Acute Asthma and Recovered Airway Tree Geometry Modeling and CFD Simulation

A thesis submitted in fulfillment of the requirements for the degree of Master of Engineering

Yong Ye

B. Eng.

School of Aerospace, Mechanical and Manufacturing Engineering

Science, Engineering and Technology Portfolio

RMIT University

August 2008

Author's Declaration

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at RMIT or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at RMIT or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

……

Yong Ye

Abstract

This study focuses primarily on the development of modeling approaches for the reconstruction of lung airway tree and arterial vessel geometry models which will assist practical clinical studies. Anatomically-precise geometric models of human airways and arterial vessels play a critical role in the analysis of air and blood flows in human bodies. The generic geometric modeling methods become invalid when the model consists of both trachea and bronchioles or very small vessels. This thesis presents a new region-based method to reconstruct the entire airway tree and carotid vessels from point clouds obtained from CT or MR images. A novel layer-by-layer searching algorithm has been developed to recognize the branches of the airway tree and arterial vessels from the entire point clouds. Instead of applying a uniform accuracy on all branches regardless of the number of available points, the surface patches on each branch are constructed adaptively based on the number of available elemental points, which leads to the elimination of distortions occurring at small bronchi and vessels.

Acute asthma is a serious disease of the respiratory system. To understand the difference in geometry and airflow patterns between acute asthma affected and recovered airway trees, a comparison study has been conducted in this research. Two computational models of the airway tree up to six generations deep were reconstructed from computed tomography (CT) scans from a single patient. The first scan was taken one day after an acute asthma episode and the second scan was taken thirty days later when the patient had recovered. The reconstructed models were used to investigate the effects of acute asthma on realistic airway geometry, airflow patterns, pressure drops, and the implications for targeted drug delivery. Comparisons in the geometry found that in general the right side of the airway is larger in diameter than the left side. The recovery of the airway was most significant in the severely asthma affected regions. Additionally the right airway branches exhibited greater dilation after recovery in comparison with the left airway especially from the fifth generation onwards. It was also found that bifurcation angles do not vary significantly between the two models, however small changes were observed which may be caused by the physical scans of

the patient being taken at different times. The inhalation effort to overcome airway resistance in the asthma affected model was twice as high as that for the recovered model. Local flow patterns showed that the changes in the airway had significant influence on flow patterns. This was especially true in the region where the airway narrowing was most severe.

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisors Dr. Songlin Ding and Professor Aleks Subic, for their encouragement, guidance, and support throughout my graduate studies. Sincere thanks to Professor Jiyuan Tu for his guidance and support on CFD in this research program. I would like to thank my family for their support throughout my education.

Finally, I would also like to thank my colleagues; Dr. Inthavoong Kiao, Marc Alexander, Dr. Zhaofeng Tian, Dr. Sherman Chi-Pok Cheung and Dr. Huafeng Li for providing invaluable technical guidance and joyful memories throughout the course.

Table of Contents

List of Figures

List of Tables

Nomenclature

A	cross-sectional area
CCB_i	constrained catchment basin for regional minimum
d	distance
D_h	equivalent hydraulic diameter
f	image
M_i	regional minimum
P	perimeter
P_i	point
S	surface patch
S_x	structuring element with origin x
V_i	constrained partial catchment for regional minimum
ν_g	viscosity
u_{ave}	mean velocity
$Wshed$	constrained watershed
$\bar{\pi}$	minikowski path
δ	distance bigger than zero.
ω	angular frequency of oscillation

Chapter 1
Introduction

1.1 Research Background

Modeling of the geometry and airflow within the respiratory system is essential in assessing respiratory health. This modeling includes two fields, geometry reconstruction and numerical simulation.

Three dimensional (3D) geometric reconstruction is a widely used technology in the aerospace, automotive and manufacturing industries. These models are mathematically expressed as 3D solid geometries in CAD/CAM (Compute Aided Design/Computer Aided Manufacturing) systems. In recent years the application of geometric modeling has been extended to the field of medical applications, such as quantification of tissue pathology, surgery simulation, and radiotherapy planning. Techniques in minimally invasive surgery, surgery performed remotely using robotic controls, and computer-assisted planning and rehearsal have also proved to be very effective. However acquisition of geometric models of object organs is a prerequisite for the application of the above techniques. With continuously improving technology, computers have been able to generate accurate 3-D images of entire body parts for surgery or treatment planning. For example, true 3-D images are produced by computer tomography (CT), magnetic resonance imaging (MRI), positron emission tomography (PET), and ultrasound imagery (USI). Currently many studies of the reconstruction of human organs have been performed based on the data obtained from the medical image types mentioned above. Some commercial software packages have recently become available in the market for the reconstruction of bronchi. However, there are many problems in the geometries generated by these packages. The models generated can not be utilized in CFD (Computational Fluid Dynamics) study directly. Extensive work must be done on the modification of these models to satisfy the CFD requirement, and that work is

extremely time-consuming. This consideration hinders the associated study of air flows in which large amount of tests should be run on different models.

Because of the undeveloped state of geometric modeling and CFD techniques the early research conducted on the physics of aerosols and airflows in the human upper airways has relied mainly on experimental studies. Experimentation has been used to obtain either velocity profiles or deposition efficiency. Generally, experimental studies can be categorized into two major approaches, using human cadavers to replicate airway models to study deposition efficiency and idealized airway models (e.g. made of smooth plastic tubes) to study velocity profiles using pressure probes or dye tracing. Although many of these in-vitro experiments provided important information, limitations were unavoidable. From a velocity profile perspective, the method of inserting the probe into a model to measure the flow can often affect the flow downstream and around the measurement point. This is especially noticeable when the cross-section of the airway tube is small. Moreover, only limited sample readings are extracted due to the cost and time required for experimental setup. Thus, the amount of information provided by experimental technique is not only restricted, but the accuracy of the results may be difficult to control. The more advanced method of using laser Doppler anemometry has been applied recently. This has resulted in improved accuracy for velocity profiles along occluded tubes without influencing the flow pattern. However, a disadvantage of this method remains. Some crucial flow velocity information, for example for the boundary layer region (near the wall) is extremely difficult to capture although the boundary layer flow is one of the major influences on particle deposition on the wall of the model. In terms of measuring deposition efficiency, there are two major approaches. One of the methods involves injecting a range of fluorescent polystyrene latex particles into a hollow silicone rubber cast of human airways, and then the cast is cut into segments. Fluorescent material in the particles deposited in the cast is extracted using ethyl acetate. The resultant solution is then filtered using a nylon membrane and the fluorescence content of the filtered solution is measured using a fluorescence spectrometer. The deposition fraction (i.e. efficiency) in each airway segment is calculated from this measurement. (Zhou and Cheng, 2005) Another method is to inject ferric oxide particles tagged with short-lived radioisotopes,

so that deposition activities within the cast can be measured in a scintillation detector system. (Schlesinger et al., 1977) Similarly, these experimental techniques require expensive equipment and extensive setup time. The major disadvantage of this type of approach is that the casts of the airway are replicated from a cadaver in which the shape and the angle of bifurcation of the airway are very likely to be distorted after the airway is taken out from a cadaver, compared to those of living human subjects. Loss of geometric detail may also occur during the replication process. Another limitation was declared by Schlesinger et al. (1977) where the large number of bronchi within the cast and limitations imposed by use of a short-lived isotope tag made a complete investigation of deposition within all branches of the cast impractical. Measurements are, therefore, made within a selected sample of bronchi.

Currently, the continuously enhancing capacity of computers and commercially available software packages help increasing numbers of scientists to contribute to this challenging area. CFD has been seen as an excellent technique for studying flow and particle behaviour within airways. One obvious advantage is the lower cost than for experimental methods. CFD can also incorporate more complicated flow and particle phenomena in a more complex airway model than with theoretical models, where simple, idealised models are often used. There have been many numerical studies related to flow profile, deposition efficiency and aerosol deposition patterns, where the close interactions between these three aspects have been researched. Another advantage is that CFD can provide clear insight into the characteristics of air flows and aerosol deposition which, in turn, may lead to better drug delivery techniques, better treatment and important information for clinical practitioners in assessing the health risks within the human airways associated with inhaled particles. From a research perspective, CFD can readily provide stable, high accuracy results in all aspects, which can be compared and validated with available experimental and theoretical results.

However, to the best of the author's knowledge there have been very few studies that compared the inhalation effort and air flow patterns between a realistic acute asthma model and the following recovered airway model (i.e. based on CT scan).

1.2 Motivations

Asthma is a chronic inflammatory disorder of the airways affecting mainly the medium-sized and small bronchi (Travis et al. 2002) characterized by a sudden or prolonged onset of airway narrowing. Long term effects of the disease can be characterized by variable degrees of tissue and structural remodeling in the airways that lead to a progressive loss of lung function (Pascual and Peters 2005; Vignola et al. 2000). During an acute asthma episode, the combination of bronchospasm, mucus plugging, and mucosal edema leads to increased airway resistance as the diameters of the airways are reduced. An increase in the inhalation effort is therefore required to obtain similar tidal volumes to those in unconstricted airways, which then leads to respiratory muscle fatigue. The increase in airway resistance also causes a lack of uniformity in ventilation throughout the lung causing hypoxemia from ventilation-perfusion mismatching. In addition the changes in the geometry and inhalation efforts will have a significant effect on drug delivery devices such as inhalers that are designed to deliver pharmacological agents to affected areas within the tracheobronchial tree. Therefore, the comparison of geometry and air flow pattern between asthma affected and following recovery lung airways is necessary to provide useful information to clinical researchers in assessing the pathogenic potential and may lead to innovation in inhalation therapies.

The availability of a 3D geometric model of real human airway trees is the prerequisite for performing comparative research about geometry and airflow pattern between asthma affected and following recovered human airway trees. Three dimensional (3D) geometric modeling is a well established technology used in aerospace, automotive and manufacturing industry. Many studies concerning the reconstruction of human organs have been performed based on the data obtained from CT or MR images, and some commercial software programs are available in the global market for research and clinical applications. For example, Pulmonary Workstation 2 (VIDA Diagnostic Inc and Materialise Inc) and Mimics (Materialise Group) can anatomically segment human airway trees from CT scans. However due to their complex geometric nature, many flaws emerge in the generated geometries when using the published algorithms or the commercial packages currently available in the market.

Extensive work is required to covert the geometries into workable numerical models. Consequently development of a new algorithm to generate human lung airway trees is very useful. Because the arterial vessels and airway tree are of similar geometric nature, tubular and tree-like, the geometry reconstruction algorithm developed for lung airway trees is also able to be used for arterial vessels.

There have been numerous studies of the air flow and deposition of aerosols in the respiratory system. Much work has been done to investigate the factors that affect the flow pattern and deposition of aerosols. The flow rate and secondary flow pattern are quite sensitive to the tube diameter and bifurcation configuration (Liu et. al, 2002, 2003). Particle properties, breathing conditions and lung geometry are known to be the main factors influencing the deposition of aerosols (Martonen et al., 1994; Balashazy et al., 1999; Lee et al., 2000; Comer et al., 2001; Darquenne, 2001; Liu et al., 2002; Hegedus et al., 2004; and Miguel et al., 2004). Different particle properties and breathing conditions have been studied extensively even though they did not use anatomically realistic airways from living humans to study aerosol deposition. These particle transport studies have provided additional detailed information of particle behavior in the lung airways and have shown that this behavior is inherently linked to the fluid flow patterns within the airways. Although most of these experimental and numerical works on flow in human airways were based on simplified, idealised airway models extracted from the early morphological studies completed by Weibel (1962) and Horsfield et al. (1971), there have been several attempts to explore the flow numerically in realistic airways which were based on computerized tomography (CT) scanner imaging (Perzl et al. 1996; Vial et al. 2005). Thus, computational fluid dynamic (CFD) techniques combined with CT reconstructed airway models have been found useful in simulating flow phenomena. However, most of previous studies considered only a straight smooth simplified airway or healthy airway, there have been few attempts to investigate the air flow pattern in realistic asthma affected airways. Therefore, this research was conducted to develop geometry modeling and CFD techniques to be used to study the differences of geometry and flow pattern between the acute asthma and following recovery lung airways.

1.3 Literature Review

1.3.1 Airway tree geometry modeling

The respiratory system is divided into the upper and lower airways, the upper extending from the nasal/oral cavity to the larynx, the lower from the larynx to the respiratory bronchioli and alveoli. The lower airways consist of the trachea and the main bronchi. The structure of human airways can be approximated to a network of repeatedly bifurcating tubes (Weibel, 1963). The position of a branch in relation to the stem branch can be described by Weibel generations which have been employed widely among researchers.

There are some features of the airway trees which must be noted during model construction and simulation. The trachea is a cylindrical tube typically 10 to 12 cm long, approximately half of which is extrathoracic and half intrathoracic. According to Weibel (1963) measurements, the diameter of the trachea is approximately 1.5 to 1.8 centimetres. The trachea wall was composed of 16 to 20 incomplete (C-shaped) rings of cartilage with fibrous and muscular tissue. The trachea divides into the two primary bronchi at the level of the fifth thoracic vertebra. There is an internal cartilaginous ridge at the point of bifurcation, named the carina.

The right bronchus is a shorter, wider tube than the left and the angle of branching from the trachea is only 20-30˚. Weibel (1963) measured the main bronchus to be 0.9 to 1.5 cm in diameter and 4 to 6 cm in length. Consequently any foreign bodies which enter the trachea are more likely to be inhaled into the right main bronchus. This may also influence the distribution of aerosol delivery to the lungs.

The branching pattern resembles that of a tree, with large branches giving rise to successively smaller branches in terms of length and diameter. This pattern of division varies and is probably genetically determined (Shannon and Deterding, 1997). Figure 1.1 shows

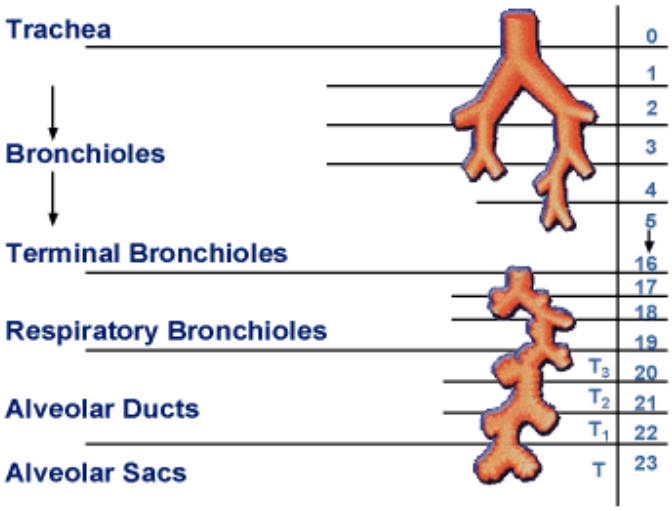


Figure 1.1 Branching generation of the airway (Cefalu, 2003)

how the lower airways are named and classified into using Weibel generations. Numbering begins at the stem branch and continues towards the peripheral branches. The trachea has the lowest generation number, and the next branch division, the main bronchus is one generation higher than the parent branch, trachea. This also applies to asymmetric trees in which terminal branches are classified in a range of generations. Branches which have different diameters and lengths can therefore be grouped together for simple classification.

The early airway tree geometry model was simplified and idealized (Weibel 1963). Weibel developed his airway tree model in 1963 based on measurements of the major airway branches. The airway geometry consisted of a symmetric in-plane double bifurcation and each airway was replaced by a round smooth pipe. Although the first four generations and 10% of the rest of the generations were measured precisely, because of the difficulty of measuring the small bronchioles, regular dichotomy and a mean branch length to diameter ratio were assumed to generate the remainder of the airway tree model. Therefore Weibel's model is of regular dichotomy and fixed length to diameter ratio (2005) used this model extensively in their studies of air-particle flow. The model reconstructed by Zhang et al based on Weibel symmetric model A's measurements is shown in Figure 1.2.

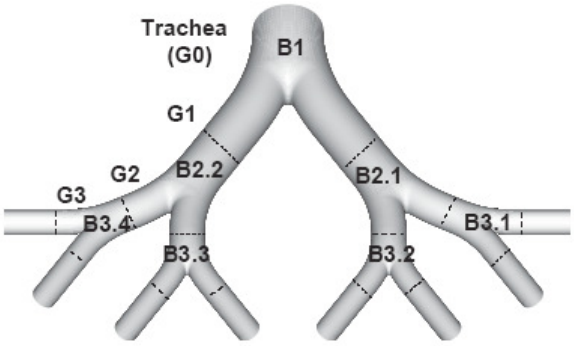


Figure 1.2 Symmetric airway model reconstructed by Zhang et al. (2005)

Horsfield and Cumming (1968) performed detailed measurements on a resin cast of the human lungs. 8,298 branches were measured from the trachea down to lobular branches. Based on their previous work, Horsfield et al. (1971) derived an asymmetric conducting airway model. The asymmetric model was based on the concept of regular asymmetry at different levels in the airway tree. Similar to Weibel (1963)'s work, they measured not only the diameters and lengths of branches, but they also obtained the flow distribution in terms of the percentage of tracheal flow, which later provided very useful reference in in comparison with other measurements of human airways. Erthruggen et al. (2005) employed Horsfield's Delta model in constructing their computational model as well as implementing the flow distributions measured by Horsfield et al. (1971) in their numerical simulation. The numerical model constructed by Erthruggen et al. (2005) based on Horsfield et al. (1971)'s measurements is shown in Figure 1.3

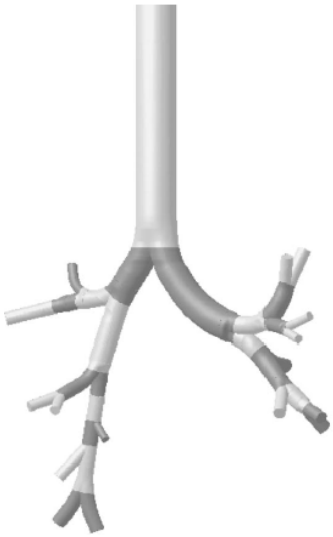


Figure 1.3 Airway model reconstructed by Erthruggen et al. (2005)

Both of above models have in-plane configuration which means that the second bifurcation units have the same symmetry planes as the first one. There are a few advantages of using a double bifurcation. One of them is running time. Based on the symmetry, only half of the geometry needs to be modeled. However the simplified model removed some useful features which have significant influence in airflow and particle deposition. For example Zhang & Finlay (2005) proved that trachea with cartilaginous rings would enhance particle deposition in the trachea for all inhalation rates and particle sizes when compared with smooth-walled trachea.

With the development of computer technology and medical imaging, the airway geometry reconstructions based on CT are becoming common place. Although all of the relevant 3-D measurements for objects of interest are available from medical imaging, it remains a difficult task to derive information about the characteristic properties (structures) of objects. Firstly, since direct interpretation and exploitation of large amounts of raw data is difficult, it is often convenient to segment image arrays into low-level entities that can be compared to higher-level entities derived from known representations of objects. For example, displaying 2-D cross-sectional images sequentially is no longer sufficient to establish a reliable

diagnosis or to prepare a critical therapy procedure. Moreover, the complexity of the anatomical structure can make their identification difficult, and the use of multi-modal complementary images requires accurate spatial registration. Therefore, segmentation, whose goal is to partition the raw image data into regions corresponding to meaningful anatomic structures, is a fundamental problem in medical imaging applications.

There are numerous algorithms developed to reconstruct the airway geometry from CT scans. These algorithms can be categorized into three types: (1) region-growing-based methods (Assimakopoulos et. al 1986, Aykac et. al 2003, Ballard et. al 1982 and Bashein 1994), (2) morphology-based methods (Bilgen 2000, Bloch 1993 and Boyden 1955), (3) fuzzy logic-based methods (Carraghan et, al 1990, Carvalho et.al 1999 and Wonkyu et. al 1998). 3D Region-growing-based methods simply use voxel connectivity and a threshold to identify regions, usually through 26-connectivity. These algorithms have been used extensively because of their simplicity, speed and flexibility. Their flexibility in terms of adding flexible rules based on the structural properties of the object has made them efficient segmentation methods. Morphology-based algorithms use grayscale morphological reconstruction to identify candidate airways on 2-D CT slices. After this initial segmentation, a bounded space dilation is applied to labelling connected airways and to reconstruct the 3-D airway tree, after which tree branch segments and tree branch points can be identified. Fuzzy-logic methods identify intrathoracic airway trees in two stages. In the first stage, airways are identified in individual 2-D slices of a 3-D volumetric CT data set. In the second stage, airway trees are constructed by context sensitive stacking of the segmented airways. Fuzzy logic plays the major role in segmenting airways in the 2-D image slices by utilizing anatomical information about airway morphology and vessel–airway relationships.

Although the methods mentioned above can be used in many medical applications, such as visualization, measurement and surgery planning, the generated model can not be used directly in post numerical simulation. These models are still stay in the triangular mesh stage.

1.3.2 Experimental studies

The initial research into velocity profiles and flow distribution within human upper airways concentrated on experimental approaches. Schroter & Sudlow (1969) observed how secondary flows formed during inspiration and expiration in a single bifurcation glass tube model using dye tracing methods with either flat or parabolic inlet profiles. The of secondary flow profile depended on the direction of flow along the tube. Double vortices were formed during inspiration in the daughter tubes whereas quadruple vortices were created during expiration in the parent tube. Depending upon the curvature of the junction, flow separation with sluggish reversed flow could be observed in daughter tubes during inspiration. Axial velocity profiles were highly asymmetric. Peak velocities were biased to the inner walls. During expiration, velocity profiles developed an axial peak. Chang & El Masry (1982) measured detailed steady inspiratory velocity profiles in a scaled plastic tube model of the human tracheobronchial airways using anemometer probes, in which the model had 12 pre-drilled measurement stations and the dimensions of their model were consistent with the lung geometry reported by Horsfield et al. (1971). Menon et al. (1984) used the same method and model used by Chang and El Masry (1982) and derived detailed velocity profiles obtained during inspiration and expiration. The results of both studies showed a high degree of asymmetry in all branches, with peak velocities near the inner wall of the bifurcation. Overall, the velocity profiles were more sensitive to airway geometry than to flow rate. However, at low oscillatory frequencies (e.g. 0.25 Hz) of inlet flow, the velocity profiles attained at peak flow rate resembled the profiles seen under steady flow conditions at the corresponding Reynolds number. As the frequency increased (e.g. 4 Hz) the velocity profiles throughout most branches tended to flatten. Details for high frequency will not however be included here because high oscillatory frequency flow is beyond the scope of this research.

The research described demonstrates that velocity profiles measured in different experiments can be comparable when correct Reynolds number, oscillatory frequency and geometrical details are used. This observation will be a good indicator in verifying the present numerical study.

1.3.3 Numerical studies

In past numerical studies, a smooth-walled (and hence simplified) airway model was mostly employed with variation only in geometry and method of dichotomy. Regular dichotomy (symmetry) airway models have been extensively studied by Comer et al. (2000a & b) and Zhang et al. (2002a & b). Irregular dichotomous airway models have also been investigated by Zhang et al. (1997) and Balásházy & Hofmann (1993). All of these studies showed that the regional deposition efficiency (DE) can be expressed as a logistic function of the inlet Stokes (St) number for the studied bifurcation geometries.

In terms of validating CFD results, Zhang and Kleinstreuer (2002) and Comer et al. (2000a & b) have extensively compared their numerical results with experimental results for velocity profile from Zhao & Lieber (1994). They demonstrated consistent results under steady flow conditions within a Stokes number range from 0.01-0.12. They also explained how particle trajectory was affected by airflow vortices before and after bifurcation using the G3-G5 model. Balásházy and Hofmann (1993) and Zhang et al. (1997) compared the DE for different bifurcation angles using the G3-G6 symmetric model. Their results indicated that DE increased with larger bifurcation angles. Furthermore, Zhang et al. (1997) investigated DE at different Reynolds (Re) numbers and showed that DE initially increased with Reynolds number, but became almost independent of Re when St > 0.1. The studies described above used idealised models where the model geometry was hardly comparable to real human airway structures. From a realistic human airway geometry perspective, Vial et al. (2005) reconstructed the tracheobronchial airway from CT scan data and simulated airflow. Their results were examined in terms of the lobar flow distributions against Katz and Martonen (1996) and Corieri (1994)'s results and geometrical details against Weibel (1963)'s dichotomous model. Ertbruggen et al. (2005) used smooth-walled models based on the morphometrical data of Horsfield et al. (1971), in which Horsfield and his colleagues measured all structure down to generation 6 of a resin cast of a normal human bronchial tree. They compared their numerical results with experimental results with Kim et al. (1996) in terms of regional deposition efficiency and velocity profiles with Calay et al. (2002)'s measurements. Since the experiments against which they compared had either respiratory or

geometrical differences, some rounding errors were expected, but basic characteristics of flow and particle behavior were still reflected in their results.

Zhang et al. (2005) proved that the larynx effect causes turbulent fluctuations at medium and high inspiratory flow rates (30 and 60 L/min) because of the enhancement of flow instabilities immediately upstream of the flow dividers. Moreover, the effects of turbulent fluctuations on micro-particle deposition are relatively important in the human upper airways. These phenomena therefore are considered in the present study.

Asgharian and Price (2006) studied the influence of airflow distribution among bronchi on particle deposition and found that the airflow rate entering each major bronchus was similar for uniform and non-uniform lung expansions and concluded that the assumption of uniform air expansion and contraction was sufficient for the prediction of regional and total deposition of particles in the lung. However, they also indicated that accurate predictions of local and site-specific deposition of particles required more detailed models of lung ventilation including accounting for non-uniform lobar expansion because of the pressure variation in the pleural cavity. In the current research, the assumption of uniform air expansion and contraction was taken into account, but the latter issue was not considered in this study due to its complex modeling requirement.

