applications"*, IEEE Transaction on Signal Processing, vol. 41, no. 2, pp. 834-848.

[83] Unser, M., Aldroubi, A., Eden, M., 1993, *"B-spline signal processing: Part I-Theory"*, IEEE Transaction on Signal Processing, vol. 41, no. 2, pp. 821-833.

[84] Wang, H.B., Northrop, C., Burgess, B., Liberman, M.C., Merchant, S.N., 2006, *"Three-dimensional virtual model of the human temporal bone: A stand-alone, downloadable teaching tool"*, Otology and Neurotology, vol. 27, no. 4, pp. 452-457.

[85] Zagorchev, L., Goshtasby, A., 2006, *"A comparative study of transformation functions for nonrigid image registration"*, IEEE Transactions on Image Processing, vol. 15, no. 3, pp. 529-538.

Appendix A

Registration Supporting Materials

A.1 Review of Non-rigid Registration for Histological Sections

A non-rigid registration method with a deformable transformation is required especially when images of soft biological tissue need to be aligned to each other. Several options for composing non-rigid registration are rigid and affine transformations, scaling and skew, spline interpolation between point landmarks, warping, multiple affine transformations, mechanical models, and rigid bodies plus warping and fluid flow [50].

However, fairly limited work has been done in reviewing non-rigid registration. Most literature only covers rigid image registration. Auer et al. [3] stated that most registration methods are only able to perform rigid-body motion and are sensitive to noise and artifacts. Non-rigid registration is rarely performed in clinical applications, because there is rarely any gold standard for evaluating the computed registration. Issues associated with non-rigid registration are how to define the accuracy of the registration results, how to validate the value of elastic coefficient, how to visualize the registration results, and how to find robust features and similarity functions.

There has been an increase in using machine and complex computer algorithms in performing image diagnoses, but the computerized process is sensitive to artifacts and requires that the images are related to each other. Artifacts must be removed or corrected and the images must be related to each other so that image comparison is meaningful and objective [3].

Since elastic deformations in the specimens occur during the preparation, rigid registration methods cannot be applied anymore. A non-rigid registration method is required especially when images of soft biological tissue need to be aligned to each other. Otherwise, the images are not well suited for comparison purposes, and many available registration methods may fail.

Problems in registration may also arise if there are artifacts in images. Artifacts produced during the specimen cutting process cannot be registered by using a continuous transformation function. Overlapping regions are not necessarily a problem for a rigid registration. Nevertheless, problems may exist in a non-rigid registration, because they occur locally and may cause undesirable local deformations [3]. Therefore, it is important that such locally mis-registered points can be filtered out. Simple consistency tests can detect such outliers. Using only a few constraints during the consistency tests can ensure an accurate local registration [3].

Affine transformations are widely used to correct scaling errors or skew in images, but they are unable to represent tissue deformation, and unable to represent differences between subjects. They are often involved in aligning functional images from different subjects (cohort studies) [50].

Problems associated with non-rigid registration are how to define the registration accuracy, how to validate the value of elastic coefficient, how to visualize the registration result, and how to find robust features and similarity function. Furthermore, there is rarely any gold standard for evaluating the computed registration. Therefore, non-rigid registration is currently rarely performed in clinical applications [30].

Appendix B

Segmentation Supporting Materials

B.1 K-means Clustering

The idea of k-means clustering is to:

1. Select k points to be the starting points for the centroids of the k clusters.
2. Assign each object to the centroid closest to the object, forming k exclusive clusters of examples.
3. Calculate new centroids of the clusters. Take the average of all attribute values of the objects belonging to the same cluster.
4. Check if the cluster centroids have changed their coordinates. If yes, repeat from the Step 2. If no, cluster detection is finished and all objects have their cluster memberships defined.