1.4 Research Objectives

To broaden the current published knowledge of lung airway tree geometry modeling and its CFD simulation, the main objectives of this research are as follows.

a. To develop a new algorithm for the reconstruction of tubular organs.

b. To develop a new algorithm to generate a human lung airway tree and arterial vessel model.

c. To compare the difference of geometries and inhalation efforts between a bronchoconstricted and a recovered airway tree associated with acute asthma.

d. To compare airflow patterns between a bronchoconstricted and a recovered airway tree associated with acute asthma.

1.5 Thesis Structure

Chapter 1 provides an in-depth review of the background for this research. The rationale for conducting this research, the scope and outlines of the thesis are also explained in this chapter. Some previous studies which relate closely to this research are considered. This study is divided in two parts, lung air way geometry reconstruction and numerical air flow simulation. Lung air way geometry reconstruction consists of four stages: medical imaging acquisition, image segmentation, cloud points triangulation, and NURBS patch fitting. There are three main aspects of air flow simulation studies of human airways: experimental, theoretical and numerical approaches. The methods used in geometry reconstruction and air flow simulation, their differences, advantage and disadvantages of the methods are discussed and compared. This literature review provides the main frame of reference for this research.

Since the main focus of this research is the use of realistic airways in the study of air flows, the modeling of airway geometry becomes an important aspect. Therefore, in chapter 2 the general medical geometry modeling processing steps are reviewed and two CT/MRI image based geometry reconstruction approaches, the reverse engineering approach and STL-triangulated model converting approach, are proposed and tested in human lung airway geometry model generation.

In chapter 3, a new region-based method is presented to reconstruct the entire airway tree and carotid vessels from point clouds obtained from CT images. A novel layer-by-layer searching algorithm is developed to recognize the branches of the airway tree and arterial vessels from the entire point clouds. Instead of applying a uniform accuracy on all branches regardless of the number of available points, the surface patches on each branch are constructed adaptively based on the number of available elemental points, which leads to the elimination of distortions occurring at small bronchi and vessels.

Chapter 4 provides comparisons of airway geometry and inhalation effort in a bronchoconstricted and recovered airway tree associated with acute asthma. Reconstructed models of an airway tree taken a day after the initial onset of an acute exacerbation of asthma and following recovery from the same patient thirty days later were developed. A computational model was created for Computational Fluid Dynamics (CFD) analysis which provides details of the geometry and airflow properties including the pressure drop.

In Chapter 5, the airflow patterns, the pressure distribution, and the implications they have for targeted drug delivery are compared between bronchoconstricted and recovered airway trees associated with acute asthma. Local flow patterns are also discussed in depth in this chapter.

The final chapter summarizes the findings and presents conclusionsconcerning the significance of the outcomes.

Chapter 2
Medical Geometry Modeling

2.1 Introduction

Three dimensional (3-D) geometric modeling is a widely used technology in the aerospace, automotive and manufacturing industries. Examples of its application include the geometric design of the vehicle bodies and the complex shapes of turbine blades in aircraft engines. These models are mathematically expressed as 3-D solid geometries in CAD/CAM (Compute Aided Design/Computer Aided Manufacturing) systems. Recently the application of geometry modeling has been extended to scientific areas, including biomedical and bioengineering for constructing geometric models of human organs in order to facilitate diagnosis and treatment of patients (Sun 2004, Hollister 2000). For example; a geometric model of a tumour in a patient's body helps the doctor prepare before commencement of surgery; a knee model provides insights into the mechanisms of articular joint contact; a geometric model of the vessels helps simulate dynamic blood flow inside the human body. These are examples of benefits largely made possible by developments in imaging technologies and in reverse engineering techniques supported equally by hardware and software technology advancements. Efforts to model human body parts in a CAD based virtual environment are referred to as Bio-CAD modeling. Most Bio-CAD modeling is image based and involves the three basic steps of 1) Image acquisition 2) Image segmentation 3) Three dimensional geometry model reconstruction.

2.2 Medical Imaging

Medical imaging is a relatively new discipline, emerging only in the last century (consider for comparison that some aspects of medicine have been in use since ancient times, and the more modern forms of evidence-based practice began to emerge — along with the scientific

method — in the 1500's). At present, many medical image acquisition modalities are available, including CT(Computed Tomography), MRI(Magnetic Resonance Imaging), fluoroscopy, ultrasound, positron emission tomography (PET), projection radiography, photo-acoustic imaging, etc. Each modality has advantages and limitations and is used in different applications (Hollister et. al 2000). In the clinical context, medical imaging is typically equated with radiology; however, medical imaging encompasses the sub-disciplines of medical physics, biomedical engineering, and radiography (image acquisition).

Medical imaging began in 1895 with Roentgen's discovery of X-rays. Commonly referred to as 'plain film,' an X-ray image is created by passing electromagnetic radiation through the object of interest. Different tissues attenuate the rays to different degrees, creating a high contrast image. The attenuation depends on the thickness, density, and atomic number of the tissue. The resultant image is a grayscale 2-D projection of a 3-D object (which can result in difficulties in distinguishing between superimposed tissues with similar attributes).

Computed tomography (CT) is a combination of mechanical and computer engineering which was first developed in 1972 by Sir Godfrey Hounsfield. Figure 2.1 (Kaiser Santa Rosa.org) shows a CT scanner. During a typical CT procedure, the patient is placed on a table. The table then moves the patient through the gantry (a donut-shaped device), which houses an X-ray tube and detector array. For each image acquired, the X-ray tube rotates around the patient and the X-rays pass through the patient to the detector array, and thousands of X-ray measurements are acquired. The computer then processes this information and displays the corresponding images on a computer screen. This imaging technique avoids superimposition of organs or tissues which might occur during other types of X-ray tomographic studies. The CT examination creates images analogous to a single slice of bread from a whole loaf or a slice from an orange. Hence, the word 'slice' is often used to describe a view of patient anatomy. Each CT image consists of a large number of small picture elements (pixels). Each pixel is assigned a numerical value (gray scale), based on the degree to which the tissue corresponding to that pixel attenuates the X-Ray beam. Figure 2.2 (Kaiser Santa Rosa.org) illustrate the concept of slices.

Figure 2.1 Computed tomography (CT) scanner (http://Kaiser Santa Rosa.org)

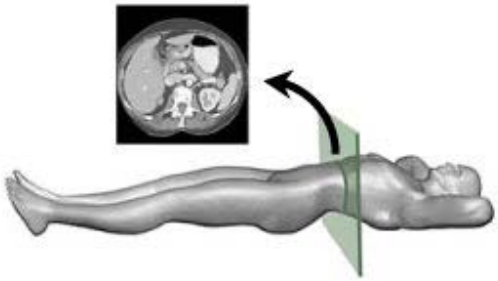


Figure 2.2 CT Slice Concepts (http://Kaiser Santa Rosa.org)

Magnetic resonance imaging (MRI) essentially provides a distribution map of hydrogen nuclei (found in water). A large magnet (typically 1.5 Tesla, which is approximately 50,000 times greater than the Earth's magnetic field) is used to align dipole magnetic moments of protons within the object of interest. An electro-magnetic radio frequency (RF) pulse is then applied to induce a 90◦ offset (spin echo). Following the pulse, the magnetic moments of the protons return to equilibrium or relaxation. There are two relaxation times of interest: longitudinal relaxation (T1) is the time for approximately 63% of protons to realign with the magnetic field along the longitudinal axis; transverse relaxation (T2) is the time for approximately 63% of protons to process out of phase. Different tissues exhibit different T1 and T2 relaxation times, which result in high-contrast images (Wright 1997). MR images are useful for creating images of soft tissue in the brain, spine, and joints. However, there are restrictions for patients with in vivo ferromagnetic objects (such as orthopaedic hardware, pacemakers, aneurysm clips, cochlear implants, etc).

2.3 Image Segmentation

2.3.1 Introduction

The data obtained from CT scans are a series of cross sectional images which contain different gray scale information with respect to different organs or tissues, and these images

are difficult to interpret by a lay person. Fortunately, the development of image segmentation techniques makes the extraction of useful information from these images more available. 2D segmentation is the extraction of the geometry of a CT scan data set (Mankovich et. al 1990). Each slice is processed independently leading to the detection of the inner and outer contours of the living tissue. The contours are stacked in 3D and used as reference to create a solid model usually through skinning operations. 3D segmentation is the partition of pixels in a 2-D image into connected regions that possess a meaningful correspondence to object surfaces in a 3-D scene represented by the image. 3D segmentation (Viceconti et. al 1999) of the CT data set is able to identify shapes within the CT data set, and extract a 'tiled surface' from them. A tiled surface is a discrete representation made of connected polygons (usually triangles). The most popular algorithm is the marching cube algorithm (Lorenson and Cline 1987). In its original formulation the marching cube method produces tiled surfaces with topological inconsistencies (such as missing triangles) and usually a large number of triangle elements. This method decomposes complex geometries into 'finite elements' and and the quality of approximations to the behaviour of the system depends on the number of these elements and the order of the approximation over each element. In the visualization process, each triangle is treated as a separated polygonal entity and the computational requirements increase exponentially with the number of triangles.

Many studies have been performed in the image segmentation area and several general-purpose algorithms and techniques have been developed for image segmentation. In general, image segmentation approaches can be classified into two broad classes: (i) structural and (ii) stochastic approaches. Structural approaches are data-driven processing which do not need any prior knowledge of the objects. Watershed, edge-based schemes and region-based schemes belong to structural approaches. Stochastic approaches are often referred to as model-based approaches where high level knowledge of objects is represented by statistical probability distributions, i.e., as the image models.

2.3.2 Locally constrained watershed algorithm

The local constrained watershed algorithm is a newly developed watershed-based algorithm which was proposed by Richard (Beare. 2006). The algorithm modified the distance function of the standard watershed algorithm. It imposes constraints on region boundaries by the introduction of a structuring element. This is different to the usual approach of adding a regularizing term to an energy function since the border constraints are implicit in the definition of a path. Also, by incorporation of the structuring element a constrained catchment basin is defined. This definition guarantees that the resulting catchment basins are disjointed, opened by the structuring elements, and that the watershed lines generally correspond to image features. The details are as follows:

Suppose an input image is denoted by $f : D \to IN$. And S denotes the structuring element. The notation S_x denotes the structuring element with origin x. Then the path of structuring elements will be constructed which is denoted by $\hat{\pi}$ and is called a $Minkowski\ path$ or $Covered\ path$. The origins of the structuring elements form a path $\pi = (p_0, ..., p_l)$. The Minikowski path $\hat{\pi}$ is:

$$\hat{\pi} = (S_{p0}, ..., S_{pl}) \tag{2.1}$$

This notation means that $\hat{\pi}$ is a sequence of translation of a structuring element with origins $(p_0, ..., p_l)$. Hence, the Minkowski distance between p and q is given by:

$$\Pi_f^{\pi}(p,q) = \min_{\pi \in [p \to q]} \Pi_f^{\pi} \quad . \tag{2.2}$$

Here, denoted by $[p \to q]$ is the collection of all paths of between p and q. The Minkowski cost between p and q is the minimum cost of all Minkowski paths between p and q.

The constrained watershed of f is therefore

$$Wshed(f) = D \setminus \bigcup_{i \in I} CCB_i \tag{2.3}$$

$$CCB_i = \left[\bigcup_{\delta \geq 0} V_i(\delta) \right] \oplus S \tag{2.4}$$

The V_i denotes the constrained partial catchment basin for regional minimum M_i. δ is a distance bigger than zero.

2.3.3 Applying the locally constrained watershed segmentation algorithm on the carotid artery

The segmentation method described previously was applied to five sets of human carotid artery CT scan images. The algorithm was implemented by using the Insight ToolKit (ITK. Org) library. The Insight Toolkit (ITK) is an open-source software system designed primarily for medical image segmentation and registration (Yoo et.al 2002, Ibanez et. al 2007). ITK is a set of cross-platform type-templated C++ class modules; supported operating systems include Windows, Unix, and MacOSX. The toolkit is organised in data-flow architecture: process objects (filters) consume data objects (images). ITK was originally released in 1999 as a joint project coordinated by the US National Library of Medicine (NLM) of the National Institutes of Health (NIH). At the time of writing the toolkit is still supported, and has active user and developer mailing lists, Version 3.2.0 released in March 2007 was used for this evaluation. ITK supports the reading and writing of n-dimensional images (2-D, 3-D, etc), various noise suppression algorithms (including Gaussian, median, gradient anisotropic diffusion, curvature flow, bilateral filtering, and others), a wide number of intensity transformations and mappings, various image registration algorithms, and a range of image segmentation and classification methods. As ITK does not support any volume rendering or illustration methods, the Visualization ToolKit (VTK. Org) was used for the model visualisation.

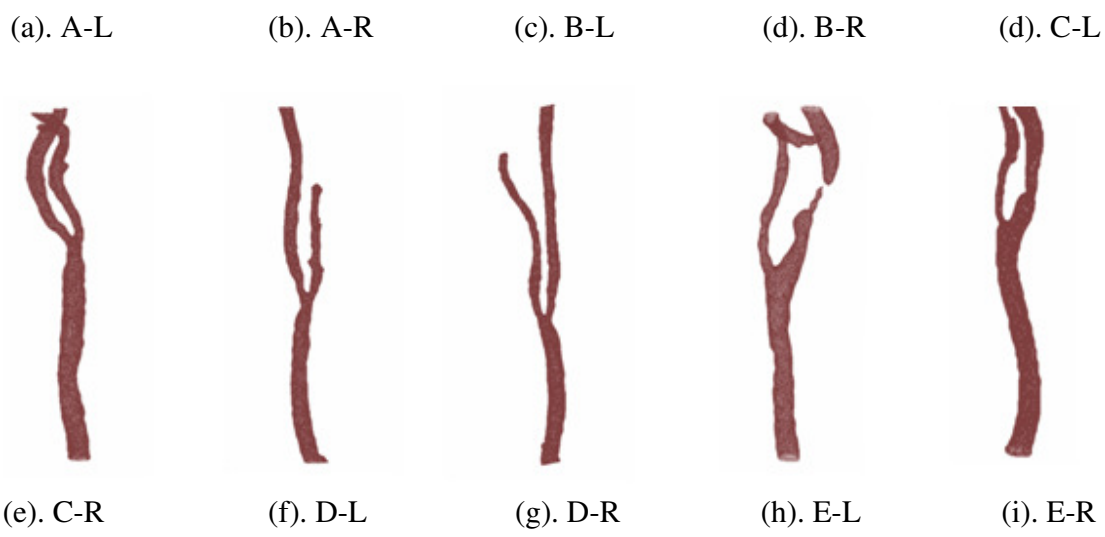


Figure 2.3 Carotid artery segmentation results (displayed with triangle mesh)

The results are shown in Figure 2.3. “L” and “R” indicate the left and right carotid artery. In most cases the carotid artery could be successfully segmented by using the locally constrained watershed algorithm. However errors occurred in two cases due to noise. Figure 2.3(h) shows an over segmented result in which it can be seen that the right branch of carotid artery is broken to two sections. It can be seen from Figure 2.3. (i) that the left branch of the carotid artery should be divided into two branches at its end area, but a lump like shape is generated because the two branches are combined.

2.3.4 Pulmonary workstation 2

The geometry of the carotid artery is much simpler than that of a lung airway. The carotid artery has no more than two generation bifurcations. In most cases the carotid artery has only one bifurcation. Based on above segmentation results, it is reasonable to draw a conclusion that the locally constrained watershed algorithm is not suitable to segment the lung airway which has more bifurcations;with branches that are more fine and close to each other. Fortunately some professional commercial segmentation software packages are available on the market. Pulmonary Workstation 2 (VIDA Diagnostic Inc.) is one of those commercial software packages.

Pulmonary Workstation 2 is a leader in commercial software packages for pulmonary image analysis and therapy guidance. It offers fully automated and reliable identification of airway trees from CT volumes. Consequently the need for human operators to manually optimize segmentation parameters has been eliminated. This software can work without any changes on different types of scans, for example, low dose and regular dose, diseased subjects and normal subjects, and the run time does not exceed a few minutes per volume.

Pulmonary Workstation 2 implements the segmentation by using a new algorithm proposed by Juerg which is based on fuzzy connectivity (Tschirren et. al, 2004). During the execution of a traditional fuzzy connectivity algorithm, two regions – foreground and background – are competing against each other. It makes this method able to overcome lack of image contrast between the airway and the airway walls and the effects of noise. The disadvantage is its relatively high computational complexity. The new fuzzy connectivity algorithm reduces computing time by splitting the segmentation space into a number of smaller subspaces and by keeping the search space as tight as possible around the airway segments. The desire to keep the segmentation within a small region, together with the need to detect possible leaks at their root, led to the idea of using a relatively small adaptive region of interest (ROI) which follows the airway tree branches as they are segmented. The ROI has a cylindrical shape and adapts its geometric dimensions, its orientation, and position to the predicted size, orientation, and position of the airway branch to be segmented. The notion of ROI brings two main advantages. One is that it leads to shorter segmentation time as the

segmentation process is kept close to the airway segments. Another is that segmentation leaks can be detected and dealt with early. This concept is illustrated by Figure 2.4 (Tschirren et. al 2004). In this figure Adaptive cylindrical regions of interest (light gray) follow airway tree branches as the segmentation proceeds. Segmentation is performed in a small region only, helping in the early detection of leaks as segmentation proceeds. Using a cylindrically shaped ROI has the advantage that the ROI adapts better to the target shape, which is close to cylindrical. This means fewer "useless" background voxels need be analysed and the computing time can be reduced.

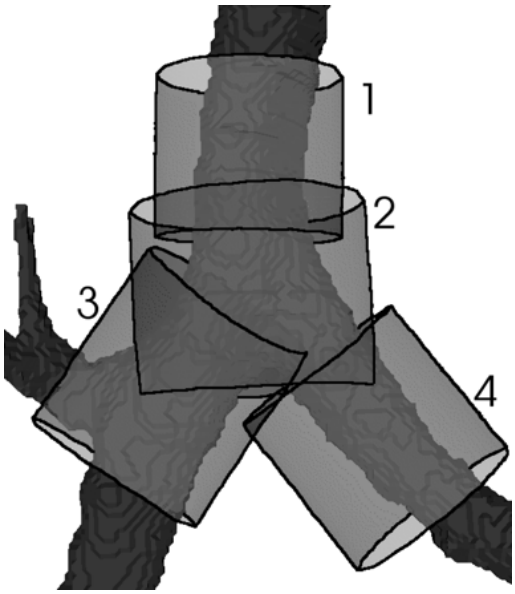


Figure 2.4 Airway tree segmentation (Tschirren et. al, 2004).

2.3.5 Applying pulmonary workstation 2 to lung airway segmentation

In this study the lung airway was segmented by applying Pulmonary Workstation 2. Two sets of CT scan images were used in the study. The two scans were performed on a 66 year old non-smoking Asian male (height 171 cm and weight 58 kg) using a helical 64 slice multidetector row CT scanner (General Electric) the day after hospital admission with an

acute exacerbation of asthma. At the time, the patient's lung function by spirometry (Spirocard, QRS Diagnostic, Plymouth, Minnesota, USA) showed severe airflow obstruction with a forced expiratory volume in 1 second (FEV1) of 1.02 L (41% predicted) Data was acquired with 1-mm collimation, a 40-cm field of view (FOV), 120 kV peak and 200 mA. At baseline, 2 cm axial length of lung caudad to the inferior pulmonary ligament was scanned during a single full inhalation total lung capacity with breath-hold, which yielded 254 contiguous images (slices) of 1-mm thickness with voxel size of 0.625 x 0.625 x 1 mm. An identical protocol was used to acquire images following recovery 30 days later, when his FEV1 was measured at 2.27 L (91% predicted).

These two sets of CT scan images were loaded into Pulmonary Workstation 2 respectively. Pulmonary Workstation 2 provides a totally automatic tool to segment the lung airway. However it does not provide the tools to write segmentation results to files on its tool menu. This software can only display the segmentation results in its analysing window. The triangle mesh result writing tool is in a separate file named "analyze2stl.exe". This tool can be used to create an ASCII STL file from an Analyze file containing the airway segmentation result. This Analyze file is located in a directory called "+data" at the place where the Pulmonary Workstation 2 is installed. The analyze file "segmentedData.hdr" and "segmentedData.img.gz" can also be found in the directory. The command "ananlyze2stl.exe" has a few optional parameters:

(1) --f n Reduce nr. of triangles to 1/n. Default: n = 1;

(2) --s0 Do not smooth surface. (Default);

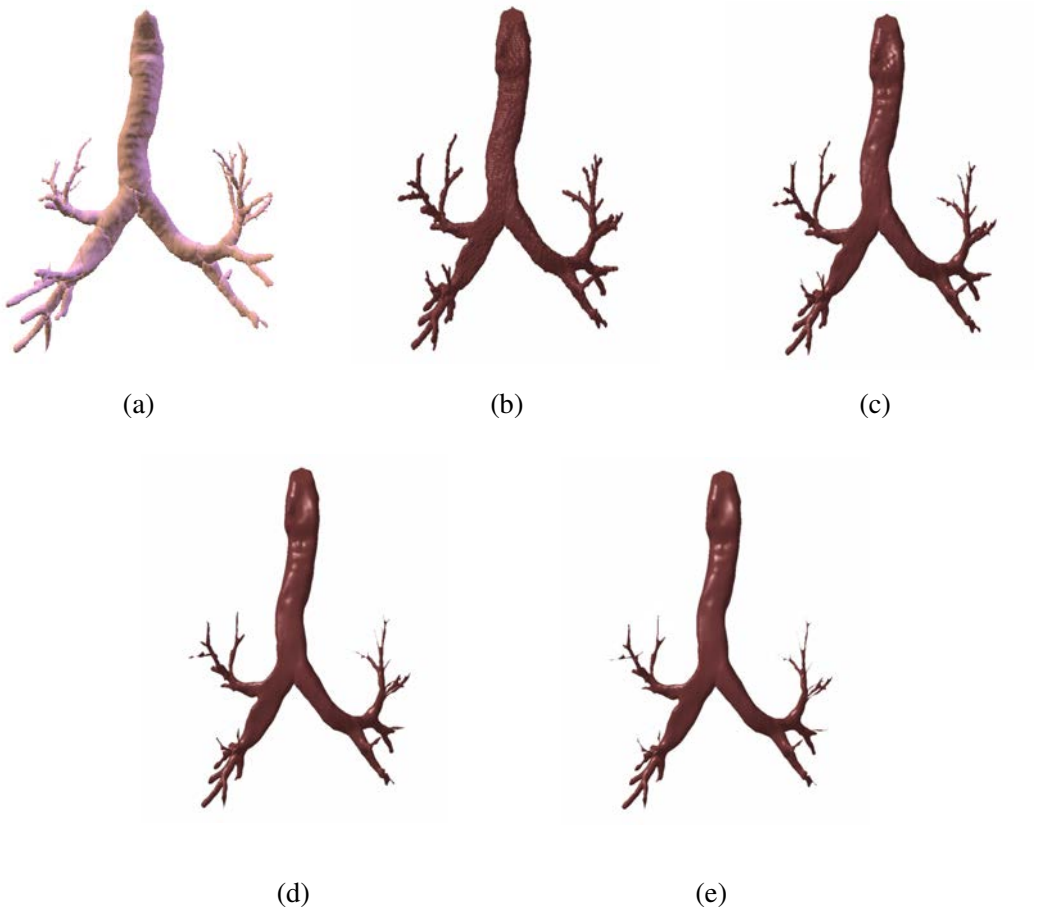
(3) --s1 Smooth surface – lightly;

(4) --s2 Smooth surface -- medium;

(5) --s3 Smooth surface -- heavily.

Based on our experiments the best results can be obtained by using "s1" smooth parameter. The segmentation results for asthma affected and recovered lung airways are shown on Figure 2.5 and Figure 2.6. From this figure the best segmentation results were obtained by

using smooth parameter “s1”. If the model is too rough it will cause problems in the down-stream post-processing (surface fitting). On the other hand, if it is too smooth some useful features will be omitted. Some fine tubes at the 4th generation or over in the model which was obtained by using “s2” or “s3” were omitted.



(a) (b) (c)

(d) (e)

Figure 2.5 Asthma affected lung airway model generated by Pulmonary Workstation. (a) Pulmonary workstation displaying model (b) Model smoothed by “s0” (c) Model smoothed by “s1” (d) Model smoothed by “s2” (e) Model smoothed by “s3”

(a) (b) (c)

(d) (e)

Figure 2.6 Recovered lung airway model generated by Pulmonary Workstation. (a) Pulmonary workstation displaying model (b) Model smoothed by "s0" (c) Model smoothed by "s1" (d) Model smoothed by "s2" (e) Model smoothed by "s3"

2.4 Three Dimensional Geometry Reconstruction

2.4.1 Introduction

Although the results of segmentation from CT/MRI can be used as an accurate 3D tissue description, the voxel-based anatomical imaging representation (triangle mesh) cannot be

effectively used in many biomechanical engineering studies. For example, 3D surface extraction requires either a large amount of computational power or extreme sophistication in data organization and handling; and 3D volumetric modeling on the other hand, while producing a realistic 3D anatomical appearance, does not contain geometric topological relations. Although they are capable of describing the anatomical morphology and are applicable to rapid prototyping through a converted STL format, neither method is capable of performing anatomical structural design, modelling-based anatomical tissue biomechanical analysis and simulation. In general, activities in anatomical modelling design, analysis or simulation need to be carried out in a vector-based modelling environment, for example using Computer-Aided Design systems and CAD-based modelling, which is usually represented as 'boundary representation' (B-REP) and mathematically described as Non-Uniform Rational B-Spline (NURBS) functions. Unfortunately, the direct conversion of the medical imaging data into its NURBS model is not a simple task. In the last few years some commercial programs, for example, SurgiCAD by Integraph ISS, USA, Med-Link, by Dynamic Computer Resources, USA, and Mimic and MedCAD, by Materialise, Belgium, were developed and used to construct geometry models from medical images. However, none of these programs has been efficiently and widely adopted by the biomedical and tissue engineering community due to the inherent complexity of the tissue anatomical structures. Effective methods for the conversion of CT data into geometry models still need to be developed.

Generally, there are three ways for the generation of a geometry model from medical imaging data: (1) directly convert imaging data to a geometry model, (2) reverse engineering approach, (3) STL-triangulated model converting approach. The outline of the process paths for generating a solid geometry model is shown in Figure 2.7.

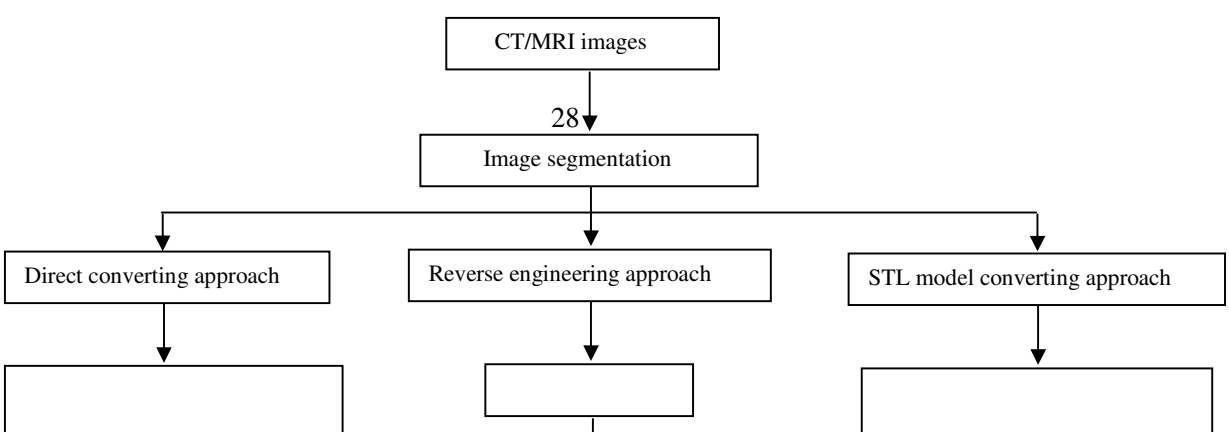


Figure 2.7 Processes for the generation of geometry models

As commercial software packages with the capacity to convert imaging data to geometry models were not available to the present study, only the last two process paths were tested in lung airway geometry model generation.

2.4.2 Applying the reverse engineering approach to lung airway geometry model generation

The reverse engineering approach uses the cloud points which are created from the image segmentation. The cloud points are first triangulated in order to find the topological information for the object. Then the triangulated model is further refined and enhanced to

reduce the file sizes and eliminate unwanted features. After being refined, the triangulated model are fed to reverse engineering software (for example, Geomagic Studios by Raindrop Inc). The freeform surfaces of NURBS patches are used to fit across the outer shape of the model by using reverse engineering software packages.

Three algorithms have been used to triangulate the cloud points. Each of these three algorithms was tested in by using the Visualization ToolKit (VTK) library. The Visualization ToolKit is an open source, freely available software system for 3D computer graphics, image processing, and visualization used by thousands of researchers and developers around the world (Schroeder et. al, 1996, 2000). It consists of a large number of C++ class modules supporting a range of visualisation algorithms for scalar, vector, tensor, and volumetric data. VTK was originally created in 1993 by three researchers from GE Corporate R&D (Will Schroeder, Ken Martin, and Bill Lorensen). Version 5.0.3 was used in this evaluation. It supports reading and writing of images, some basic data preparation and segmentation algorithms, and volume rendering. The VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modelling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, and contouring. In addition, dozens of imaging algorithms have been directly integrated to allow the user to mix 2D imaging / 3D graphics algorithms and data. The design and implementation of the library has been strongly influenced by object-oriented principles.

The three algorithms used to triangulate the cloud points in this study were: (1) Zero-set method (Hoppe et. al, 1992), (2) Power crust algorithm (Amenta et. al, 2001), (3) α-shape algorithm (Edelsbrunner et. al, 1994). The cloud points used as source data were the asthma affected lung airway segmentation results obtained from Pulmonary Workstation.

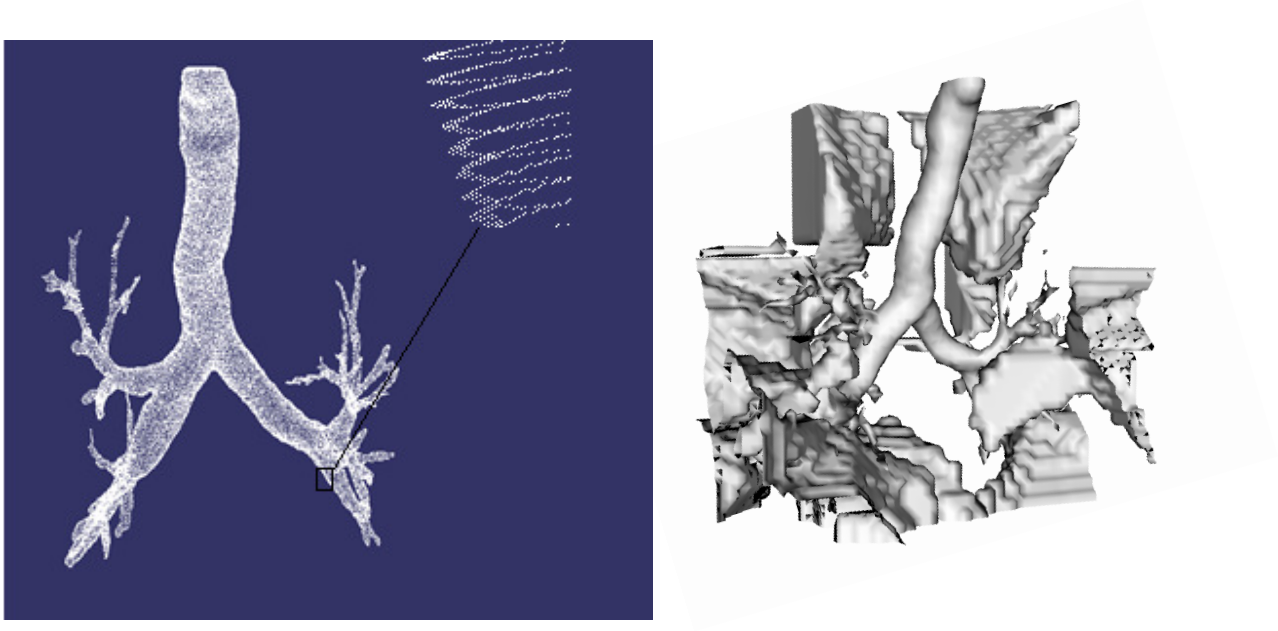
Hoppe et. al (1992) reconstructed the surface by using the zero level set of a distance function defined by the input point samples. It is assumed that a set of data points is on or near an unknown surface. The signed distance from an arbitrary point to a known surface is then defined. Knowing the signed distance function is equivalent to knowing the surface because an implicit representation for the surface is given by the zero set. In other words,

although the surface is unknown, the surface can be estimated by firstly estimating the signed distance from the data points then extracting an approximation of its zero set. The key ingredient for estimating the signed distance functions is to associate an oriented plane with each of the data points. These oriented tangent planes are then used to define the signed distance function to the surface. The marching cubes algorithm is used to extract the isosurface. This algorithm has several advantages. It has the capability of dealing with large numbers of points and can deal with the cloud points including noisy data.

The Crust algorithm (Amenta et. al, 1998) is the first algorithm which has been proposed with theoretical guarantees in reconstruction. It exploits the structures of the Voronoi diagram of the input point set to reconstruct the surface. Amenta et. al introduced the power crust algorithm which improved Crust both in theory and practice. The concepts of power crust and power shape were introduced in the power crust algorithm. In this algorithm, it outputs the power diagram faces separating the cells of inside and outside poles as power crusts and outputs the regular triangulation faces connecting inside poles as power shapes. The power crust algorithm works well on dense data sets and is capable of keeping the sharp corner features of the mechanical parts. However this algorithm introduces many extra points in the output and generates more faces than other algorithms.

The α-shape algorithm is a Delaunay based algorithm. A very early paper on the problem of approximating a surface in three dimensions from its sample points was presented by Boissonnat who proposed a sculpting of the Delaunay triangulation for reconstruction (Boissonnat, 1984). A more refined sculpting strategy was designed by Edelsbrunner and Mucke in their α-shape algorithm (Edelsbrunner et. al, 1994). In the α-shape algorithm, the α-shape is defined as a generalization of the convex hull of a point set. The notion of α-ball and α-hull are also introduced in this algorithm. An α-ball is defined as an open ball with radius α. For $\alpha = 0$, it is a point, and for $\alpha = \infty$, it is an open half space. The α-hull is defined as the complement of the union of all empty α-balls. This algorithm generates the approximate surface of a set of point by three steps: (1) construct the Delaunay triangulations by using flips; (2) compute the α-intervals for all simplexes in a Delaunay triangulation; (3) sort the endpoints of these intervals.

The point set of the lung airway obtained from Pulmonary Workstation 2 was input into the above three surface approximation algorithms. The point set contained 46810 points. Experiments were carried out using a personal computer with an Intel Core 2 CPU, 1.86 GHz, and 1 GB RAM. Only the Hoppe's algorithm could complete the computation and its triangulation result is shown on Figure 2.8. It can be seen from the figure that the trachea and first generation of the branch can be reconstructed correctly. However, the deeper generations of branches failed to generate because of the complexity of the geometry and the inherently noisy nature of the sample points.



(a) (b)

Figure 2.8 Triangular surface generated from cloud points. (a) Point clouds of asthma affected lung airway (b) Triangular surface generated by Hoppe's algorithm

As the sample points are obtained from image segmentation, it can not be guaranteed that all of the points are on the same surface of the object. It is possible that some points are on the inner surface of the object and other points are on the outer surface of the object. From the zoomed-in area in Figure 2.8(a), it can be found that each slice has a thickness of points around the tube, but the surface reconstruction algorithms tried to generate a surface going

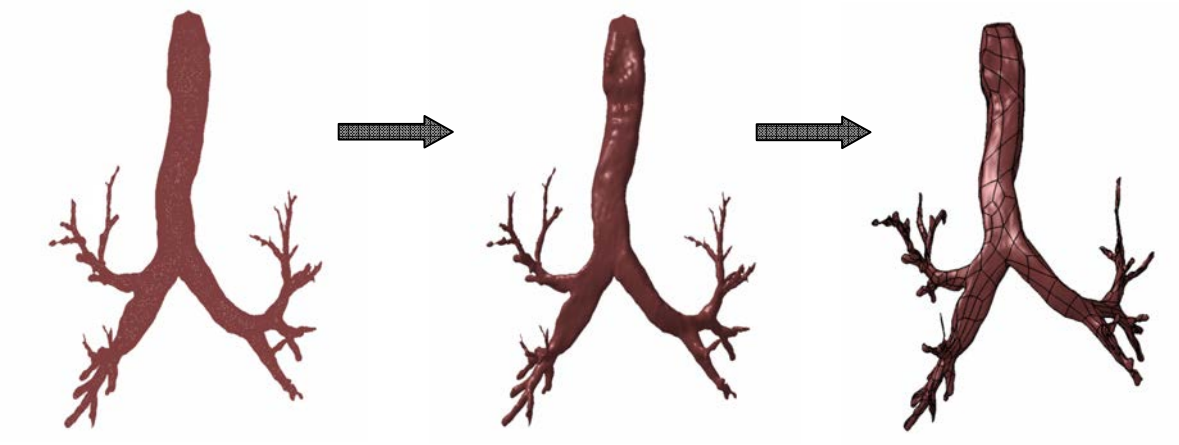
through all of these points. So this resulted in the failure of the surface reconstruction processing. Both the power crust algorithm and the α-shape algorithm failed to complete their calculation for the same reasons.

Based on the above experiments, it is apparent that the reverse engineering approach does not suit medical geometry model reconstruction. At the triangulated surface generation stage, it is difficult to generate a surface from the points which are obtained from image segmentation bacuase of the points being on t both inner and outer surfaces of the object. It is easier to extract the inner or outer surface of an object of interest from the original volume image at the image segmentation stage, because much more information may be used to generate its surface, for example, the relationships between slices. The reverse engineering approach to reconstructing surfaces can only apply the arbitrary point information.

2.4.3 Applying the STL-triangulated model converting approach on lung airway geometry model generation

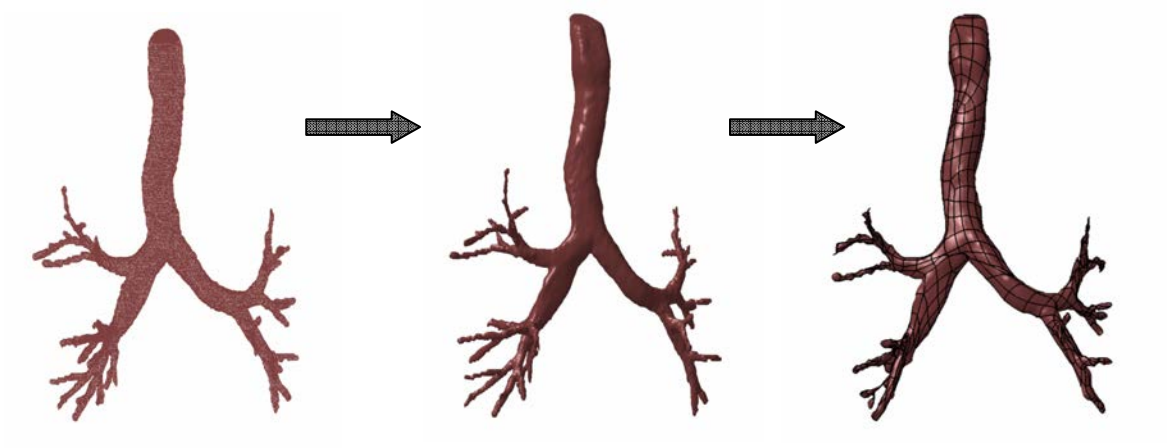
The 3D image segmentation can also export an STL file which can then be imported into reverse engineering software for surface refinement and NURBS surface generation. The difference between this approach and the reverse engineering approach is that this approach uses the STL-triangulated surface rather than the point clouds data as modeling input.

In this study the reverse engineering software, Geomagic Studio 7.0 (Raindrop Inc), was used for the fitting of NURBS surfaces. Geomagic Studio 7.0 is a powerful tool for reverse engineering of surfaces. It has a rich functionality, including point cloud processing (noise reduction, outlier removal, thinning), alignment of different data sets, triangulated surfaces (triangulation from points, smoothing, hole filling, triangle reduction), and NURBS modelling. The lung airway STL-triangulated models obtained from Pulmonary Workstation 2 were used as source data. Both the asthma affected and recovered STL-triangulated models were tested.



(a) (b) (c)

Figure 2.9 A NURBS surface model of asthma affected lung airway tree generated from triangle mesh. (a) Triangle mesh (b) Faceted model (c) NURBS surface model



(a) (b) (c)

Figure 2.10 A NURBS surface model of recovered Lung airway tree generated from triangle mesh. (a) Triangle mesh (b) Faceted model (c) NURBS surface model

The refinement of triangular meshes will increase the numbers of triangles in the polygon and result in a smoother surface. Although a mesh refinement function is provided by Geomagic Studio 7.0, the triangular meshes obtained from Pulmonary Workstation 2 are

smooth enough, as the triangular meshes have been refined by Pulmonary Workstation 2, so no further refinement is necessary in Geomagic Studio for the generation of rectangle CFD meshes. The asthma affected and recovered lung airway triangular meshes were respectively loaded into Geomagic Studio 7.0 to generate its NURBS surface. NURBS surface are the standard surface representations used in CAD, CAM, CAE (Computer Aided Engineering) programs. Gometric Studio 7.0 provides two tools: manual and automatic tools, to create a watertight NURBS surface for exporting to downstream applications. The exported NURBS surface can be saved in IGES (Initial Graphic Exchange Specification) or STEP (Standard for the Exchange of Product Model Data) format. The manual tools include detecting contours, subdividing/extending contours, specifying sharp lines, etc. The manual tools are suitable for the generation of surfaces with some known mechanical features. As lung airway geometry does not have any mechanical features such as sharp corners, fillets, and chamfers, the automatic tool was applied to generating the lung airway NURBS surface. 1007 NURBS patches were generated in the asthma affected lung airway NURBS surface model; 1084 NUREBS patches were generated in the recovered lung airway NURBS surface model. Both the asthma affected and recovered lung airway NURBS surface model were saved as IGES files. The asthma affected and recovered lung airway NURBS surface model generations are shown in Figure 2.9 and Figure 2.10. In this figure, the faceted models are from the same STL file as the triangle mesh model. The difference between these two models is that different visualization methods are used. The faceted model uses planer triangular faces to fill every hole of the triangle mesh model in order to make the model more realistic. From this figure, it is found that the faceted models are able to show more surface features than triangle mesh models because of surface reflection of light. Also it can be found that the NURBS surface models are smoother than the faceted models. The NURBS surface fitting process can be executed well from trachea (generation 0) to generation 3 and the generated NURBS surface is a true watertight surface; however over 3 generations the generated surface contained some errors such as gaps or overlaps between patches and some fine tubes were omitted in the NURBS model. The gaps or overlaps result obtained in the NURBS model were not suitable for the following CFD simulation. Based on the author's

experiments a large amount of manual post processing is still needed. More details about post processing for CFD use are explained in Chapter 5.

2.5 Summary

In this chapter, two CT/MRI image based geometry reconstruction approaches, the reverse engineering approach and the STL-triangulated model converting approach, have been proposed and have been tested in human lung airway geometry model generation. Three triangulation algorithms have been investigated with the reverse engineering approach. The results indicate that it is difficult to triangulate the points obtained from medical image segmentation when the object geometry shape is complex. Generally, the selection of the geometry reconstruction approach depends on the particular application it is intended for. The reverse engineering approach would be selected when the generated surface model or models have less overall complexity. The STL-triangulated model converting approach is preferred when a rapid prototype of the model is needed for surgical planning or display.

Chapter 3
Region-based Geometric Modeling of Human Airways and Arterial Vessels

3.1 Introduction

The rapid improvement in the speed of computers and available memory size over the last decade has led to the emergence of computational fluid dynamics (CFD) as an alternative, cost-effective means of simulating real biomedical flows as in human respiratory and vascular systems. An anatomically-precise geometric model of human airways or vessels is the basic prerequisite to CFD analysis of air and blood flow in human bodies (Tgavalekos et. al 2005, Ertbruggen et. al 2004 and Burrowes et. al 2005). This analysis is critical in the calculation of respiratory and hemodynamic parameters such as breath patterns and wall shear stress which are now widely used in the diagnosis and management of respiratory and arterial diseases like asthma stroke (Hassan et. al 2004 and Quatember et. al 2004).

Three dimensional (3D) geometry modelling is a well established technology used in the aerospace, automotive and manufacturing industries (Choi 2003). With the advance of computational technology, its application has been extended to biomedical engineering in modelling various human organs. Much research on the reconstruction of human organs has been performed based on data obtained from CT or MR images (Koikkalainen et. al 2004 and Tawhai et. al 2004), and some commercial software programs are available in the global market for research and clinical applications (VIDA Diagnostic Inc and Materialise Inc). For example, Volkau et al. (Volkau et. al 2005) have proposed an anatomy-based approach for the efficient construction of a 3D human normal cerebral arterial model from segmented and skeletonized angiographic data. Palágyi et al. (Palágyi et. al 2006) developed an automated method for skeletionization, branch-point identification and quantitative analysis of tubular tree structures. Long et al. (Long et. al 2003) presented a reproducibility study of 3D

geometrical reconstruction of the human carotid bifurcation from in vivo MR images. By using a robust mathematical morphology operator – the selective marking and depth constrained connection cost, Fetita et al. (Fetita et. al 2004) developed an energy-based 3D reconstruction algorithm to construct the bronchial tree from multi-slice CT acquisitions up to the sixth – seventh order subdivisions. By ensuring harmonic mean curvature vector distribution on the surface, a new model construction and analysis method was developed by Antiga et al. (Antiga et. al 2004) for patient-specific reconstruction and meshing of blood vessels.

Anatomically, the human airway is a tree-like structure (Tschirren et. al 2005). The trachea and bronchi are tubular in shape. The diameter of the trunk (the first generation of the tree), the trachea, is approximately 20 mm for an adult, whereas that of the bronchus in the 4th generation is approximately 3 mm only, which is 1/6 of the dimension of the trachea in Generation 1. In chapter2 some experiments are described concerning reconstruction of the airway tree geometry model by using the published algorithms and the commercial packages currently available in the market. However, many flaws as detailed in Section 2 emerge in the generated geometries, which make it impossible to utilize the model directly in downstream CFD studies. Extensive work must be done to convert the geometries into workable numerical models. This hinders the downstream research on air and blood flows in which large amount of tests are needed to be run on different models. The availability of an accurate geometric model becomes a limiting factor for clinical application of CFD results. Therefore it is important to develop a new algorithm to reconstruct airway tree geometry models which can be directly used in downstream CFD studies.

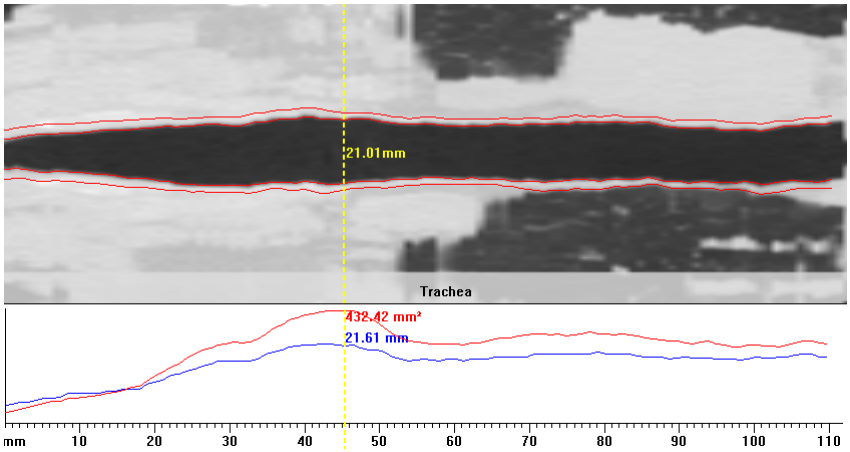
This study presents a new region-based algorithm that can be used to reconstruct geometric models of human airways and carotid vessels from point clouds obtained from CT and MR images. With this method, the generations of the airway tree or the branches of the vessel are firstly recognized from the entire point clouds with a new searching algorithm. The points on the same branch are grouped together. Because the diameters of bronchi of different generations may be significantly different, for example, the trachea may be 6 times thicker than the bronchus on the 4th generation in the airway tree, different numbers of points

are used to construct the surface patches, that is different accuracies or tolerances are applied in the approximation of surface patches on different branches. Meanwhile, the bronchi in the same region or on the same generation may have similar diameters although their physical locations in the lung are different and the same accuracy may be used in the approximation process. With this strategy, the number of elemental points used in the construction of different branches is adaptive to the locations of the branches, and the overall quality of the surface patches can be improved.

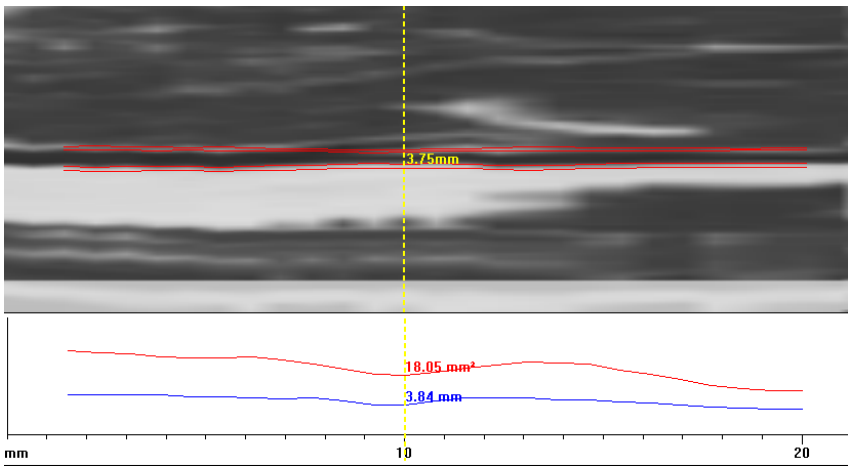
3.2 Geometric Characteristics of Airway and Blood Vessels

3.2.1 Geometric characteristics of airway and blood vessels

Human airways and blood vessels can be classified as related human organs in terms of their geometric profiles: both have the similar tube-like shape with air or blood flowing inside. Figure 3.1 illustrates the dimensions of the trachea and a bronchus of the 4th generation. The diameter of the trachea is approximately 21.6 mm and the bronchus at the 4th generation has a diameter of approximately 3.8 mm, which is 1/6 of that of the trachea. The diameter of a bronchus at higher generation is even smaller. Therefore, it may be concluded that there is a significant difference in diameters of the airway branches of different generations. This special characteristic affects the accuracy of models generated with the ordinary methods commonly used in reverse engineering (RE).



(a)



(b)

Figure 3.1 Dimensions of airways (a) Diameter of the trachea (b) Diameter of a bronchus of the 4th generation

3.2.2 Various flaws in the model

The number of elemental points used in the approximation process is a key factor controlling the number of surface patches used in describing a model. The general procedure to reconstruct such a model in RE is initially to set up the tolerance for the first

approximation. The model is then approximated by applying this overall tolerance to the entire model.