As it shows in Figure B.1, the difference between the current clustering results and the previous one is tremendous. After cycles of running, the difference value is

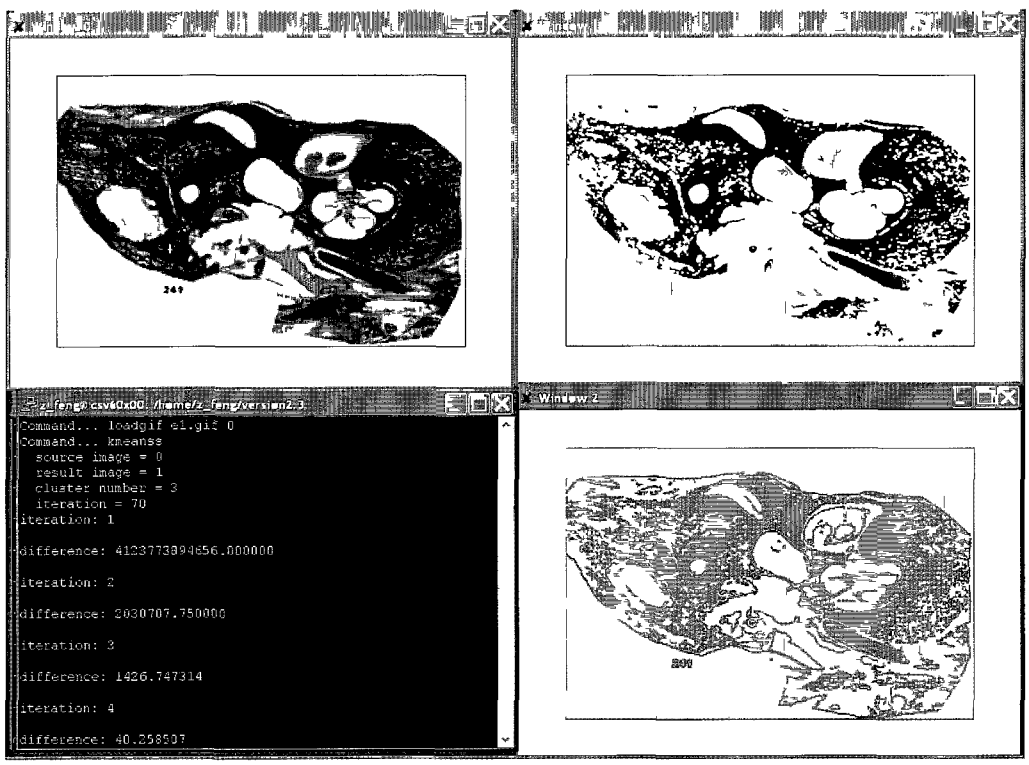

Figure B.1: Performing k-means clustering on a histological section image. From top left to bottom right (a, b, c, d): a. The input image; b. The gray scale result image with 3 clusters, c. The command field, d. The color result image

40.259, which is greatly reduced. After 31 cycles, the different does not change any more. Therefore, the result is generated, and the program can be terminated.

The results show that the method is useful in clustering different regions of interest into certain groups. In this project, it shows that the program can find three clusters efficiently. The method of initial setting of the centroids leads to good results, because k-means clustering is sensitive to initial condition, and different initial condition may produce different result of cluster. The algorithm may be trapped in the local optimum. If the cluster number is three, the initial centroids have pixel values of 0, 85, and 170. The setting leads to the better results than initializing randomly. Different cluster numbers as input have been tested. The results show that the program can find three clusters efficiently. Meanwhile, after approximately 32 cycles of running, the results can be generated.

k-means clustering has many weaknesses. The number of cluster, K, must be determined before hand. It is difficult to know which attribute contributes more to the grouping process since we usually assume that each attribute has the same weight.

B.2 Region Growing

A "traditional" region growing algorithm was implemented and tested. It has the following steps:

1. Select a rectangular patch of the ROI completely inside the bright part which needs to be segmented.

2. Calculate the mean value, avg, and standard deviation, σ, of the selected patch.

3. Check each pixel in the image and search for those with gray value of avg. During this step, the seed can be found. Its position in the original image is stored using a binary image with value 1 correspondingly.

4. For every position in the binary image with value 1, the algorithm checks the eight neighbor pixels in the original image (or four neighbor pixels in another version of the algorithm). If the difference between the gray level of these surrounding pixels and m is less than σ, the position of the neighboring pixel is stored in the binary image.

5. Each time a new pixel position is added to the binary image, the mean and standard deviation of the pixel values referenced by the binary image are recalculated.