Because of the special topologies of human airways and vessels, this principle does not apply in this application. Various flaws such as overlap and gap (Figure 3.2) emerge in the model generated by using these packages. Severe errors as shown in Figure 3.2 occur at locations where the dimensions of the structure are small.

Figure 3.2(c) illustrates the airway tree generated from CT images of a male adult by using two commercial packages. The model has 7 generations; the diameter of the trachea is more than 10 times larger than that of the bronchioles at the 6th generation. In the enlarged view (Figure 3.2(d)) it can be seen clearly that the bronchus which is tubular becomes a flat surface patch. Similar errors also occur in the reconstruction of arterial vessel. These types of errors occur in many cases and much work is needed before a model is pre-processed for mesh generation.

3.2.3 Analysis

One possible reason for the errors described is the redundant or insufficient number of surface patches used in the approximation process for the smaller structures.

In the point clouds obtained from medical images, fewer points could be obtained at the location of small structures due to the noise and limited resolution of the images. If a uniform number of points is used in the approximation process regardless of the dimension of the patch, errors will occur at locations where less points are available. On the other hand, if the size of a surface patch is reduced to satisfy the requirement of small bronchus, a large number of small surface patches will be generated redundantly in the trachea or larger bronchus in the airway tree, which makes it not practical in clinical practice because of the large files size resulting from the large amount of surface patches.

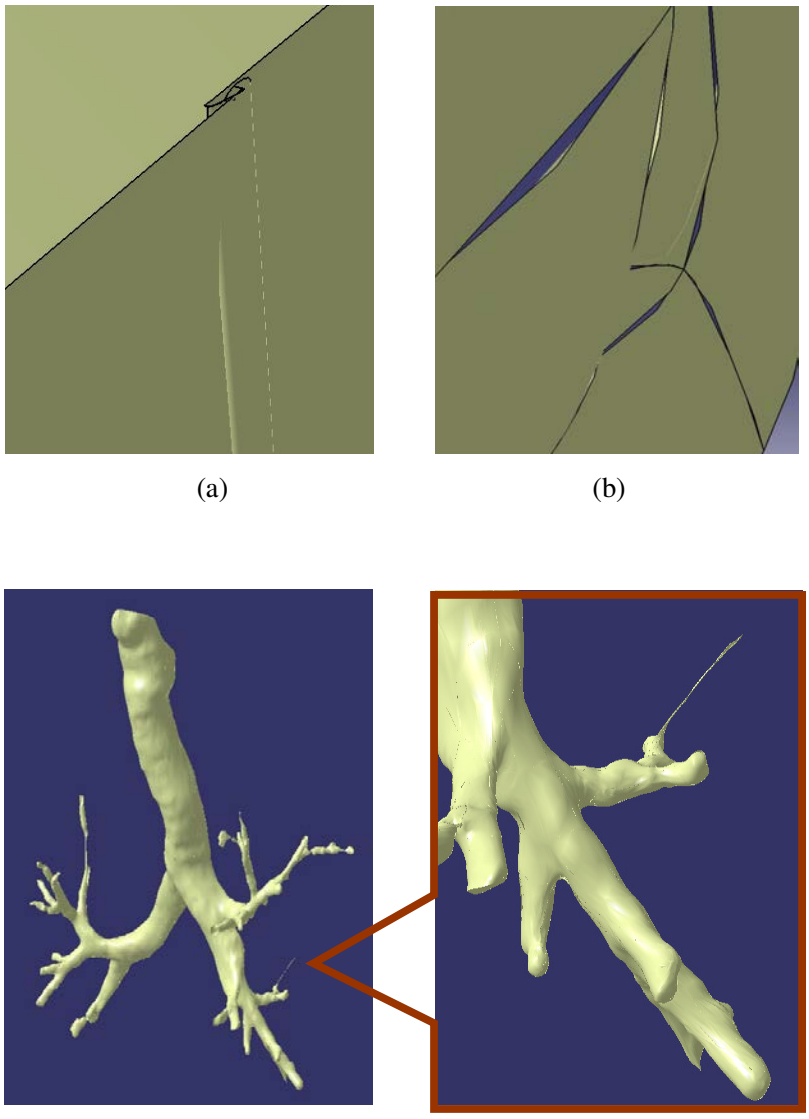


Figure 3.2 Geometric model of airways generated by a commercial package (a) Overlap on surfaces (b) Gaps between surface patches. (c) Solid model of airways (d) Enlarged view of bronchus

To solve this problem, an “adaptive tolerance” modelling strategy was developed. Under this strategy, a larger tolerance is used in approximating the main branches of the airway, which is the first or second generation of the airways tree. At the fifth or sixth generation of

the airway where very small structures have small diameters, a smaller tolerance is applied, which resulting in more surface patches at the end of each branch. With this strategy, the numbers of points used to construct the surface patches at different locations of the airway tree are different. Although the tolerance is higher at the smaller bronchi, the overall number of surface patches does not increase significantly and correct geometries can be generated.

To implement this adaptive algorithm, the location of the points should firstly be identified. That is, the points that consist of each branch should be recognized from the entire point cloud.

3.3 Surface reconstruction

The new algorithm consists of two steps: to recognize points on each branch by using the new search algorithm, and to construct the surface patch for each identified branch. The method developed by Eck and Hoppe is used in the second step (Eck et. al 1996).

3.3.1 Identification of branches

With current image processing algorithms, it is not difficult to segment the medical image and output the results as point clouds in STL (Stereolithography) format. Points and triangles are the basic elements used to describe geometric features in an STL file, and no overlap between triangles is allowed in this format.

To identify the branches, two essential concepts are defined as follows:

Definition 1: Associate Triangle

The associated triangle of Point P is a triangle with P being one of the vertex.

Definition 2: Closed Circle

A “closed circle” is piece of curve whose start and end points are vertices of the same triangle.

To find a bifurcation point or identify the points on each branch, the identification or reorganization of point clouds is done through following six phases.

1. Cut open the trachea and small bronchi

2. Divide the point cloud into sub-regions

   According to Z coordinates, an entire point cloud is divided into 12 sub-regions as shown in Figure 3.3. The purpose of this division is to reduce the number of points involved in the searching process so that the overall calculation time can be shortened.

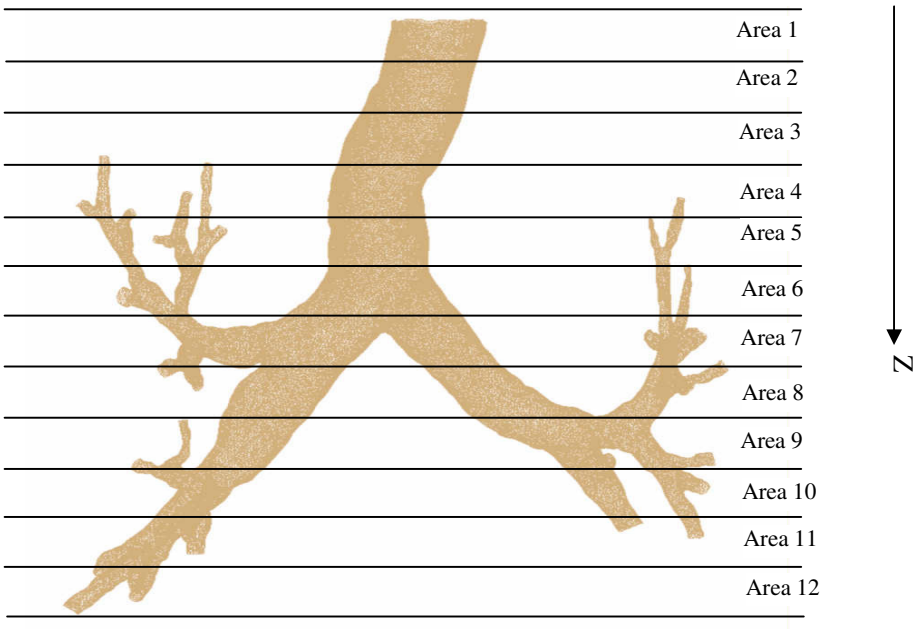


Figure 3.3 Point clouds are divided into sub-regions

3. Search for points on the top boundary of the trachea

   (1) Since the trachea starts from the upper airway and the bronchi are at lower locations in the lung, it is determined that the point in Area 1 with the minimum Z coordinate is a point on the boundary of the trachea. This point will later be used as the seed point to find the remaining points representing the trachea. If more than one point is found

having the same minimum Z coordinate, the seed point can be randomly selected from these candidate points.

Assume the coordinates of the seed point are P_1 (x_1, y_1, z_1); P_{1L} (x_{1L}, y_{1L}, z_{1L}) and P_{1R} (x_{1R}, y_{1R}, z_{1R}) are the two neighbouring points of P_1.

$\therefore z_1 \leq z_{1R}$ and $z_1 \leq z_{1L}$, where z_1 = minimum { $z_i \mid i \in$ index of points in the clouds }

(2) Identify all triangles associated with P1.

The search for these triangles is limited only to Area 1 as Point P1 is located in Area 1. Thus, the amount of calculation can be reduced significantly compared to the searching processes that are carried out on the entire point clouds.

(3) Among the triangles associated with P1, identify the isolated vertices which are not shared by any adjacent triangles. The two isolated points found are points on the top boundary.

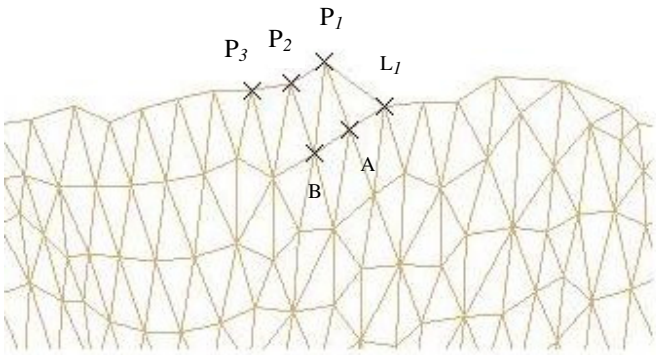


Figure 3.4 Searching for boundary points

It is clear that no isolated point actually exists on the boundary of the trachea. The isolated point here refers to the point which is a vertex of only, one associated triangle of P_1 it is not the vertex of another associated triangle of P_1.

Each boundary point has two neighbouring points on the boundary. The two boundary points are isolated vertices of triangles associated with this boundary point. Because P_1 is on the boundary, there must exist at least two triangles on its two sides among all its associated triangles.

In Figure 3.4, point P_1 is the seed point on the boundary with the lowest Z coordinate found among the entire point cloud. Three triangles, ΔP_1L_1A, ΔP_1AB and ΔP_1BP_2, are found to be associated with P_1. Points A and B are not isolated points, they are the common points shared by two adjacent triangles associated with P_1: $\Delta P_1 L_1A$ and ΔP_1AB, and ΔP_1AB, and ΔP_1BP_2, respectively. Points P_2 and L_1 are the isolated vertices of associated triangles: $\Delta P_1 L_1A$ and ΔP_1BP_2. neither poiny being shared by any other triangle associated with P_1. Therefore, P_2 and L_1 are points on the boundary.

(4) Use one of the two points found in Step 3, P_2, as the new seed point, repeat Step 2 and 3. Of the two newly found points, the one which is not P_1 is the new boundary point.

In Figure 4.4, assuming that P_2 is used as the new seed point, P_3 and P_1 are the newly found points from triangles associated with P_2. Because $P_3 \neq P_1$ it is certain that P_3 is a point on the boundary.

(5) Repeat Steps 2, 3 and 4 to identify all points on the boundary.

The iteration stops when the last boundary point is found to be P_1. Because the geometry of human airway or arterial vessel is tubular, by $P_i = P_1$, it means the

iteration has come to the end of the loop and all points on the boundary have been recognized.

4. Search for points in the 2nd layer

(1) Mark all triangles in Area 1 as "Non-visited".

(2) All points on the boundary which are found in Phase 3 belong to the first layer. Select the first and lastly found boundary points in the first layer (Points P_{11} and P_{1L} in Figure 3.5(a)) and find the triangles associated with them. Both points belong to the "Non-visited" triangles.

(3) Identify the common triangle shared by $P_{11}$and P_{1L}. The third vertex P_{21} of this triangle is a point in the second layer. There is no ambiguity about this common triangle because no overlap between triangles exists in the geometries described by the STL file.

(4) Mark this triangle as "Visited" and delete it from the group of triangles which associates with point P_{11}.

(5) Find all points in the 2nd layer which are associated with Point P_{11}.

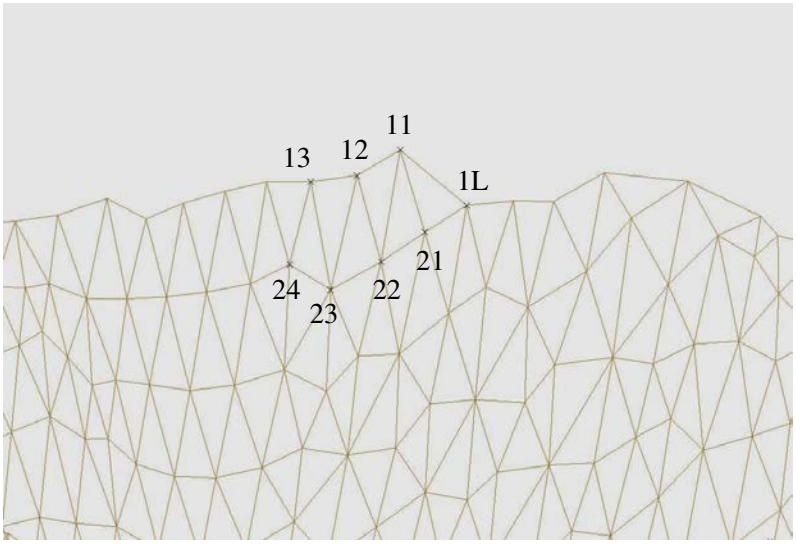
(i) Among the non-visited triangles associated with P_{11}, find the common triangle shared by P_{11} and P_{21}. The third vertex of this triangle is a point in the 2nd layer (P_{22} in this case).

(ii) Mark this triangle as "Visited" and delete it from the group of triangles associated with Point P_{11}.

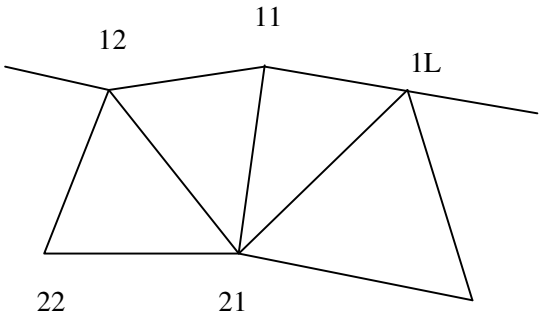
(iii) Replace P_{21} with the newly found point in the 2nd layer and repeat Step (i) and (ii), until the point found is a point in Layer 1 (P_{12} in this case).

Two points (P_{21} and P_{22}) associated with Point P_{11} are found in Layer 2 in Figure 3.5(a). Because flags of triangles which had been visited have been changed to "Visited", they will not be processed again in Step (i), therefore, the point in Layer 1 found in Step (iii) must be Point P_{12}, it can not be Point P_{1L}.

For the case where only one point in Layer 2 is related to Point P_{11} (Figure 3.5(b)), the point found in Step (i) will be P_{12}, a point in Layer 1 other than a point in Layer 2. Therefore, the checking process used in Step (iii) should be applied in Step (i) as well.



(a)



(b)

Figure 3.5 (a) Search for points on the 2nd layer with the layer-by-layer method (b) A point on the 1st layer relates to only one point in the 2nd layer

As shown in Figure 3.5(a), P11, P12, P13 and P1L are points in the first layer. Point P11 is the first point used in the searching process, while Point P1L is the last one. Points P21, P22, P23 and P24 are points in the second layer. Point P21 is the first point that is found on Layer 2.

(6) Mark all triangles associated with this point (P_{11}) as "visited", and delete these triangles from the "Non-visited" group.

(7) Among the Non-visited triangles, locate all the triangles associated with the second point (P_{12}) on Layer 1.

(8) Repeat Steps 5, 6 and 7 until all points in the first layer are processed.

(9) Verify if the points in the 2nd layer formed a closed circle.

5. Search for points in the next layer by using the layer-by-layer method

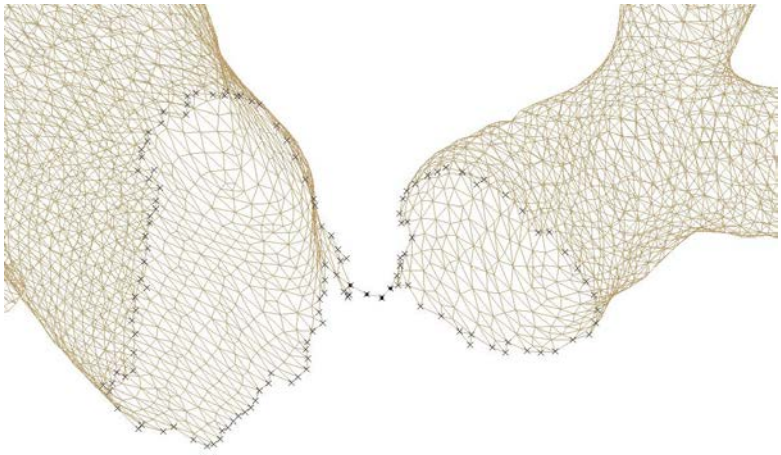
   Repeat Phase 4 to find all points in the next layer. If the maximum Z coordinate is found to be close to the minimum Z value predefined in this region (region *i*), triangles being processed are now at the bottom end of the region Triangles in the next region (region *i*+1) will be added to the group of "Non-visited" triangles to ensure that no triangle or point will be missed at the boundary between the two adjacent regions. The searching process stops when the next layer is found to be an open cycle, or two closed cycles can be formed with the points in that layer. The two closed cycles can be described in three ways:

   - Two cycles share one common point, as shown in Figure 3.6(a)
   - Two cycles share more than one point (Figure 3.6(b))
   - No overlap exists between the two circles, but there are triangles which link the circles together (Figure 3.6(c)).

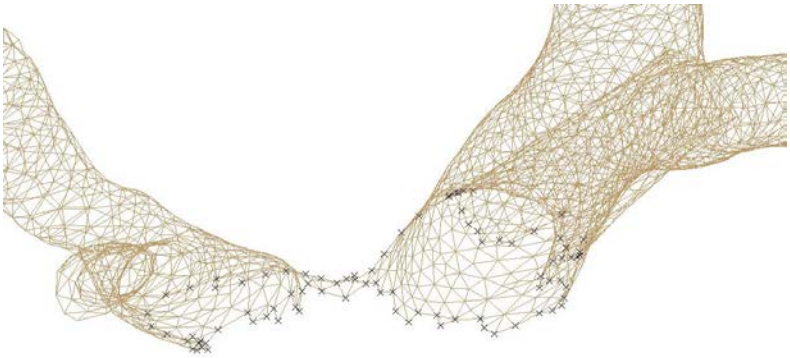
   If two closed circles are formed, it can be concluded that the search has reached the bifurcation layer and a new branch has emerged on this layer. All the points from the previous layer through this bifurcation belong to this new branch.

6. Regard the points on the closed circle as the points on the "first layer" and repeat Phase 3 and 4. All points on this branch should be recognized.

(a)



(b)



(c)

Figure 3.6 (a) Two circles share one common point (b) Two circles share more than one point (c) A triangle exists to connect two non-overlapping circles.

3.3.2 Reconstruction of surface patches

The surface patches in each branch are constructed by using the shortest distance algorithm which has been used by Eric and Hoppe (Eck et. al 1996) in surface re-construction i.e., through minimizing the distance function:

$$\text{Edist}(S) = \sum_{i=1}^{N} d^2(P_i, S) \qquad (3.1)$$

where: Edist( ) is the distance function, d( ) is the distance, Pi is the point , S is the surface patch.

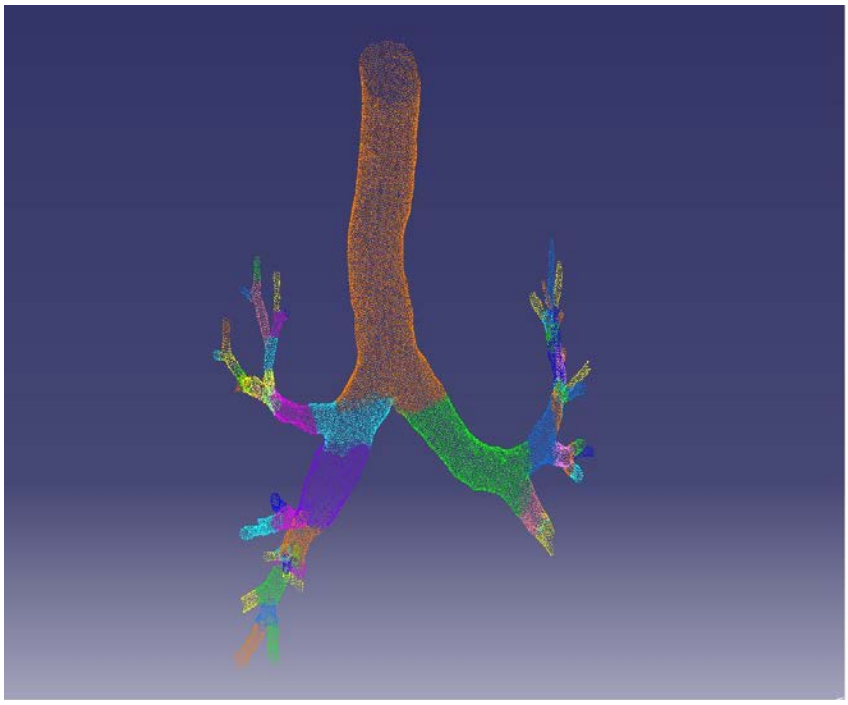
The main advantage of this algorithm is that it has the capability of dealing with a large number of points and can deal with the cloud points including noisy data. It assumes a set of data points to be on or near an unknown surface. The signed distance from an arbitrary point to a known surface is then defined. The availability of the signed distance function is equivalent to that of the surface because an implicit representation of the surface is given by the zero set. In other words, although the surface is unknown, it can be estimated by firstly

estimating the signed distance from the data points, and then extracting an approximation of its zero set. The key aspect of estimating the signed distance function is to associate an oriented plane with each of the data points. These oriented tangent planes are then used to define the signed distance function to the surface.

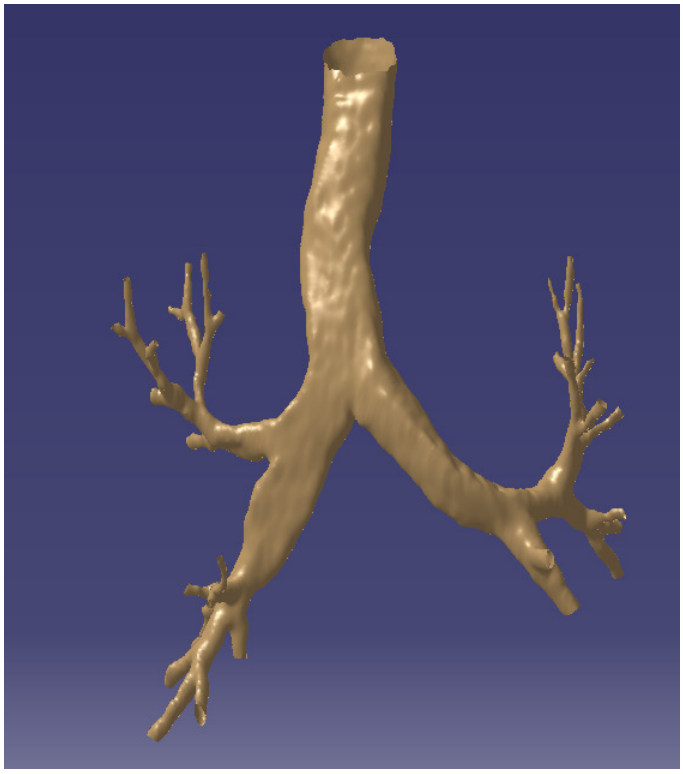
For large branches like the trachea, the point clouds are further divided into sub groups if there are too many points.

3.4 Examples

The algorithm is implemented with Matlab and Visual C++. The recognized branches of the carotid vessel and the airway tree are shown in Figure 3.7. Figure 3.7(a) demonstrates the recognized branches or generations of the airway tree which consists of 46810 points. Figures 3.7(b) illustrates the corresponding solid geometric model.



(a)



(b)

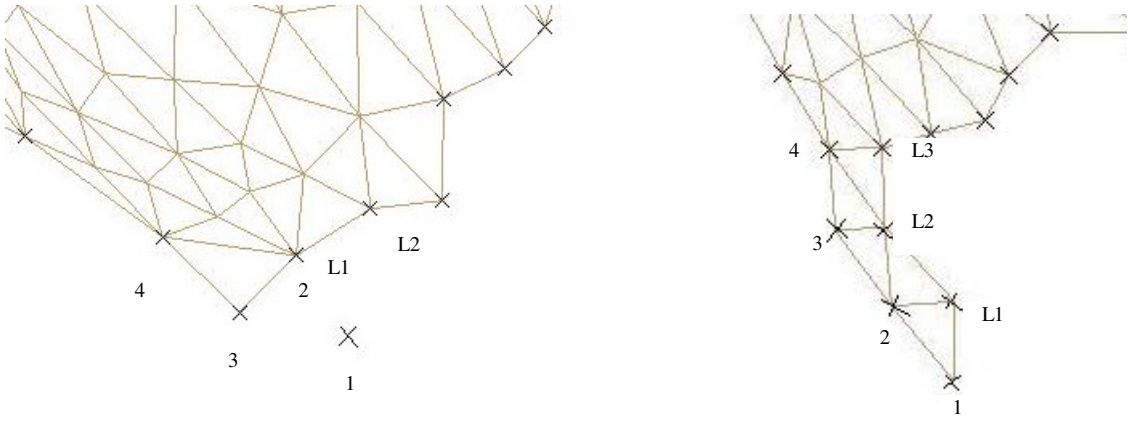
Figure 3.7 (a) An airway tree with 7 generations of bronchi (b) solid model of the airway tree

3.5 Discussion

In the searching process in Step 5 of Phase 4, it is critical to ensure that all points are on a closed circle. The iteration can not start and stop properly if the circle is not closed. Two types of closed cycle may be encountered in the searching process:

1 If the last point overlaps the points identified in the first two iterations, the points before the overlap should be cut off. Ten points are used in our preliminary tests. As shown in Figure 3.8(a), Point 2 and Point L1 overlap; therefore Points 1 and 2 should be cut off.

2 If the points identified in the last two iterations do not overlap with any points obtained in the first two iterations (ten iterations are used in our calculation), but connecting triangles exist to link them together, the points in the connecting triangles should be cut off.



(a) (b)

Figure 3.8 (a) Overlap points exist in closed loops (b) Linking triangles exist in closed loops

As shown in Figure 3.8(b), Points 1，2，3 and 4 are points identified in the first 4 iterations, Points L1，L2 and L3 are identified in the last three iterations. Here, Point L1 connects to Point 2, Point L2 connects to Point 2 and 3. Point L3 connects to Point 4. Because “4” and “L3” are the highest in the two iterations, the points in the linking triangles – Points 3, 2, 1 and Points L2, L1, are cut off.

3.6 Summary

Geometric modelling is an integrated step in the generation of geometric models from medical images. A new modelling algorithm is presented in this thesis to reconstruct human airway and arterial vessels which consist of tubular geometries with large differences in diameters. The method consists of two steps. The first step is to recognize the branches of

the airway tree and bifurcations of the arterial vessels from the point clouds obtained from medical images, by using a new searching algorithm. On each branch the surface patches are constructed according to the number of available points by using an established method. In this way, different approximation accuracy can be applied in different branches with the availability of the branch information, which can lead to better surface quality.

The topological relationship between branches and the relative locations of each branch could be identified once the searching direction is fixed in each layer. Because the points on each branch have been grouped together, the new searching algorithm can also be utilized in the labelling of airway trees.

Chapter 4
Inhalation effort comparisons in a bronchoconstricted and recovered airway tree associated with acute asthma

4.1 Introduction

Asthma is a chronic inflammatory disorder of the airways affecting mainly the medium-sized and small bronchi (Travis et al. 2002) characterised by a sudden or prolonged onset of airway narrowing. Long term effects of the disease can be characterised by variable degrees of tissue and structural remodelling in the airways that lead to a progressive loss of lung function (Pascual and Peters 2005; Vignola et al. 2000). During an acute asthma episode, the combination of bronchospasm, mucus plugging, and mucosal edema leads to increased airway resistance as the diameters of the airways are reduced. An increase in the inhalation effort is therefore required to obtain similar tidal volumes in unconstricted airways, which then leads to respiratory muscle fatigue. The increase in airway resistance also causes a lack of uniformity in ventilation throughout the lung causing hypoxemia from ventilation-perfusion mismatching. In addition the changes in the geometry and inhalation efforts will have a significant effect on drug delivery devices such as inhalers that are designed to deliver pharmacological agents to affected areas within the tracheobronchial tree. This paper descripts reconstructed airway tree models of an acute asthma episode and following recovery from the same patient performed thirty days apart. The computational model was created for Computational Fluid Dynamics (CFD) analysis which provides details of the geometry and airflow dynamics such as the pressure drop. A comparison between the two models was undertaken to determine the disparities in the inhalation efforts and the airway branch diameters caused by physiological changes in the airways.

Airway geometry details and pressure drops are often difficult to acquire completely and accurately using experimental methods because of their invasive nature. Numerical studies

such as those performed by (Comer et al. 2000a, b; Zhang et al. 2002a, b) provide a non-invasive technique which creates an opportunity to obtain fine details of the airway. However the majority of studies involving airflow dynamics such as pressure drop have been based on regular dichotomy airway models (symmetrical or idealised geometries). Irregular dichotomies have also been studied but to a lesser extent. This may be due to the limitations of technology in the past to provide the detailed data to construct realistic models. The studies of Balashazy and Hofmann (1993), Balashazy et al. (2003), Choi et al. (2007) and Kim and Fisher (1999) are some of the few published investigations in which an irregular dichotomy airway model was used. One advantage of using a realistic model is the inclusion of the effects of cartilaginous rings present in the trachea. It has been shown by Zhou and Zhang (2005) that there is a disturbance in the airflow within the trachea caused by the presence of cartilaginous rings but this influence does not propagate to bifurcations further downstream. Recently, with advances in medical imaging technology, high resolution CT scans have provided a pictorial insight into the manifestations of asthma including thickening of the bronchial wall, narrowing of the bronchial lumen, areas of decreased attenuation, and air trapping on expiration (Isabela et al. 2004).

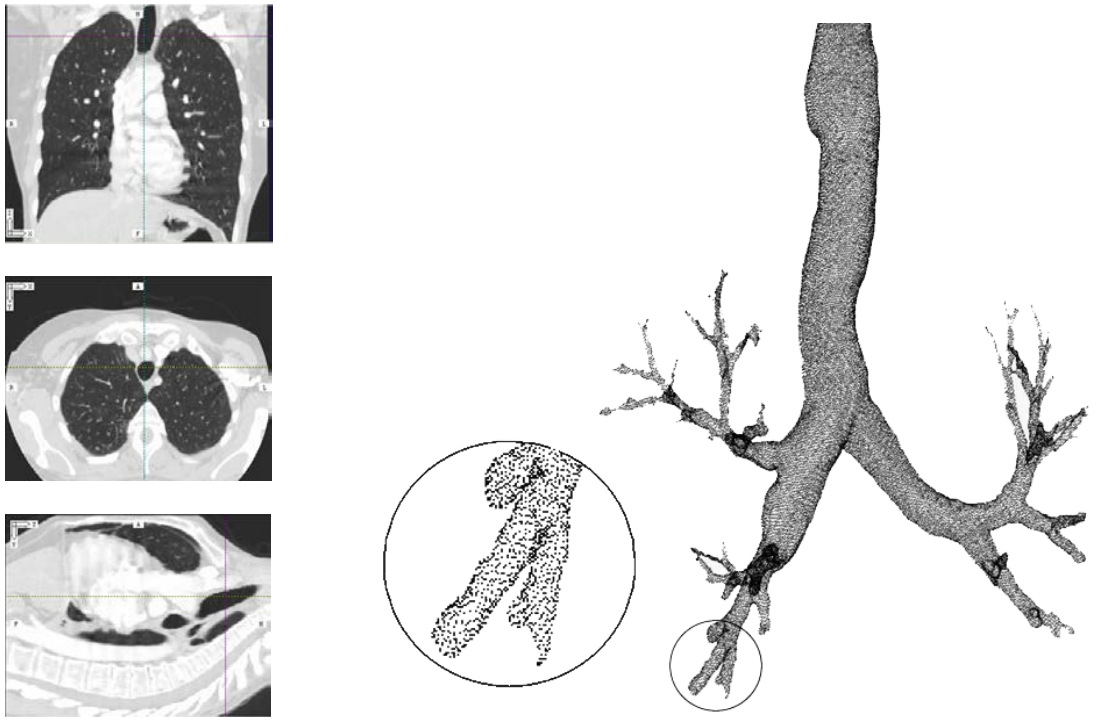
In terms of validating the numerical results, (Zhang et al. 2002a, b) and (Comer et al. 2000a) have extensively compared their numerical results with experimental results in regard to the velocity profile. The influence of the larynx on the airflow in the trachea and its lower airways has been included in many experimental and numerical simulations through complete upper airway analysis (from oral cavity to lower generations). Zhou and Cheng (2005) indicated that the deposition efficiency of inhaled particles with the larynx effect is generally larger than without it and hence the larynx affects the flow pattern of the tracheal region. They also suggest that the turbulence induced by the laryngeal jet could likely persist in the first few generations of the tracheobronchial airways, resulting in higher deposition efficiencies. However, (Zhang et al. 2005) also showed that the larynx effect causes turbulent fluctuations at medium and high inspiratory flow rates (30 and 60 L/min) because of the enhancement of flow instabilities just upstream of the flow dividers. Due to the low inspiratory flow rate used in this study and the complex nature of the laryngeal and tracheal

regions, the inclusion of the larynx effect, oral cavity, nasal cavity, and larynx were not considered.

4.2 Methods

4.2.1 Geometry generation

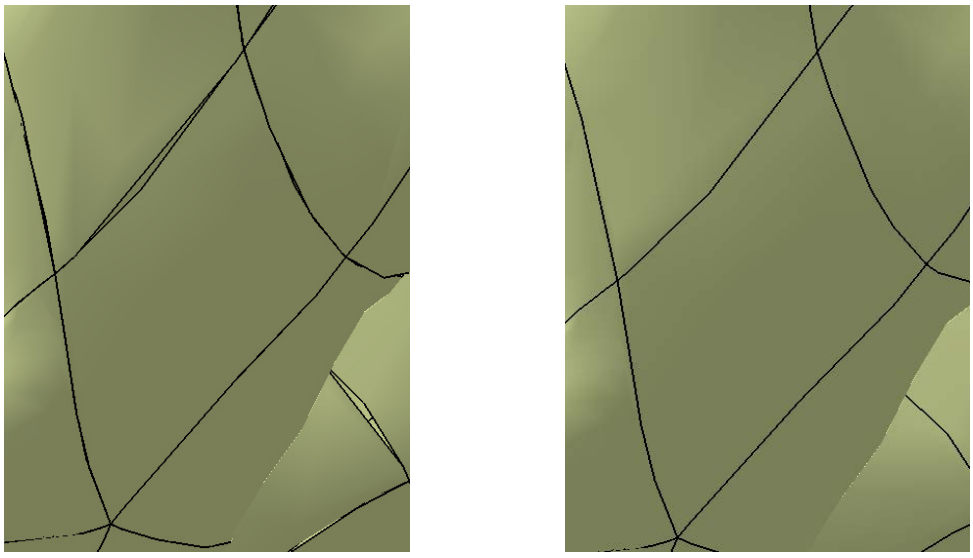
The reconstruction of the asthma and recovered airway tree geometries was based on data obtained from CT images shown in Figure 4.1. The geometry was generated through three steps, image segmentation, surface reconstruction and surface fitting. More details are in Chapter 2.



(a) (b)

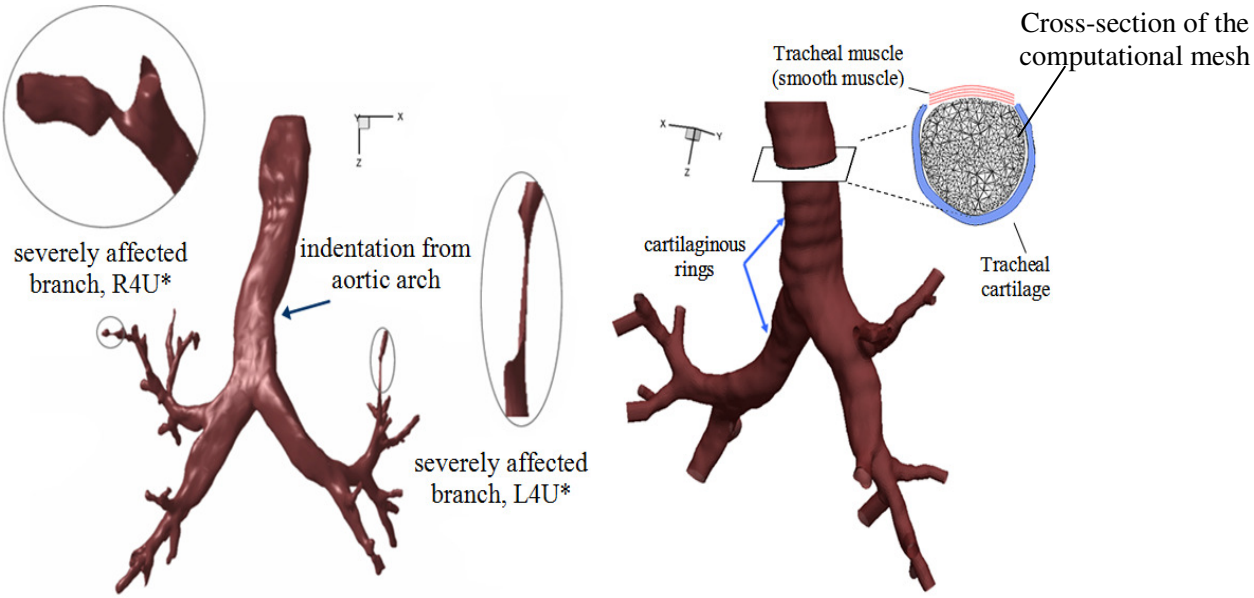
Figure 4.1 (a) CT scans of the airway (b) the resulting cloud points of the airway tree obtained from the image segmentation stage.

The geometry obtained using the methods outlined in Chapter 2 still contains many flaws such as overlap and gap. Manual work, referred to as boundary bound surface generation manipulationis performed in CATIA (CATIA Inc) with the algorithm in this study developed to deal with these problems. A surface patch which is overlapped or has gap with other patches is deleted firstly. Then a new surface patch is generated by using its adjacent four boundaries as bound (Figure 4.2). The modified asthma affected and recovered models are shown in Figure 4.3. Figure 4.3(a) shows that two branches of the acute asthma affected airway model were severely affected by the airway narrowing. These branches were modified and plugged at their ends acting as blocked airways. In terms of CFD modelling this was necessary as it is difficult to apply a suitable computational mesh on such small dimensioned narrowed geometries. Figure 4.3(b) is a reverse view of the recovered model which shows the presence of the tracheal cartilage rings and the airway smooth muscle. The transverse cross-section depicts the computational mesh used for the CFD analysis.



(a) (b)

Figure 4.2 Boundary bound surface generation (a) gap between patches (b) after regenerated patch



(a) Acute Asthma (b) Recovered

Figure 4.3 **(a)** A frontal view of the reconstructed computational model of the acute asthma affected airway. **(b)** A reverse view of the recovered model showing the presence of the tracheal cartilage rings and the airway smooth muscle.

4.2.2 Mesh and boundary conditions

Using the meshing software GAMBIT 2.2 (ANSYS Inc.), the face, volume, mesh and extension tubes at outlets were created and a mesh file was produced, which was then read into FLUENT 6.2.16 (ANSYS Inc.). For this comparative study a constant entrance profile was used with the inlet extended further back to allow the inlet velocity to develop naturally. Therefore, by allowing the flow to develop, a more accurate inlet velocity profile at the trachea was formed. The developing length formula (laminar flow = 0.05Re*Diam) was used in this study. This rationale for extending the trachea to allow flow development is that the velocity profiles are susceptible to being skewed by the upper carina ridges (Yang et al. 2006). Zhang et al. (2005) also demonstrated that different flow rates had some effect in the trachea region after the glottis and hence the profile would be distorted somewhat due to additional fluid dynamics such as secondary flows and turbulent dispersion.

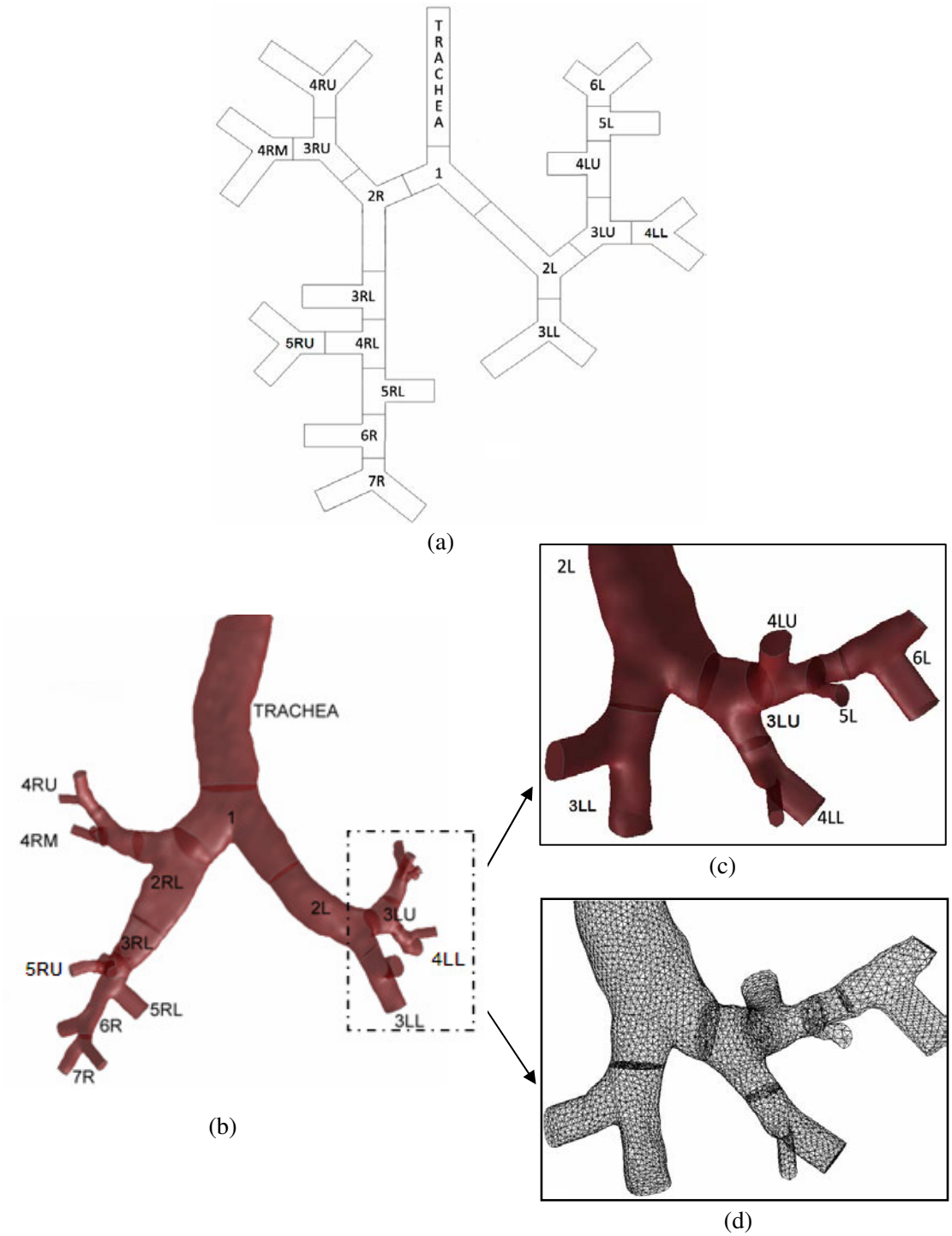


Figure 4.4 (a) Schematic diagram with branch identification inside each branch. (b) front view of the 3D airway model. (c) magnified frontal view of the selected section rotated (d) computational mesh of the selected section.

An initial model with a coarse mesh was simulated to establish data to be used for analysis to improve the mesh. The model was then refined by cell adaptation techniques which included refining large volume cells, cells that displayed high velocity gradients, and near wall refinements. Average velocity profiles within the airway were taken for each model and compared. The velocity profiles became independent of the grid size as the number of cells in the model reached one million. Figure 4.4 (a) shows the schematic diagram with branch identification inside each branch. The numbers assigned represent the generation number. The second letters represent the Left or Right side of the lung. The third letters, U, M and L represent upper, middle and lower respectively. Figure 4.4 (b) is a front view of the 3D airway model. Figure 4.4 (c) shows a magnified frontal view of the selected section rotated. A small section showing the mesh density is given in Figure 4.4 (d). For this study the method adopted from Balashazy and Hofmann (1993); Longest et al. (2006) where a parabolic inlet profile from an extended trachea is used. In a recent study by (Li et al. 2007), it was found that the type of velocity inlet condition and existence of cartilaginous rings influences the air flow field; however, their impact is less important in comparison with the variations in the upper airway geometry, e.g., branch curvature. A constant flow rate of 15L/min was applied, which corresponds to a Reynolds number of 1079. In recent studies the flow regime has been considered transitional for flow rates of 15-60 L/min and for flow rates greater than 30L/min (Re~2500) based on the existence of the laryngeal jet producing transitional behaviour. Other researchers however, have applied a laminar flow regime for flow rates of 28-30L/min (Re~2000-2500) (Nowak et al. 2003; van Ertbruggen et al. 2005) based on the argument that RANS turbulence models cannot predict the transitional behaviour reliably, that the flow within the lung branches rapidly becomes laminar after the initial bifurcations, and that the flow is mostly laminar because the Reynolds number is globally below the transitional value. In an investigation into the turbulence structures using Large Eddy Simulations (LES) showed that at a flow rate of 15L/min inlet turbulence is damped out and that laminar flow prevails. Interestingly deposition patterns were found to be unaffected by turbulent dispersion at 15L/min, although particle deposition is enhanced by turbulence for flow rates of 30 and 60 L/min when $St \leq 0.06$. Based on these findings and the limitation of this study (exclusion of the larynx) a steady laminar flow is used for a flow rate of 12L/min. In addition using a turbulence model for flow rates that exhibit some light transitional flow does not necessarily imply greater accuracy. For example RANS turbulence models are designed for fully developed turbulent flows, although modifications have been made to formulate low-Reynolds number models

that reproduce the transitional behaviour. However in these models the physics of transitional flow behaviour is not actually resolved but rather a modelled approach is applied. Furthermore in local regions of low flow rates, where laminar effects are dominant, turbulence models provide greater diffusion due to the turbulence production inherent in turbulence models. The numerical solutions of the fluid flow equations were obtained using the commercial finite volume based program, FLUENT 6.3 (ANSYS).

4.3 Results

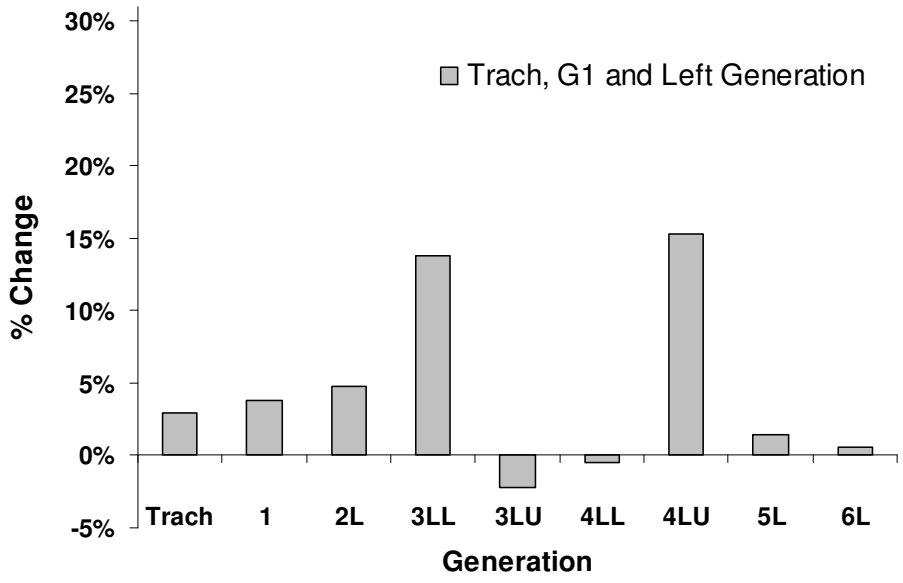
4.3.1 Airway geometry comparisons

The equivalent hydraulic diameter, Dh defined by $D_h = 4A/P$, where A is the cross-sectional area and P is the perimeter, was calculated for the entrance and bifurcating ends of each generation branch. The diameters given in Table 4.1 for each generation branch are taken from the averages of the three diameters from the entrance and two bifurcation ends and are compared between the acute asthma model, the recovered model and results from other airway models (Choi et al. 2007) and (Weibel 1963). In general the right side of the airway is larger in diameter than the left side. The recovery is measured as the dilation of the inner diameter given in terms of the percentage increase in the average diameter and may be thought of as dilation of the airway smooth muscle (ASM). A large increase was expected for the branches that were severely affected by asthma (4LU and 4RU). Besides these affected branches, 3LL and 3RU had a dilation greater than 10% (Figure 4.5). Overall the right airway exhibited greater dilation than the left airway especially from the fifth generation onwards. Three generation sections, 3LU, 4LL and 6R showed minor negative increases (-2.2%, -0.5% and -1.8% respectively) which means that these sections were actually slightly more constricted. It is uncertain if this is within the margin of the modelling error or whether it reflects a real physiological feature. This may be due to the after effects of airway narrowing by inflammation and edema of the lining airway mucosa, or even the accumulation of mucus and other fluids, which can plug the airways. It supports the

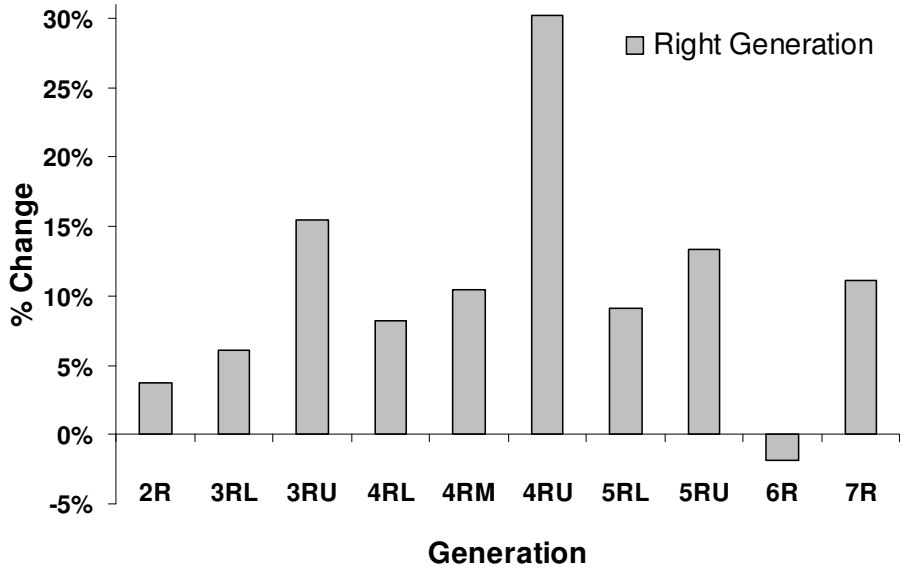
heterogeneity of bronchoconstriction in the airway tree in acute asthma, which leads to major shifts in ventilation with the potential for catastrophic bronchospasm (Venegas et al 2005).