6. Repeat the above two steps until no more neighboring pixel is added. At the same time, the result image is represented by a gray scale image with values 0 and 255.

Appendix C

Standalone GUI Programs

Several standalone GUI programs connecting a list of methods into a streamline are described in this Appendix. They were mainly developed using .NET C# programming.

C.1 Transformation Programs

The Transform1 program does rigid transformations on an image. It loads an image and takes input of translation and rotation parameters from users. The screen shot of the program is shown in Figure 5.2.

The Transform2 program does the affine transformation on an image. Besides translation and rotation, it does scaling and shearing. In Figure C.1, slice No.51 is loaded and displayed in a transformation program that we developed in C#. It shows a result of an affine transformation that has been performed. Originally, the bottom left corner of the image is in the origin (0, 0) point. It includes translation, rotation, scaling and shearing to obtain this result image.

C# programs were developed to understand and study image registration and transformation techniques. Amira is used for rigid registration on histological sec-

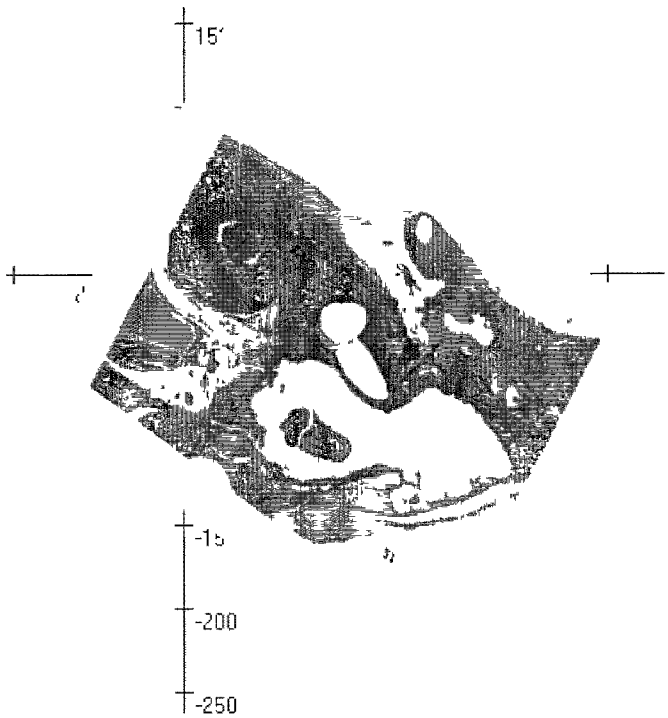

Figure C 1: Affine transformation on a histological section, the Transform2 program.

tions.

C.2 Rename1 Program

The Rename1 program renames a list of images to GIF images using original names, as in PSD files. To bring original names back to files, Rename1 was developed to:

1. Read a list of images with original names, and have an option to automatically save them to GIF images.

2. Save the original names in a file that can be used for renaming images in any later step.

3. Load another list of images and rename them with the original names.

4. Convert them to GIF images for processing in CVlab.

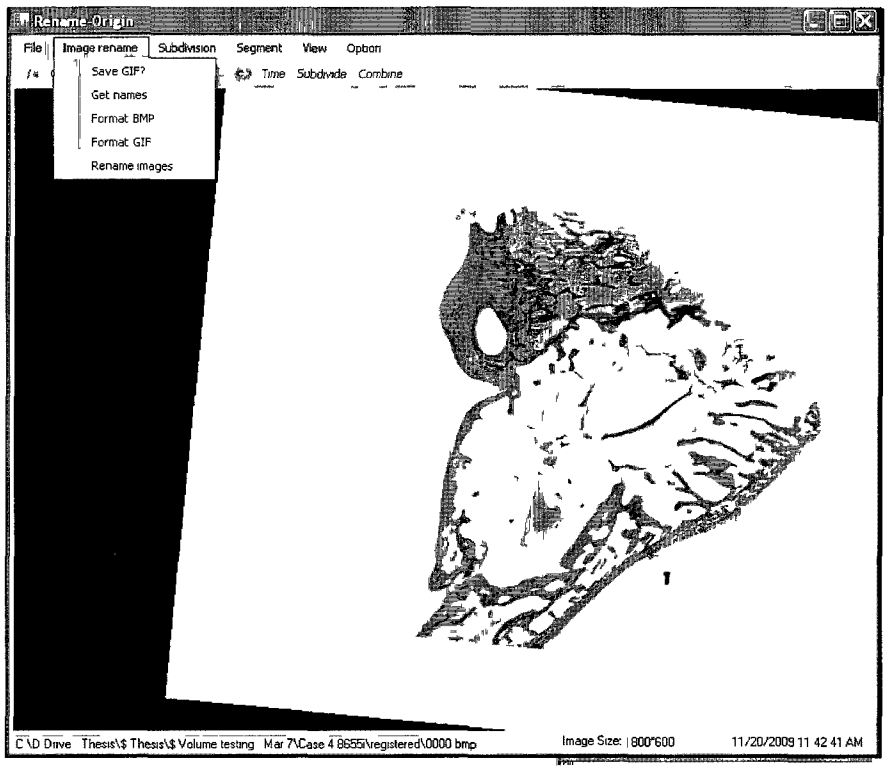

Figure C.2: A screenshot of the program Rename1.

The program allows loading all images by one click on only one image in the folder. It uses array-list data type to store the image list, and another array-list to store all image names. Since image names are strings and need to be sorted, when registered images are saved from Amira, they need to be 001, 002, etc., instead of 1, 2, etc. Original image names also should be named 001, 016, 123, etc., instead of 1, 16, 123, etc. Only if these string operations are handled, correct names can be added to image files accordingly in an order.

The program GUI is shown in Figure C.2:

C.3 Rename2 Program

We developed another C# program, Rename2. The GUI is shown in Figure C.3.

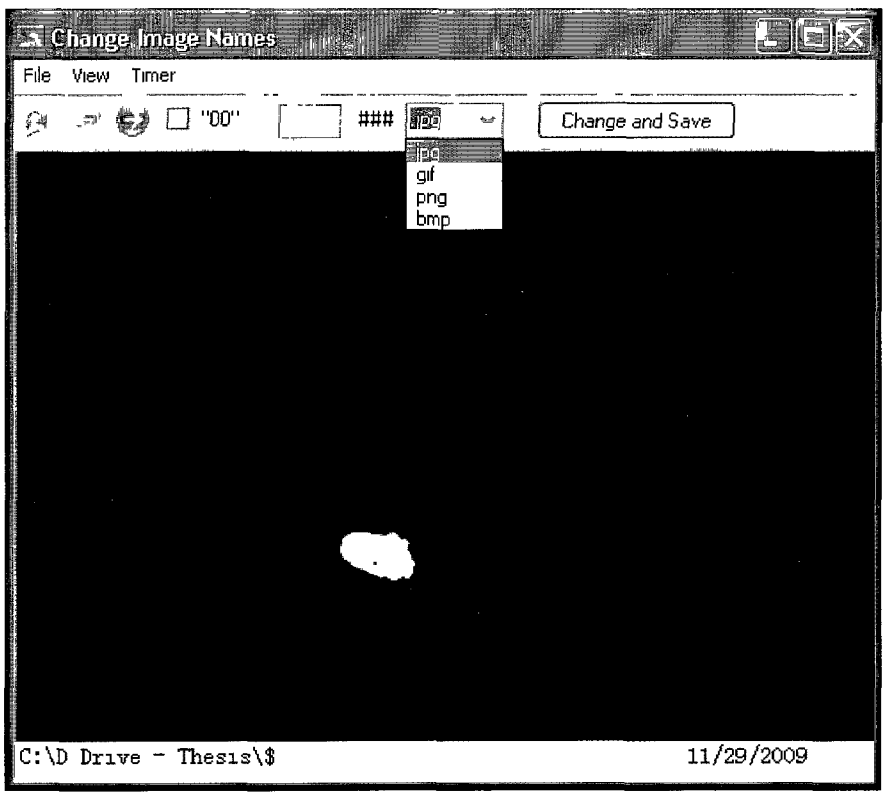

Figure C.3: A screenshot of the program Rename2.