Table 4.1 Average diameter of the tracheobronchial tree model (dimensions in millimeters)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Acute Asthma	18.0	15.4	2R 11.2 2L 8.8	G3R.U 5.0 G3R.L 8.2 G3L.U 6.2 G3L.L 5.8	G4R.U 3.7 G4R.L 6.6 G4L.U 5.1 G4L.L 4.3	G5R.U 3.3 G5R.L 5.5 G5L 3.8	G6R 4.8 G6L 3.6	G7R 4.1
Recovered	18.5	15.8	G2R 11.6 G2L 9.2	G3R.U 5.8 G3R.L 8.7 **G3L.U 6.1** G3L.L 6.6	G4R.U 4.0 G4R.L 7.1 G4L.U 5.3 **G4L.L 4.3**	G5R.U 3.8 G5R.L 6.0 G5L 3.8	**G6R 4.7** G6L 3.7	G7R 4.5
Choi et al. (2007)	16	13-16	10-14	6-11	5-8	4-7	N/A	N/A
Weibel (1963)	19	9-16	7-11	5-7	4-6	3-4	N/A	N/A



(a)

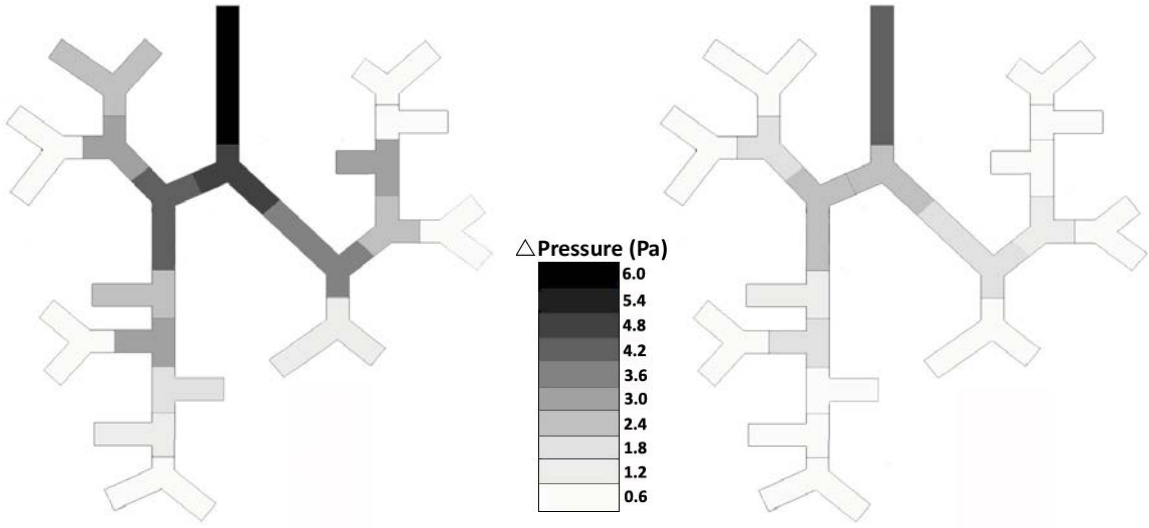


(b)

Figure 4.5 Percentage increase in the averaged diameter from the acute asthma model to the recovered model.

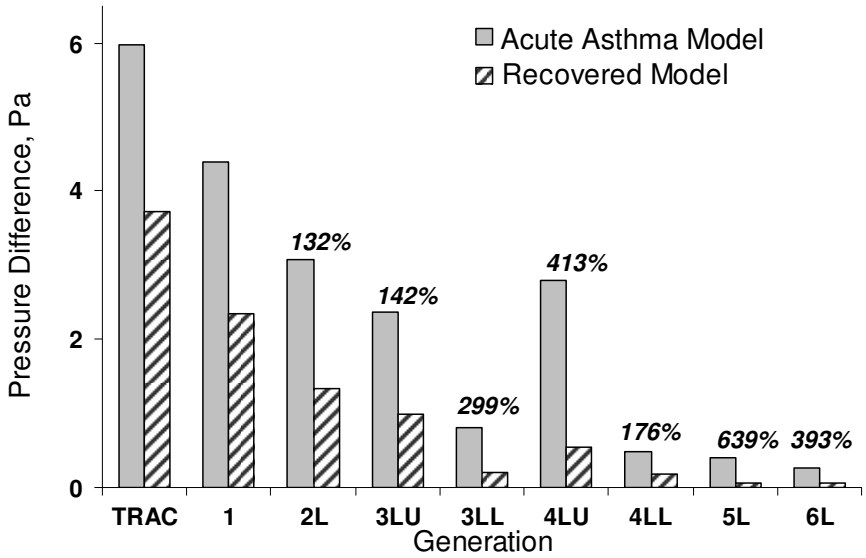
4.3.2 Inhalation effort comparisons

Pressure drop in the bifurcating airways plays an important role in the respiratory process. For a flow rate of 15L/min the local pressure distribution in each airway branch was recorded and is shown in Figure 4.6. The required pressure difference at the inlet for the AA-model was 5.98 Pa, which is nearly twice the value for the recovered model (3.73 Pa).

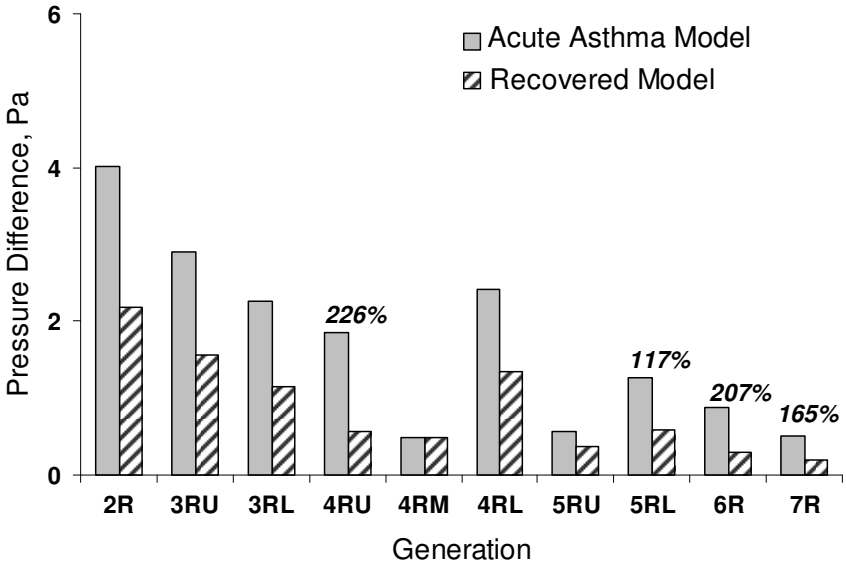


Acute asthma pressure distribution Recovered pressure distribution

Figure 4.6 Distribution of the average pressure difference between the airway generation and the pressure at the outlets.



(a)



(b)

Figure 4.7 Branch comparisons for the acute asthma model and the recovered model. (a) central and left side of the airway (b) right side of the airway.

Comparisons between the two models show that larger pressure differences occur in the AA-model within the first four generations of the airway particularly branches 4LU and 4RU. Branch comparisons in Figure 4.7 show vast pressure differences between the two models despite the same flow rate. The percentages show the increase in the pressure difference between the A-A model and the REC model for the specified generation. Only increases above 100% are displayed in this figure. The difference is mainly due to the two affected branches in the AA-model which effectively reduces the number of flow paths by two as the constriction in 4LU and 4RU is so great that no flow passes through. In all the branches except for 4RM higher pressure is experienced in the AA-model. The greatest percentage change in the pressure difference from the AA-model to the REC-model is found on the left side of the airway where seven branches experience a change of over 100%. Greater resistance is found in the AA-model because of the reduced cross-sectional areas in the AA-model (Figure 4.5).

4.4 Discussion

Recovery within the airways was measured by the increases in the diameters of the airways within the tracheobronchial tree, measured thirty days after the initial episode of asthma. The increase in diameters also reflects the increase in the total lung capacity that is available as a larger diameter would provide a greater volume. The right airway exhibited a greater percentage increase in the diameters in comparison with the left side. This correlates with the fact that the right airway for the patient was also larger in diameter than the left side. Whether this correlation is a physiological feature caused by the airway size as one determinant for the magnitude of smooth muscle contraction and hence a cause for airway hyperresponsiveness (AHR) is uncertain. This uncertainty is due to the inherent role of the airway structure as a determinant of airway narrowing which is not a simple, well known function. Rather AHR is caused by the manifestations of complicated relationships between the magnitude of ASM shortening, baseline airway structure, and changes in lung volumes or airway diameters.

There is general agreement that the total amount of smooth muscle is increased in asthma (Carroll et al. 1993; Ebina et al. 1993; Kuwano et al. 1993), while increased muscle mass has the potential to enhance airway responsiveness, thus contributing to asthma symptoms (Lambert et al. 1993). The larger airways present greater complexities as they do not behave isotropically unlike smaller airways when ASM is stimulated (Mitchell and Gray 1999). More recently in a published viewpoint by (Permutt 2007), it was hypothesised that dynamic hyperinflation caused by narrowing of larger airways is a major determinant of airway hyperresponsiveness in asthma. The study by (Brown et al. 2006) shows that airway structure and lung volumes are important determinants of AHR and pulmonary function, while the work of (Lundblad et al. 2007) and (Wagers et al. 2007) suggest that a link exists between airway structure and its function that manifests as a change in lung volumes. While these data have supported the hypothesis, other researchers (Sorkness 2007; Thompson et al. 2007) in response have cast doubts over the claim and hence the role of large airways on smooth muscle contraction in asthma remains controversial. Additionally, doubts may be cast as a result of the conversion of the CT scans into a computational model during the image processing stage. Regardless of the underlying doubts, the geometrical data provides a quantitative driven perspective on the issue and provides a useful data set for further comparative studies into the physiological effects of asthma.

Ventilation in the respiratory airway is accomplished by the transport of inspired air down pressure gradients within the airways. This process involves the alternating contraction and relaxation of the respiratory muscles which overcomes the pressure drop caused by viscous losses such as shearing forces within the fluid, the friction between the air and walls of the airways, and the resistance presented by the irregularities of the airways. Under steady laminar flow conditions of 15L/min the required effort by the respiratory muscle to overcome the pressure difference for the AA-model is nearly twice as high as the REC-model. The primary cause of greater resistance in the AA-model is the narrowed airways and the occlusion of two of the branches. This suggests that during the period of an acute asthma episode, the work of breathing for the patient in order to achieve the same tidal volumes, is double that of the recovered state, which can lead to respiratory muscle fatigue. In reality the

patient may breathe with greater effort invoking deep inspiration. In this case the flow may become turbulent which increases the amount of viscous loss effects. The walls created within the model are assumed rigid and smooth whereas in reality the walls may exhibit some roughness and elasticity. The inclusion may alter the predicted magnitudes of the pressure drop but it is unlikely to alter the ratio of the two pressure drops from the computational models. In addition the inclusion of these attributes may actually cause further errors as the elasticity and roughness of the walls are not accurately known.

4.5 Conclusion

The effects of hyperresponsiveness as a result of asthma on the physical geometry of the tracheobronchial tree were measured by computational modelling. It was found that the right sided airway tree had greater bronchodilation than the left airway tree following recovery after an acute asthma episode in this individual. This correlated with the fact that the right airway was larger in size. The correlation partially supports the hypothesis that dynamic hyperinflation associated with widespread airway narrowing can be a major determinant of airway hyperresponsiveness in asthma.

Under steady laminar flow conditions at 15L/min the required effort by the respiratory muscle to overcome the pressure difference for the AA-model is nearly twice as high as the REC-model. This requirement suggests that the respiratory muscles exert more effort during an asthma attack in order to achieve the same tidal volumes which can lead to respiratory muscle fatigue. Additionally the flow patterns within the airways will also exhibit greater velocities and have a follow-on effect on the delivery of drug particles.

Chapter 5
Comparative study of airflow patterns from the effects of acute asthma on the tracheal-broncho airway tree

5.1 Introduction

After the comparison of the geometry and inhalation effort described in Chapter 4, a comparative study of airflow patterns was then conducted between acute asthma and following recovered airway trees in terms of velocity profile and local flow features. The geometry models of acute asthma and recovery airway trees were the same as those described in Chapter 4. The numerical modeling method used here is superior to the method described in Chapter 4 and is described in this chapter. During the analysis process, advanced visualization technique was used. Therefore the potential "hot-spots" of high aerosol concentration could be immediately and easily observed. Results are validated with experimental data from other studies. Finally, some recommendations are drawn which could be useful to clinical researchers.

5.2 Methods

5.2.1 Numerical modeling

Acute asthma and following recovered airway geometry models used here were the same as described in chapter 4. After obtaining geometries, using the meshing software GAMBIT 2.2 (ANSYS Inc.), the face, volume, mesh and extension tubes at outlets were created and a mesh file was produced, which was then read into FLUENT 6.3 (ANSYS Inc.). To model physically accurate profiles in the lung the entire effects of the upstream larynx jet should be considered. However, the focus of this study was to evaluate the effects on air flow patterns of the physical changes in the airway caused by acute asthma. It is assumed that it is not

necessary to fully simulate the effects of the laryngeal jet as these effects will be dampened by the trachea curvature and carinal ridges found in the current airway geometry. In addition other studies of local bifurcation downstream of the trachea have neglected the upstream effects and assumed a constant, parabolic or blunt inlet profile (Balashazy and Hofmann 1993; Longest et al. 2006; Zhang and Kleinstreuer 2002; Zhang et al. 2005). For the present study the trachea was extended the required distance to allow the flow to develop naturally so that a more accurate inlet profile at the trachea was formed, rather than using a uniform profile. The rationale for extending the trachea to allow flow development is that the velocity profiles take into account the effects of the carina ridges (Yang et al. 2006a) and the effects of the trachea curvature.

A flow rate of 15L/min was applied at the inlet, which corresponds to a Reynolds number of 1079. The low flow rate provides the advantages of simpler modelling by assuming a steady laminar flow based on the consideration that the flow throughout the airway is mostly laminar. The flow regime has been considered as laminar or lightly transitional at flow rates of 28-30L/min (Re~2000-2500) (Nowak et al. 2003; van Ertbruggen et al. 2005). To use a turbulence model for such a flow rate may be less accurate. For example two-equation RANS turbulence models are designed for flows that are fully turbulent, although modifications have been made to formulate low-Reynolds number models that reproduce the transitional behaviour from laminar to turbulent flow. However in these models the physics of transitional flow behaviour is not actually resolved and at low flow rates, where laminar effects are dominant, it may prove to be more accurate to use a laminar flow as the turbulence model may cause an over-prediction in the diffusion process. In addition sensitivity studies were performed to determine the differences between k-ω turbulence models and a laminar model, and showed little difference in the flow profiles. A steady flow was applied based on a variety of criteria (Isabey and Chang 1981; Slutsky et al. 1981; Sullivan and Chang 1991) such as the Womersley parameter, $\alpha = D/2\left(\omega/\nu_g\right)^{0.5}$; a variant of the Womersley parameter, $\alpha^* = 1/2\left(\omega D/0.0075u_{ave}\right)^{0.5} < 1$ (Pedley et al. 1977); and the

Strouhal number, $S = \omega D / u_{ave}$ where u_{ave} is the mean velocity, ω is the angular frequency of oscillation (=2πf), D is the diameter of the tube and v_g is the viscosity.

Homogeneous outflow conditions were assumed where the mass flow was evenly distributed across each of the outlets. Furthermore, the outlets were artificially extended downstream given by, $L_{extension} = 0.05 \,\mathrm{Re}\, D$ to obtain fully developed profiles and hence avoid any reverse flow , effects of which can arise from an abrupt end to the flow field. An initial model with 150000 cells was initially simulated. The model underwent mesh refinement by cell adaption techniques which included refining large volume cells, cells that displayed high velocity gradients, and near wall refinements. Velocity profiles were compared between each subsequent model until the profiles remained the same and hence became independent of the grid size. The final model consisted of 1.3million cells.

Due to the complex geometry of the airway tree, a commercial CFD code, FLUENT, was utilised to predict the continuum gas phase flow under steady-state, isothermal, and incompressible conditions through solution of the conservation equations of mass and momentum. These equations in Cartesian tensor notation are:

$$\frac{\partial}{\partial x_i}(\rho u_i) = 0 \tag{5.1}$$

$$\rho u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right) \tag{5.2}$$

which were discretised using FLUENT's finite volume approach. The third-order accurate QUICK scheme was used to approximate the momentum equation while the pressure–velocity coupling was resolved through the SIMPLE method. The convergence criteria for the air flow properties (resolved velocities and pressure) were set at 10^{-5}. The convergence criteria were further checked by comparing the simulation of airflows along the section at R2 with the convergence criteria values of 10^{-5} and 10^{-7}. The difference of air velocity profile between the two simulations was negligible.

5.3 Results

5.3.1 Airway geometry and velocity profile validation

The reconstructed model of the bronchial tree exhibits an asymmetric dichotomous branching pattern. The beginning of the bronchial tree begins with the trachea, which is a hollow cylinder in the shape of a horseshoe because of the C-shaped supporting cartilage found anteriorly and laterally. Completing the tracheal cylinder on the posterior side is a flat band of muscle and connective tissue called the posterior tracheal membrane. Along the airway downstream, the cartilage support becomes progressively smaller and less complete. An indentation can be found in the left lateral wall of the distal trachea caused by the aortic arch. Average branch diameters have been investigated as described in chapter 4 and are shown in Table 4.1. The bifurcation angles between daughter branches are given in Figure 5.1. Obtaining bifurcation angles involved manually orientating the airway to determine the middle axis between the two daughter branches. This method introduces some subjective uncertainty associated with manual detection of the middle axis, and is exacerbated for smaller branches. Bifurcation angles were limited to the main bronchi (L1, R1) and the upper lobes (R2, L2 and its extensions) as these branches are larger, affect the distribution of airflow more significantly, and are less prone uncertainty in the methodology.

The bifurcation at the carina produces the right main bronchus (R2, 41^{o}) at a more obtuse angle than that of the left main bronchus (L2, 25^{o}) measured by the interbronchial axis (centreline of the bronchus) with the long axis of the trachea. The sum of the two daughter branching angles (i.e. R2+L2 = 66^{o}) forms the tracheal carinal angle which compares with a mean value of 73^{o} from a sample of 65 men and 55 women; age range 17–85 years; mean age 56 years (Karabulut 2005) and 63^{o} for a sample of 21 patients with an age range of 51-60 years (Haskin and Goodman 1982). The right main bronchus continues and bifurcates posterior-inferiorly into the right upper lobe bronchus (R3U) and an intermediate bronchus (R3L). This bifurcation occurs earlier on the right than on the left lung. R3U is always found in patients and usually extends into three main branches: (i) the apical bronchus which is the immediate continuation of the upper lobe bronchus, (ii) the posterior bronchus (R4P) and (iii)

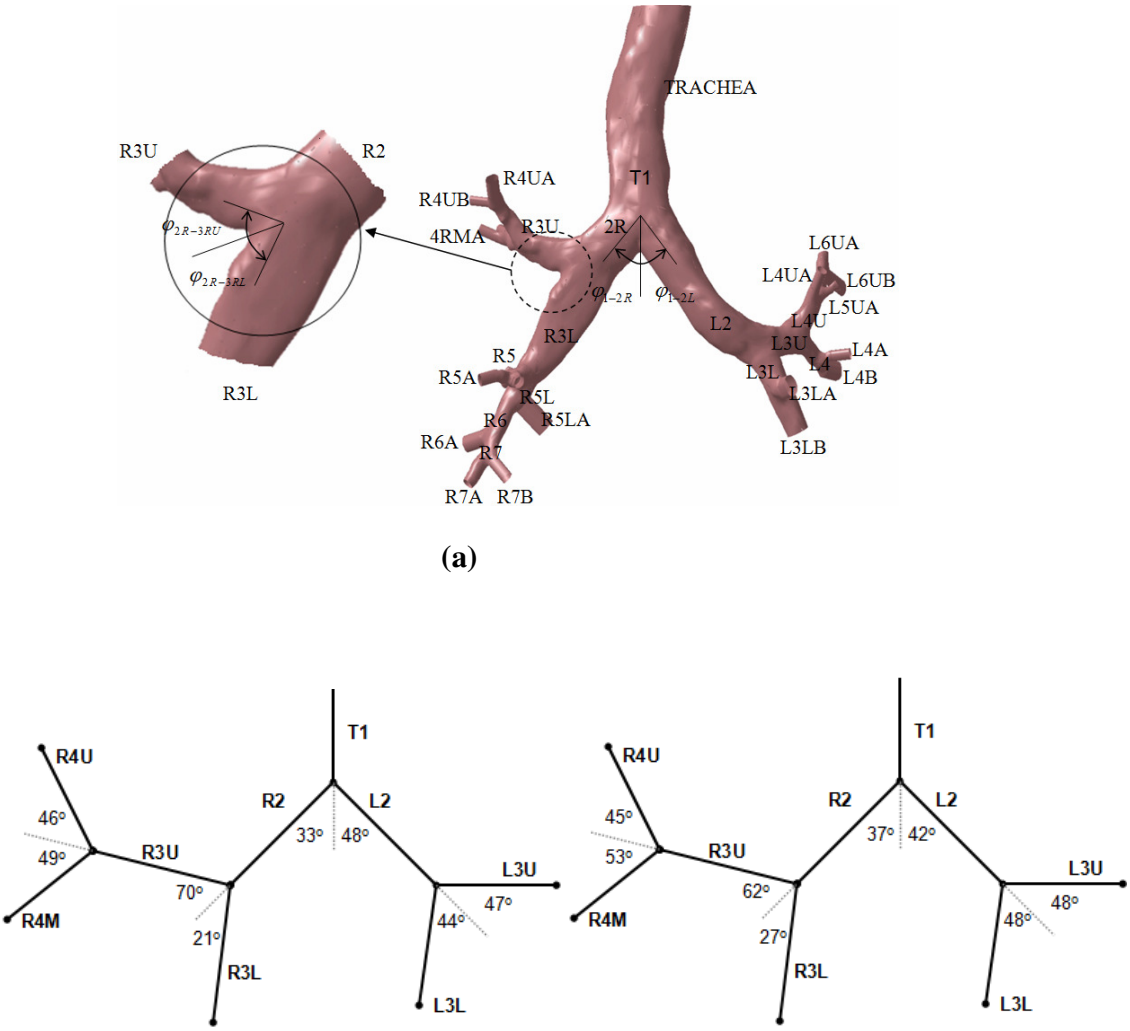


Figure 5.1 (a) Branch angle definition where two angles are defined at the bifurcation. (b) Recovered Model (REC-model) and (c) the Asthma Affected model (AA-Model)

the anterior bronchus (R4A). R3L is the branch that continues after R3U has bifurcated from it, and is of variable length. It has no branches over its length and directly continues as the

lower lobe bronchus (R4L). In the left lung, the left main bronchus (L2) is longer than R2 and extends laterally forming a gentle curve that runs almost horizontally. It bifurcates into the upper lobe bronchus (L3U) and the lower lobe bronchus (L3L). Overall the bifurcation angles do not vary significantly between the AA-model and the REC-model, however small changes are observed which may be due to the scans which were taken of the patient thirty days apart where the positioning and inhalation may have differed.

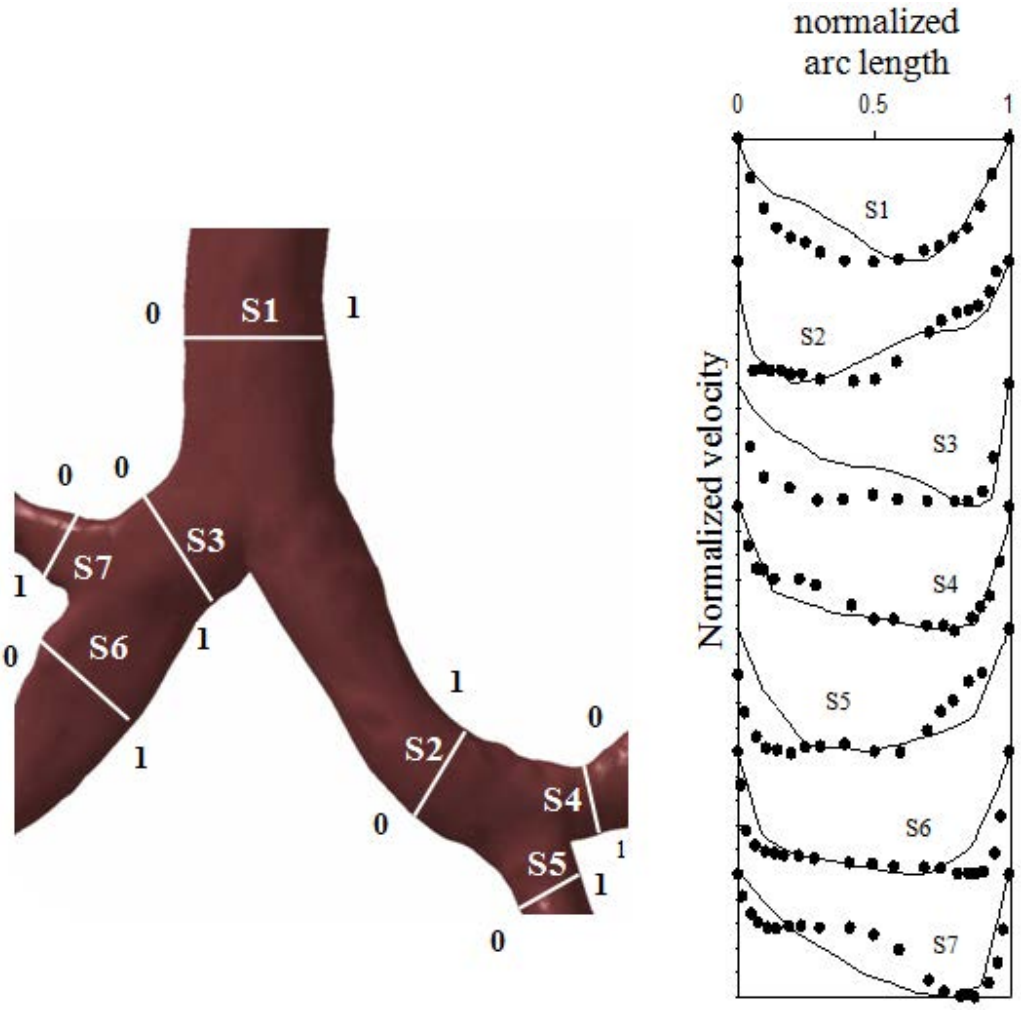


Figure 5.2 Normalized axial velocity profile and experimental data of Menon et al. (1984)

Normalized velocity profiles (Figure 5.2) in the trachea and main bronchi were taken and compared with experimental data from a 3:1 scaled up acrylic plastic model of the human central airways (Menon et al. 1984). Normalized axial velocity profile is plotted as a function of the normalized arc length. The experimental data of Menon et al. (1984) are plotted as (●) for the corresponding locations. The development of the profiles has been discussed in the publication by Menon et al. (1984) and for brevity is not reported here. However, local flow patterns are discussed in detail later in this report. The numerical results generally have good agreement with experimental results in terms of characteristic features. Some differences are found due to the inherent differences of the geometries used. For example at S1 the CFD results show a skewed velocity profile which is a result of the indentation from the aortic arch as well as the trachea cartilages. It can also observed that the tracheal bifurcation at the carina ridge produces a biased profile towards the inner walls of the subsequent daughter branches (S2, S3) because of the upstream flow momentum.

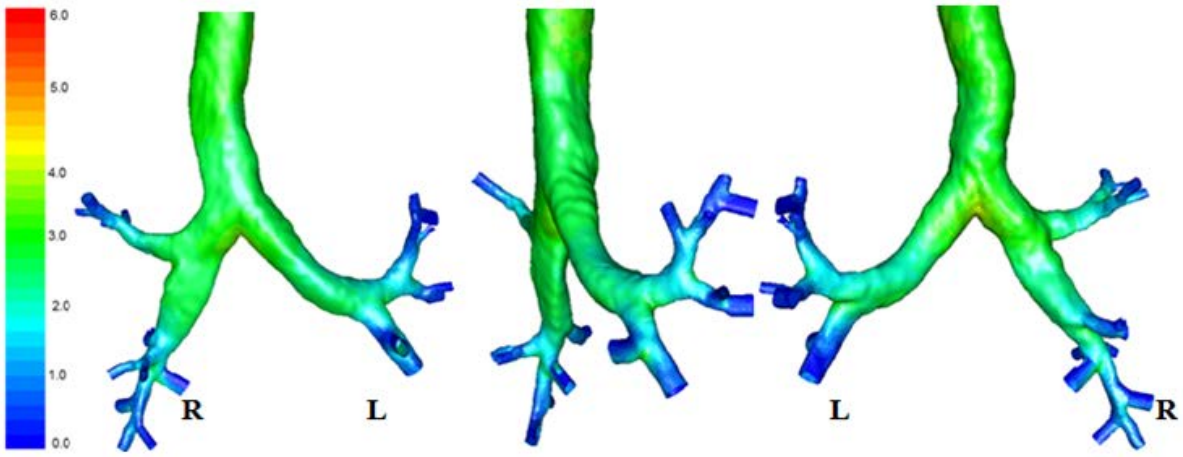
5.3.2 Pressure distribution

Contours of the total pressure in reference to the outflow pressure are shown in Figure 5.3. The required pressure difference at the inlet for the AA-model was 5.98 Pa, which is nearly twice the value for the recovered model (3.73 Pa). Along the main bronchus the pressure decreases steadily while there is a definite pressure drop from each main branch into the subsequent daughter branch. This is due to the bifurcation ridge where a local maximum occurs as a result of a build up of pressure, similar to a stagnation point.

5.3.3 Local flow features: main bronchus bifurcation

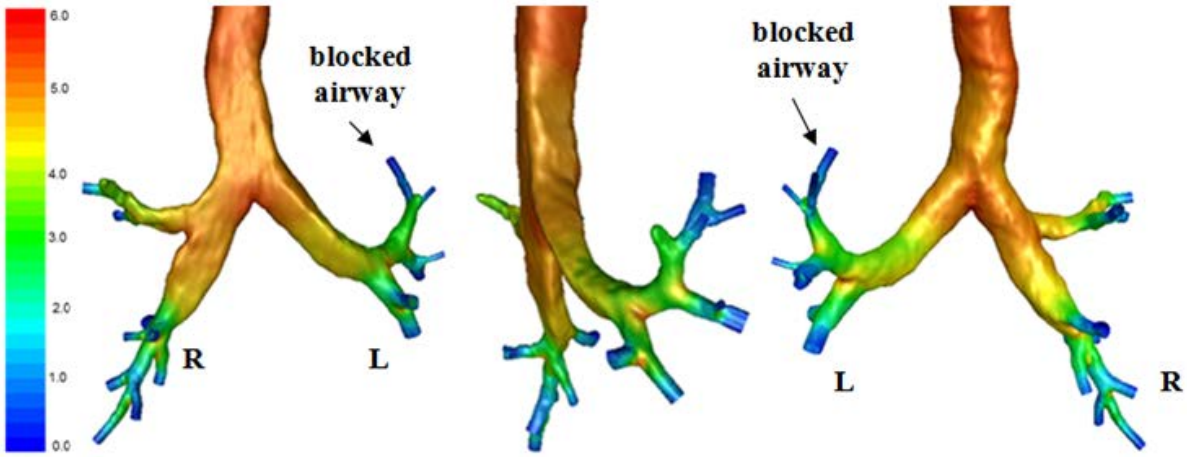
Crossflow streamlines overlayed on contours of axial velocity at cross-sections a-a' to f-f' for the REC-model are shown in Figure 5.4 and for the AA-model in Figure 5.5. The cross-sectional views are taken from the upstream flow point of view looking downwards in the axial direction. The three cross-sections preceding the bifurcation (a-a', b-b', and c-c') depict the flow before it bifurcates at the carina. Regionally the flow is concentrated on the left lateral wall due to the indentation by the transverse portion of the aortic arch. The streamlines are directed to the right side in response to the concentrated flow on the left wall. At b-b' the

streamlines are moving towards the ventral side and at cc' just before the bifurcation, the flow begins to divide into the left and right side with ventrally. In the AA-



Ventral Left Dorsal

REC-Model



Ventral Left Dorsal

Figure 5.3 Total pressure contour plots. The pressure value is the pressure difference between the local region with a reference pressure at the outlets.

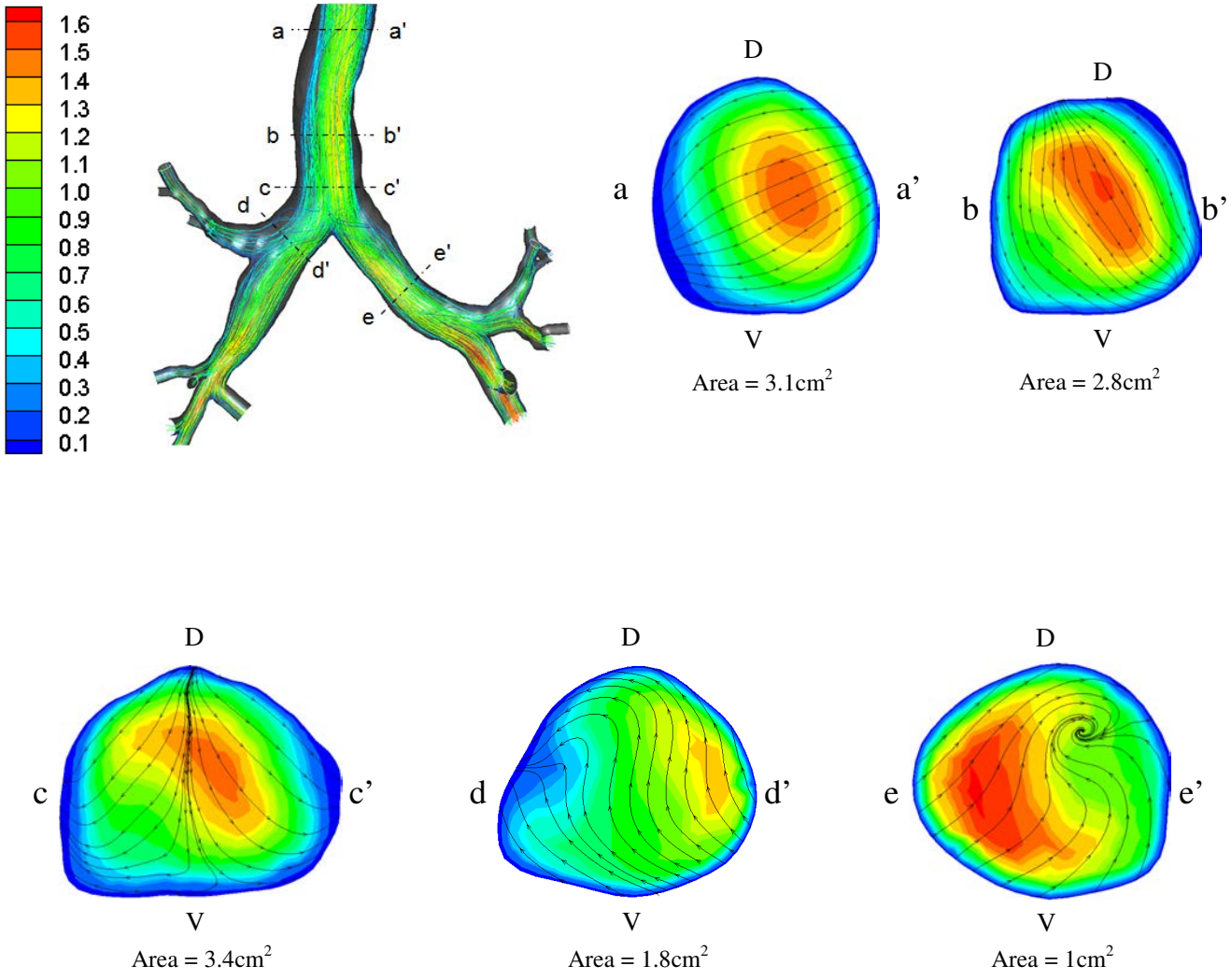


Figure 5.4 Cross-flow streamlines overlayed onto axial velocity contour plots at specified cross sections in the main bronchi of the REC-model.

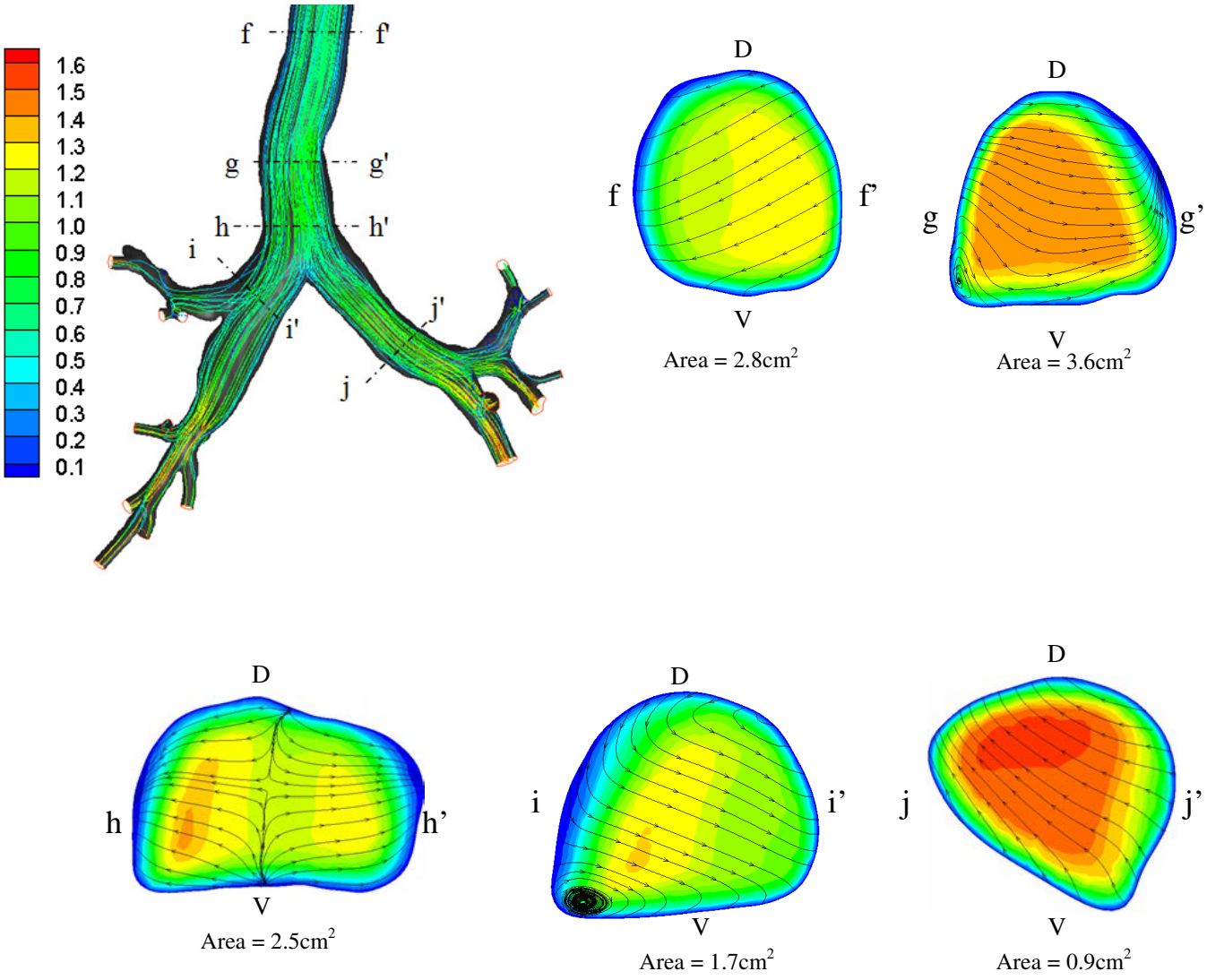


Figure 5.5 Cross-flow streamlines overlayed onto axial velocity contour plots at specified cross sections in the main bronchi of the AA-model.

model the axial contours are more evenly distributed. The effect of the aortic arch indentation is less significant, however after the indentation the streamlines are directed to the left side of the airway. Just prior to the bifurcation, the flow divides and the streamlines are moving more laterally than in the REC-model.

Following the flow through the right main bronchus (d-d'), of the REC-model the bulk flow remains close to the inner wall side with the flow momentum shifting towards the outer wall as characterised by the streamlines. In the left main bronchus at e-e', the flow is found on the inside wall and a region of recirculation is present in the low axial flow region because of: i) the upstream velocity profile that is biased to the left side, ii) a more obtuse bifurcation angle that curves laterally producing a centrifugal acceleration, and iii) the critical carina geometry. The bifurcation of the left main bronchus into the upper lobe bronchus (L3U) occurs laterally, extending to an almost horizontal position, while the lower lobe (L3L) is more directly in line with the main bronchus. This leads to a high distribution of slow passing through the L3L. The AA-model shows some differences in airflow patterns particularly at i-i' and j-j'. The streamlines in left main bronchus (j-j') suggest that the flow is moving from the ventral to the dorsal direction while locally the maximum axial velocity exists in the dorsal region. The main difference in the airway geometry is the presence of a narrowed rounded ridge on the ventral side. The geometry of the right main bronchus (i-i') has a flat ventral profile and high axial velocity is found more centrally in the cross-section. A region of recirculation appears in the left ventral region. It is interesting that recirculation regions are found within the first two branches after the carina for both models.

5.3.4 Local flow features: right upper lobe bifurcation

Local flow features were investigated at the right upper lobe bifurcation region which is near to one of the asthma affected regions. The occluded airway is present in the apical bronchus (4RM) which is downstream of cross-section p-p'. Because of the orientation of the cross-sections, the labelling of the orthogonal axis to the labels given in the figure are of the

superior and inferior sides with respect to the ventral and dorsal views given in Figure 5.3. The cross-sections k-k' and n-n' given in Figure 5.6 and Figure 5.7respectively are

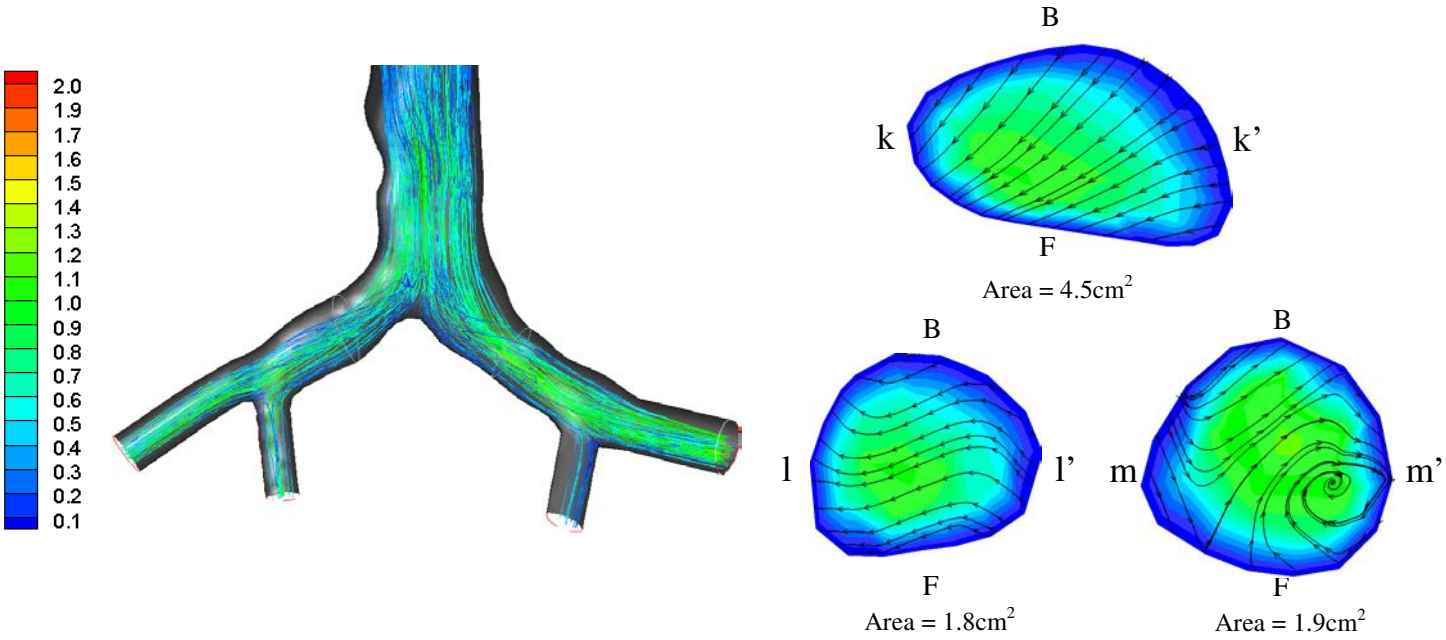


Figure 5.6 Cross-flow streamlines overlayed onto axial velocity contour plots at specified cross sections in the right upper lobe for the REC-model. Label S and I represent superior and inferior sides respectively.

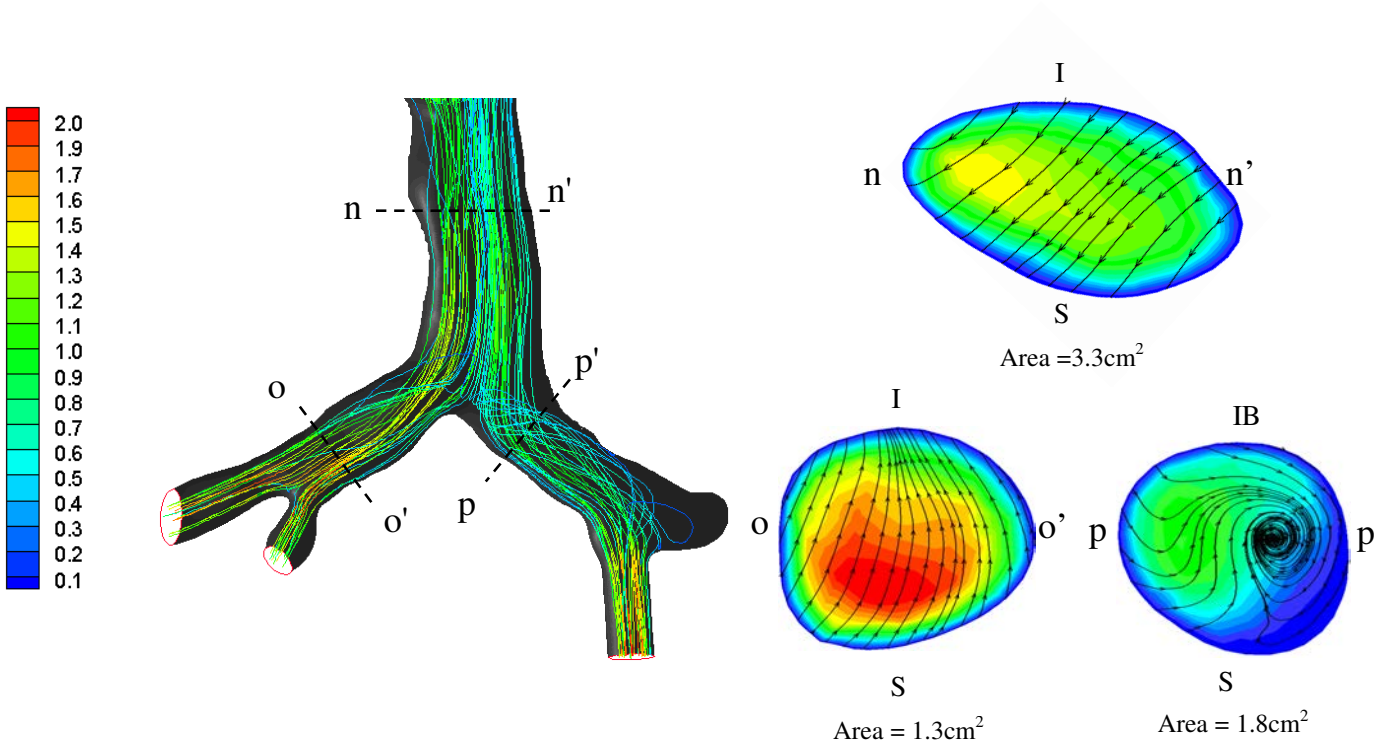


Figure 5.7 Cross-flow streamlines overlayed onto axial velocity contour plots at specified cross sections in the right upper lobe for the AA-model. Label S and I represent superior and inferior sides respectively.

located just after the right upper lobe (R3U) bifurcation from the main bronchus posterior-inferiorly which divides into the apical bronchus (R4M) and the posterior bronchus (R4P). As a result the streamlines are directed towards the superior side and the axial velocity is localised to one side. In both models a local recirculating region is found on the inner bifurcation side of the airway (m' in the REC-model and p' in the AA-model). This recirculation is caused by the flow separation as the airway bifurcates with the change in direction of the airway. The flow on the inner wall of the curvature and the corresponding axial velocities increase in comparison with the velocities on the outer wall. The flows in the posterior bronchi (l-l' and o-o') are considerably different to each other. The increase in axial velocity is produced by the narrowing of the airway coupled with its irregular shape. In the AA-model the streamlines are directed inferiorly while the axial flow is concentrated superiorly. This suggests that the airway is curving inferiorly and that drug particles for therapy which reach this region may impact on the superior side of the wall. The severe blockage in the right apical bronchus, distal to the cross-section p-p', shows some additional recirculation where the airway is blocked. Given this blockage, low flowrates in the bifurcating airway are present which reduces the likelihood of drug particles depositing onto the walls.

5.4 Discussion

Treatment of asthma is most commonly employed during the onset of the asthma episode. Particles are atomised into smaller particles through ventilators or other drug delivery devices and inhaled through the mouth. The flow profiles and features presented show that the changes in the airway caused by airway narrowing from asthma have a significant influence on flow patterns. This is especially true in the region where the airway narrowing is most severe. Drug particles are delivered with the target being the region of airway occlusion; however the airflow patterns may not be conducive to transporting the particles to the

targeted sites. For example smaller particles with less particle inertia will follow the path streamlines fairly well as if they were entrained in the flow. This is advantageous for deep lung particle penetration but also reduces the likelihood of particles depositing in regions of low flow i.e. airway occlusions. On the other hand, larger particles which exhibit higher particle inertia are more likely to experience early inertial impaction, thus reducing the particle trajectory length. Therefore a compromise or alternative solution is needed to provide effective drug delivery to the targeted site.