Rename2 is developed for more general purposes:

1. Image conversion among formats, e.g., JPG, GIF, PNG, and BMP
2. Adding or removing letters in front of all names
3. Putting "0" or "00" before all image names

C.4 SelectObject Program

The SelectObject program runs in the following steps:

1. At program start, automatically load all images and pop-up an Excel sheet that is for data analysis.
2. Select one object in the first image using a patch, shown in red rectangle, inside the ROI
3. Click NEXT or press ENTER and bring up the next image after the selection is satisfactory
4. The image shows up and display the previous selection with the selected patch (shown yellow) from the last image
5. If the yellow patch applies for the current image, click NEXT to the next image; if not, update the selection, and click NEXT
6. Iterate the above two steps until the last image. The program pops up a "Complete" message box, after all images are selected
7. The CVlab script is written into a file, storing selection coordinates x and y in a list of commands.

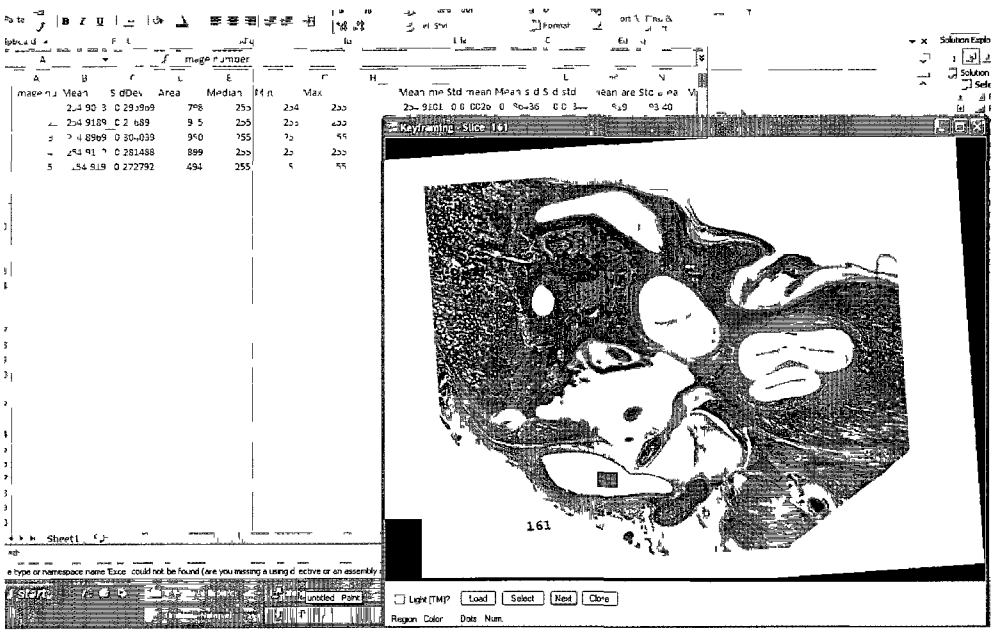

Figure C.4: A screenshot of a GUI program for selecting an object (ROI)

8. For each selection, store a list of attribute values, and calculation in an Excel sheet that started in the beginning Stored attributes are seven columns of data in the selected patch region, e.g., image number, mean, standard deviation, median, min, max pixel, and selected area values.

The program GUI is shown in C.4, with the synchronized Excel sheet behind: The program has several advanced features and advantages:

1. It has simple and user-friendly GUI design. Selection of 50 images can be done in less than one minute manually

2. Scripting for CVlab, Excel sheet writing and calculation is fully automatic. Besides seven columns of values, the Excel sheet can update the mean and standard deviation values for them. It is convenient for data analysis, particularly for robustness and accuracy analysis, e g., sensitivity to selection offset.

3 The program finds and stores all target images in a list without manual oper-

ation so that testing several times become convenient. It also knows when the list is finished.

4. It saves data in a newly created folder named by current time, e.g., "Selection 2009-3-18 0-49-49". It helps organizing when testing for many times.

5. An advantage of the current algorithm (especially the region growing part) is that it works for both light and dark regions.

C.5 Advantages of GUI programs

Above GUI programs were developed to assist simple and automatic data transmission between Windows and Linux systems, as well as different software applications.

Particularly, Amira, a commercial software package with advanced 3D visualization features, was used for the registration and final 3D reconstruction step. CVlab, a Linux program that has many basic image processing and computer vision routines, was applied to develop a portion of important algorithms. Several GUI applications were developed to integrate all steps into one streamline. They mainly provide features, such as script writing, data passing, image renaming and format conversion.

The reason not having all programs in Windows A reason to keep segmentation in CVlab is that it uses a number of existing algorithms in CVlab, e.g., histogram, mean and standard deviation calculations, sobel algorithm, a variety of filtering, normalization and so on.

Segmentation algorithms do not compile, unless all of C programs and their subroutines are ported to c# programs. CVlab has been useful, since there are many basic algorithms available. There would be a large amount of work for porting segmentation to Windows. Then, there would be integration work needed to connect segmentation

programs in Windows and other c# programs. Also, the code structure /architecture will be changed. There should be more issues than one could see too.

Besides, for testing one image step, copying the images between CVlab and c# program only need less than 1 minute. In testing, most algorithms need several seconds to complete. There is not a particular step that is very slow.

Such a layout of algorithm components in a streamline seems suitable in our applications. The GUI tools usually require two to three clicks, but provide convenient connectivity for steps in Amira, Linux CVlab, and main c# programs.

Appendix D

Interpolation Supporting Materials

Table D.1 lists various PCHIP implementations in C++, python and Fortran77. They are analyzed for porting and then integrating into C#.

An actual dataset sample (CT images, incus, No. 36 to No. 66) was tested. Every fifth image contour points were interpolated using both C# program (calling Hugin C++ pchip routine) and Matlab. The interpolants as in Figure D.1 match perfectly. Besides, several sample interpolated values are in Table D.2 to show they are equal.

Name	Language	Author/Resource
PCHIP	Fortran77	Fred Fritsch [23]
spline.cpp	C++	John Burkardt
MonotCubicInterpolator.cpp/hpp	C++	public.ict.sintef.no
pychip.py	Python	Chris Michalski

Table D.1: Various PCHIP implementations

z Value	C# Pchip DLL	Matlab Pchip Routine
39	263.976	263.976
40	263.232	263.232
42	263.8	263.8
43	265.8	265.8
49	277.608	277.608
54	277.056	277.056
55	276.312	276.312

Table D.2: Sample interpolated values show equal results of Matlab and C# program, CT incus

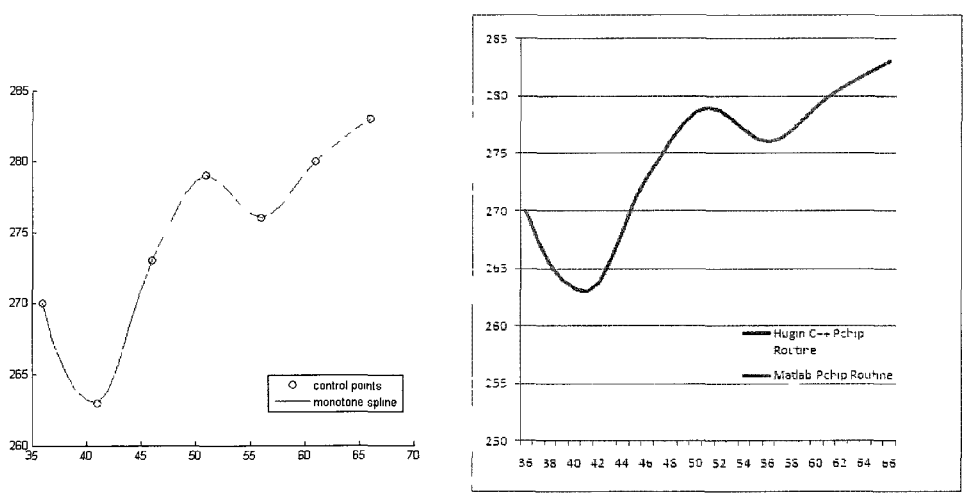

Figure D.1: Verification test on Hugin C++ Pchip Routine using Matlab, CT incus test. From left to right: a. matlab Pchip interpolation on control points; b: interpolants in Matlab and C# program calling Hugin C++ Pchip DLLs overlap completely.