5.5 Conclusion

Two models with six generations of the airway tree from an acute asthma episode and following recovery from the same patient thirty days apart were reconstructed from computed tomography (CT) scans in order to investigate the effects of acute asthma on realistic airway geometry, the airflow patterns, the pressure drop, and the implications it has on targeted drug delivery. The comparisons in the geometry found that in general the right side of the airway is larger in diameter than the left side. The recovery of the airway was most significant in the severely asthma affected regions. Overall the right airway exhibited greater dilation in comparison with the left airway especially from the fifth generation onwards. In addition it was found that bifurcation angles do not vary significantly between the AA-model and the REC-model, however small changes are observed which may be due to the scans which were taken thirty days apart where the positioning and inhalation may have differed. The required pressure difference at the inlet for the AA-model was 5.98 Pa, which is nearly twice the value for the recovered model (3.73 Pa). This suggests that during the period of an acute asthma episode, the work of breathing for the patient in order to achieve the same tidal volumes, is double compared to the recovered state, which can lead to respiratory muscle fatigue. Local flow patterns showed that the changes in the airway had significant influence on flow patterns. This is especially apparent in the region where the airway narrowing is most severe. This means that studies of therapeutic drug delivery in the

airway should consider the effects of airway narrowing and not a recovered or a healthy airway.

Chapter 6
Conclusions and Recommendations

6.1 Concluding Remarks

The main objectives of this research were,

a. To develop a new algorithm for the reconstruction of tubular organs.

b. To develop a new algorithm to generate a human lung airway tree and an arterial vessel model.

c. To compare the difference of geometries and inhalation efforts between a bronchoconstricted and recovered airway tree associated with acute asthma.

d. To compare the difference of airflow patterns between a bronchoconstricted and recovered airway tree associated with acute asthma.

The work conducted in this research to satisfy these objectives has resulted in the following conclusions:

6.1.1 Conclusions on medical organs geometry modeling

1. Two CT/MRI image based geometry reconstruction approaches, the reverse engineering approach and the STL-triangulated model converting approach, are proposed.

2. The reverse engineering approach and the STL-triangulated model converting approach have been compared in human lung airway geometry model generation. The results indicate that it is difficult to triangulate the points obtained from medical image segmentation if the object geometry shape is complex.

3. The selection of the geometry reconstruction approach depends on the particular application it is intended for. The reverse engineering approach would be selected when the generated surface model or models have less overall complexity. The STL-triangulated model converting approach is preferred when a rapid prototype of the model is needed for surgical planning or display.

6.1.2 Conclusions on the region-based algorithm for human lung airway tree and arterial vessel model generation

1. A new region-based algorithm is proposed to reconstruct a geometry model of human airway and arterial vessels which consist of tubular geometries with a large difference in diameters. The method consists of two steps. The first step is to recognize the branches of the airway tree and bifurcations of the arterial vessels; the second step is to construct each of the branch surface patches by using an established method. On each branch the surface patches are constructed according to the number of available points by using an established method. In this way, different approximation accuracy can be applied in different branches depending on the availability of the branch information, which can lead to better surface quality.

2. A new searching algorithm has been developed to recognize the branches from the point clouds obtained from medical images. The searching principle is layer by layer. The topological relationship between branches and the relative location of each branch could be identified once the searching direction is fixed in each layer. Because the points on each branch have been grouped together, the new searching algorithm can also be utilized in the labeling of airway tresses.

6.1.3 Conclusions on geometry and inhalation effort comparison between the acute asthma and recovered airway trees

1. The recovery within the airways was measured by the increase in the diameters of the airways within the tracheobronchial tree, measured thirty days apart from the initial episode of asthma. The increase in diameters is also reflective of the increase in the

total lung capacity that is available as a larger diameter would provide a greater volume.

2. It was found that right sided airway tree had greater bronchodilation than the left airway tree following recovery after an acute asthma episode in this individual. This correlated with the fact that the right airway was larger in size.

3. Under steady laminar flow conditions of 15L/min, the required pressure difference at the inlet for the acute asthma model was 5.98 Pa, which is nearly twice the value for the recovered model (3.73 Pa). This requirement suggests that the respiratory muscles exert more effort during an acute asthma episode in order to achieve the same tidal volumes which can lead to respiratory muscle fatigue.

6.1.4 Conclusions on airflow patterns comparison between the acute asthma and recovered airway trees

1. The acute asthma model shows some differences in airflow patterns. The streamlines in the left main bronchus suggest that the flow is moving from the ventral to dorsal direction while locally the maximum axial velocity exists in the dorsal region. The main difference in the airway geometry is the presence of a narrowed rounded ridge on the ventral side. The geometry of the right main bronchus has a flat ventral profile and high axial velocity is found more centrally in the cross-section. A region of recirculation appears in the left ventral region. It is interesting that recirculation regions are found within the first two branches after the carina for both models.

2. The severe blockage in the right apical bronchus, distal to the cross-section p-p', shows some additional recirculation where the airway is blocked. Given this blockage, low flow rates in the bifurcating airway are present which reduces the likelihood of drug particles depositing onto the walls.

6.2 Recommendations for Further Study

Based on the results obtained during this investigation, the author suggests the following recommendations should be considered in future work.

The first major issue is to include the upper airway consisting of the oral/nasal cavity down to the larynx before joining the trachea and lower generations. It has been known that the larynx effect can significantly influence the entering flow and particle profile at the trachea and hence it can affect the deposition considerably in the first few bifurcations within the airways. If more computer resources (e.g. more powerful computers, enhanced image acquisition and processing software, etc) can be allocated, then lower generations (after generation 6) should also be included, because the entering profiles at the outlets were assumed to be flat when cyclic flow cases were applied at the current study. In practice, it has been known that the flow profile after bifurcation is hardly a flat profile. The significance of this issue in the final outcome is hence not incorporated in this study.

Secondly, the walls created within the model are assumed rigid and smooth whereas in reality the walls may exhibit some roughness and elasticity. The exclusion of these attributes may alter the predicted magnitudes of the pressure drop. In addition this exclusion may actually cause further errors as the elasticity and roughness of the walls are not accurately known. Fouke has developed a method to examine the elastic behavior of the upper airway (Fouke et al. 1989). Therefore the elastic behavior of airway wall should also be included in airway tree models in future studies.

The final recommendation is to compare and quantify the results and obtain more CT scan data of other patients from an acute asthma episode and following recovery. Some further study may advance particle deposition prediction and its dynamic nature in comparison between the acute asthma and recovered models.

Appendix

Code of layer-by-layer searching algorithm (used in Chapter 3)

****

Code of Root Bifurcation Boundary Searching

****

```
%This code is designed to find the root bifurcation boundary
%from triangulated points. The topology of the subject is tube
%like thing.

%read in the triangle from stl file and checking the point with
%the lowest Z coordinate

fid1=fopen('C:\Documents and
Settings\S3126701\Desktop\lung\RootPoints.txt','a');
fid=fopen('C:\Documents and Settings\S3126701\Desktop\lung\lung2.stl','r');
i=1;
LowestPoint{1}=[100 100 100];
line=fgetl(fid);
while feof(fid)==0
   line=fgetl(fid);
   if feof(fid)~=0
       break
   end
   line=fgetl(fid);


   line=fgetl(fid);
      t{1}=sscanf(line,'%*s %e %e %e',[1,inf]);
     if t{1}(1,3)<LowestPoint{1}(1,3)
          LowestPoint{1}=t{1};
     end

   line=fgetl(fid);
      t{2}=sscanf(line,'%*s %e %e %e',[1,inf]);
      if t{2}(1,3)<LowestPoint{1}(1,3)
          LowestPoint{1}=t{2};
      end

    line=fgetl(fid);
      t{3}=sscanf(line,'%*s %e %e %e',[1,inf]);
      if t{3}(1,3)<LowestPoint{1}(1,3)
          LowestPoint{1}=t{3};
      end

   line=fgetl(fid);

   line=fgetl(fid);
    T{i}=t;
```

```
  i=i+1;

end

%*********grouping the mesh*************
n31=1;
n32=1;
n33=1;
n34=1;
n35=1;
n36=1;
n37=1;
n38=1;
n39=1;
n40=1;
n41=1;
n42=1;
for j=1:(i-1)
    if T{j}{1,1}(1,3)<56.2
        gp1TriNo(n31)=j;
        n31=n31+1;
    else if T{j}{1,1}(1,3)<70.4
            gp2TriNo(n32)=j;
            n32=n32+1;
        else if T{j}{1,1}(1,3)<84.6
                gp3TriNo(n33)=j;
                n33=n33+1;
            else if T{j}{1,1}(1,3)<98.8
                    gp4TriNo(n34)=j;
                    n34=n34+1;
                else if T{j}{1,1}(1,3)<105.9
                        gp5TriNo(n35)=j;
                        n35=n35+1;
                    else if T{j}{1,1}(1,3)<113
                            gp6TriNo(n36)=j;
                            n36=n36+1;
                        else if T{j}{1,1}(1,3)<120.1
                                gp7TriNo(n37)=j;
                                n37=n37+1;
                            else if T{j}{1,1}(1,3)<127.2
                                    gp8TriNo(n38)=j;
                                    n38=n38+1;
                                else if T{j}{1,1}(1,3)<134.3
                                        gp9TriNo(n39)=j;
                                        n39=n39+1;
                                    else if T{j}{1,1}(1,3)<141.4
                                            gp10TriNo(n40)=j;
                                            n40=n40+1;
                                        else if T{j}{1,1}(1,3)<155.6
                                                gp11TriNo(n41)=j;
                                                n41=n41+1;
                                            else if T{j}{1,1}(1,3)<184
                                                    gp12TriNo(n42)=j;
```

```
                                                    n42=n42+1;
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
%**********end of grouping the mesh*************

UnvisitedTriNo=gp1TriNo;
VisitedTriNo=[];

%finding the triangles with LowestPoint from unvisited triangles
%m=l-1 is the number of associated triangle.
%No.AssTri is the arry to store the sequence number of associated triangle

l=1;
for k=1:(n31-1)
   if T{UnvisitedTriNo(k)}{1,1}==LowestPoint{1}
       AssociatedTri{l}=T{UnvisitedTriNo(k)};
       No.AssTri(l)=UnvisitedTriNo(k);
       l=l+1;
   else if T{UnvisitedTriNo(k)}{1,2}==LowestPoint{1}
            AssociatedTri{l}=T{UnvisitedTriNo(k)};
            No.AssTri(l)=UnvisitedTriNo(k);
            l=l+1;
       else if T{UnvisitedTriNo(k)}{1,3}==LowestPoint{1}
              AssociatedTri{l}=T{UnvisitedTriNo(k)};
              No.AssTri(l)=UnvisitedTriNo(k);
              l=l+1;
           end
       end
   end
end


%finding the NextPoint from AssociatedTri.
%n-1 is the number of NextPoint1,
%NextPoint1 are the points with lower z coordinate in each AssociatedTri.
%NextPoint is the point with bigger x coordinate in NextPoint1.

n=1;
for m=1:(l-1)
    if AssociatedTri{m}{1,1}(1,1)~=LowestPoint{1}(1,1)% x of lowest point
unequal
```

```
        if AssociatedTri{m}{1,2}(1,1)~=LowestPoint{1}(1,1)% x of second
point unequal
            if AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                NextPoint1{n}=AssociatedTri{m}{1,1};
            else NextPoint1{n}=AssociatedTri{m}{1,2};
            end
         else if AssociatedTri{m}{1,2}(1,2)~=LowestPoint{1}(1,2)% y of
second point unequal
                if AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                    NextPoint1{n}=AssociatedTri{m}{1,1};
                else NextPoint1{n}=AssociatedTri{m}{1,2};
                end
             else if AssociatedTri{m}{1,2}(1,3)~=LowestPoint{1}(1,3)% z of
second point unequal
                        if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                            NextPoint1{n}=AssociatedTri{m}{1,1};
                        else NextPoint1{n}=AssociatedTri{m}{1,2};
                        end
                    else if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,3}(1,3)%second point =
NextPoint
                            NextPoint1{n}=AssociatedTri{m}{1,1};
                        else NextPoint1{n}=AssociatedTri{m}{1,3};
                        end
                    end
             end
         end
      else if AssociatedTri{m}{1,1}(1,2)~=LowestPoint{1}(1,2)% y of first
point unequal
              if AssociatedTri{m}{1,2}(1,1)~=LowestPoint{1}(1,1)% x of
second point unequal
                if AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                    NextPoint1{n}=AssociatedTri{m}{1,1};
                else NextPoint1{n}=AssociatedTri{m}{1,2};
                end
              else if AssociatedTri{m}{1,2}(1,2)~=LowestPoint{1}(1,2)% y of
second point unequal
                        if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                            NextPoint1{n}=AssociatedTri{m}{1,1};
                        else NextPoint1{n}=AssociatedTri{m}{1,2};
                        end
                    else if AssociatedTri{m}{1,2}(1,3)~=LowestPoint{1}(1,3)%
z of second point unequal
                            if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                                NextPoint1{n}=AssociatedTri{m}{1,1};
                            else NextPoint1{n}=AssociatedTri{m}{1,2};
                            end
                        else if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,3}(1,3)%second point =
NextPoint
```

```
                                NextPoint1{n}=AssociatedTri{m}{1,1};
                            else NextPoint1{n}=AssociatedTri{m}{1,3};
                            end
                        end
                    end
                end
            else if AssociatedTri{m}{1,1}(1,3)~=LowestPoint{1}(1,3)% z of
first point unequal
                    if AssociatedTri{m}{1,2}(1,1)~=LowestPoint{1}(1,1)% x of
second point unequal
                        if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                            NextPoint1{n}=AssociatedTri{m}{1,1};
                        else NextPoint1{n}=AssociatedTri{m}{1,2};
                        end
                    else if AssociatedTri{m}{1,2}(1,2)~=LowestPoint{1}(1,2)%
y of second point unequal
                            if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                                 NextPoint1{n}=AssociatedTri{m}{1,1};
                            else NextPoint1{n}=AssociatedTri{m}{1,2};
                            end
                        else if
AssociatedTri{m}{1,2}(1,3)~=LowestPoint{1}(1,3)% z of second point unequal
                                if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,2}(1,3)
                                    NextPoint1{n}=AssociatedTri{m}{1,1};
                                else NextPoint1{n}=AssociatedTri{m}{1,2};
                                end
                            else if
AssociatedTri{m}{1,1}(1,3)<AssociatedTri{m}{1,3}(1,3)%second point =
NextPoint
                                    NextPoint1{n}=AssociatedTri{m}{1,1};
                                else NextPoint1{n}=AssociatedTri{m}{1,3};
                                end
                            end
                        end
                    end
                else if
AssociatedTri{m}{1,2}(1,3)<AssociatedTri{m}{1,3}(1,3)%first point =
NextPoint
                        NextPoint1{n}=AssociatedTri{m}{1,2};
                    else NextPoint1{n}=AssociatedTri{m}{1,3};
                    end
                end
            end
        end
        n=n+1;
end
n1=1;
NextPoint{1,1}=LowestPoint{1};
for n1=1:(n-2)
    if NextPoint1{n1}(1,1)>NextPoint1{n1+1}(1,1)
```

```
        NextPoint{1,2}=NextPoint1{n1};
        No.NextPoint=No.AssTri(n1);


    else NextPoint{1,2}=NextPoint1{n1+1};
         No.NextPoint=No.AssTri(n1+1);
    end
end
%
%****************************************
%finding the NextPoint starts from NextPoint.
%
%
 n7=3;
 r=n31-1;
 while NextPoint{1,(n7-1)}~=LowestPoint{1};%finishing the loop of finding
rootbundary
                                %when it reach start point(LowestPoint)

    VisitedTriNo=[VisitedTriNo,No.NextPoint];
    %r=size(UnvisitedTriNo1,2);
    %delet visited triangle.
    %n2=find(UnvisitedTriNo==No.NextPoint);
    %aa=[];
    %ab=[];
    %for n3=1:r
    %    if n3<n2
    %        aa(n3)=UnvisitedTriNo(n3);
    %    end
    %    if n3>n2
    %        ab(n3-n2)=UnvisitedTriNo(n3);
    %    end
    %end
    %UnvisitedTriNo=[aa,ab];
    UnvisitedTriNo(No.NextPoint)=0;
    %finding associated tri from unvisited tri
    %r=size(UnvisitedTriNo,2);
    i=1;
    No.AssTri=[];
    for n4=1:r
        if UnvisitedTriNo(n4)~=0
           if T{UnvisitedTriNo(n4)}{1,1}==NextPoint{(n7-1)}
               No.AssTri(i)=UnvisitedTriNo(n4);
               i=i+1;
           else if T{UnvisitedTriNo(n4)}{1,2}==NextPoint{(n7-1)}
                    No.AssTri(i)=UnvisitedTriNo(n4);
                    i=i+1;
               else if T{UnvisitedTriNo(n4)}{1,3}==NextPoint{(n7-1)}
                      No.AssTri(i)=UnvisitedTriNo(n4);
                      i=i+1;
                   end
               end
           end
```

```
        end
     end

    if i<3%in the case which has only one associated tri
        if T{No.AssTri(i-1)}{1,1}(1,1)~=NextPoint{(n7-1)}(1,1)% x of first
point unequal
            if T{No.AssTri(i-1)}{1,2}(1,1)~=NextPoint{(n7-1)}(1,1)% x of
second point unequal
                if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-1)}{1,2}(1,3)
                    NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                end
            else if T{No.AssTri(i-1)}{1,2}(1,2)~=NextPoint{(n7-1)}(1,2)% y
of second point unequal
                    if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                        NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                    else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                    end
                else if T{No.AssTri(i-1)}{1,2}(1,3)~=NextPoint{(n7-
1)}(1,3)% z of second point unequal
                        if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                            NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                        else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                        end
                    else if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,3}(1,3)%second point = NextPoint
                            NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                        else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,3};
                        end
                    end
                end
            end
        else if T{No.AssTri(i-1)}{1,1}(1,2)~=NextPoint{(n7-1)}(1,2)% y of
first point unequal
                if T{No.AssTri(i-1)}{1,2}(1,1)~=NextPoint{(n7-1)}(1,1)% x
of second point unequal
                    if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                        NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                    else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                    end
                else if T{No.AssTri(i-1)}{1,2}(1,2)~=NextPoint{(n7-
1)}(1,2)% y of second point unequal
                        if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                            NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                        else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                        end
                    else if T{No.AssTri(i-1)}{1,2}(1,3)~=NextPoint{(n7-
1)}(1,3)% z of second point unequal
```

```
                                if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                                    NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                                else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                                end
                            else if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,3}(1,3)%second point = NextPoint
                                    NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                                else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,3};
                                end
                            end
                        end
                    end
                else if T{No.AssTri(i-1)}{1,1}(1,3)~=NextPoint{(n7-1)}(1,3)% z
of first point unequal
                        if T{No.AssTri(i-1)}{1,2}(1,1)~=NextPoint{(n7-1)}(1,1)%
x of second point unequal
                            if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                                NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                            else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                            end
                        else if T{No.AssTri(i-1)}{1,2}(1,2)~=NextPoint{(n7-
1)}(1,2)% y of second point unequal
                                if T{No.AssTri(i-1)}{1,1}(1,3)<T{No.AssTri(i-
1)}{1,2}(1,3)
                                     NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                                else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                                end
                            else if T{No.AssTri(i-1)}{1,2}(1,3)~=NextPoint{(n7-
1)}(1,3)% z of second point unequal
                                    if T{No.AssTri(i-
1)}{1,1}(1,3)<T{No.AssTri(i-1)}{1,2}(1,3)
                                        NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                                    else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                                    end
                                else if T{No.AssTri(i-
1)}{1,1}(1,3)<T{No.AssTri(i-1)}{1,3}(1,3)%second point = NextPoint
                                        NextPoint{1,n7}=T{No.AssTri(i-1)}{1,1};
                                    else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,3};
                                    end
                                end
                            end
                        end
                    else if T{No.AssTri(i-1)}{1,2}(1,3)<T{No.AssTri(i-
1)}{1,3}(1,3)%first point = NextPoint
                            NextPoint{1,n7}=T{No.AssTri(i-1)}{1,2};
                        else NextPoint{1,n7}=T{No.AssTri(i-1)}{1,3};
                        end
                    end
                end
            end
            No.NextPoint=No.AssTri(i-1);
```

```
        n7=n7+1;
    else NextPoint1={};%clear the NextPoint1
      for n5=1:(i-1)%exclude the NextPoint{n7-1} from associated tri
          if T{No.AssTri(n5)}{1,1}(1,1)~=NextPoint{(n7-1)}(1,1)% x of first
point unequal
              if T{No.AssTri(n5)}{1,2}(1,1)~=NextPoint{(n7-1)}(1,1)% x of
second point unequal
                  NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                  NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};

              else if T{No.AssTri(n5)}{1,2}(1,2)~=NextPoint{(n7-1)}(1,2)% y
of second point unequal
                      NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                      NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};

                  else if T{No.AssTri(n5)}{1,2}(1,3)~=NextPoint{(n7-
1)}(1,3)% z of second point unequal
                          NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                          NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};

                      else NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
%second point = NextPoint
                          NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,3};

                      end
                   end
              end
          else if T{No.AssTri(n5)}{1,1}(1,2)~=NextPoint{(n7-1)}(1,2)% y of
first point unequal
                  if T{No.AssTri(n5)}{1,2}(1,1)~=NextPoint{(n7-1)}(1,1)% x
of second point unequal
                      NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                      NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};

                  else if T{No.AssTri(n5)}{1,2}(1,2)~=NextPoint{(n7-
1)}(1,2)% y of second point unequal
                          NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                          NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};

                      else if T{No.AssTri(n5)}{1,2}(1,3)~=NextPoint{(n7-
1)}(1,3)% z of second point unequal
                              NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                              NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};
                          else
NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};%second point = NextPoint
                               NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,3};

                          end
                      end
                  end
              else if T{No.AssTri(n5)}{1,1}(1,3)~=NextPoint{(n7-1)}(1,3)% z
of first point unequal
```

```
                    if T{No.AssTri(n5)}{1,2}(1,1)~=NextPoint{(n7-
1)}(1,1)% x of second point unequal
                        NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                        NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};
                    else if T{No.AssTri(n5)}{1,2}(1,2)~=NextPoint{(n7-
1)}(1,2)% y of second point unequal
                            NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                            NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};
                        else if
T{No.AssTri(n5)}{1,2}(1,3)~=NextPoint{(n7-1)}(1,3)% z of second point
unequal
                                NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};
                                NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,2};
                            else
NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,1};%second point = NextPoint

NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,3};

                            end
                        end
                    end
                else NextPoint1{n5}{1,1}=T{No.AssTri(n5)}{1,2};%first
point = NextPoint
                     NextPoint1{n5}{1,2}=T{No.AssTri(n5)}{1,3};
                end
            end
        end
    end
    %finding the nextpoint2 which doesn't equal to other points in
    %associated tri
    NextPoint2={};
    b={};
    for n10=1:(i-1)
        l1=1;
        l2=1;
        for n9=1:(i-1)
            if NextPoint1{n9}{1,1}(1,1)==NextPoint1{n10}{1,1}(1,1)%==
first point x
                if NextPoint1{n9}{1,1}(1,2)==NextPoint1{n10}{1,1}(1,2) %==
first point y
                    if NextPoint1{n9}{1,1}(1,3)==NextPoint1{n10}{1,1}(1,3)
%== first point z
                        b{n10}(1,1)=l1;
                        l1=l1+1;
                    end
                end
            end
            if NextPoint1{n9}{1,2}(1,1)==NextPoint1{n10}{1,1}(1,1)%==
second point x
                if NextPoint1{n9}{1,2}(1,2)==NextPoint1{n10}{1,1}(1,2) %==
second point y
                    if NextPoint1{n9}{1,2}(1,3)==NextPoint1{n10}{1,1}(1,3)
%== second point z
```

```
                        b{n10}(1,1)=l1;
                        l1=l1+1;
                    end
                end
            end
            if NextPoint1{n9}{1,1}(1,1)==NextPoint1{n10}{1,2}(1,1)%==
first point x
                if NextPoint1{n9}{1,1}(1,2)==NextPoint1{n10}{1,2}(1,2) %==
first point y
                    if NextPoint1{n9}{1,1}(1,3)==NextPoint1{n10}{1,2}(1,3)
%== first point z
                        b{n10}(1,2)=l2;
                        l2=l2+1;
                    end
                end
            end
            if NextPoint1{n9}{1,2}(1,1)==NextPoint1{n10}{1,2}(1,1)%==
first point x
                if NextPoint1{n9}{1,2}(1,2)==NextPoint1{n10}{1,2}(1,2) %==
first point y
                    if NextPoint1{n9}{1,2}(1,3)==NextPoint1{n10}{1,2}(1,3)
%== first point z
                        b{n10}(1,2)=l2;
                        l2=l2+1;
                    end
                end
            end
        end
    end
    n11=1;
    for n12=1:(i-1)
        if b{n12}(1,1)<2
            NextPoint2{1,n11}=NextPoint1{n12}{1,1};
            No.NextPoint2(n11)=No.AssTri(n12);
             n11=n11+1;
        end
        if b{n12}(1,2)<2
            NextPoint2{1,n11}=NextPoint1{n12}{1,2};
            No.NextPoint2(n11)=No.AssTri(n12);
            n11=n11+1;
        end
    end%after finish this step only two points left in this turn

    for n13=1:3% the next point is the point which is not in same tri
with last nextPoint
        if T{No.NextPoint}{1,n13}(1,1)==NextPoint2{1,1}(1,1)
            if T{No.NextPoint}{1,n13}(1,2)==NextPoint2{1,1}(1,2)
                if T{No.NextPoint}{1,n13}(1,3)==NextPoint2{1,1}(1,3)
                   NextPoint{1,n7}=NextPoint2{1,2};
                   No.NP=No.NextPoint2(2);
                   n7=n7+1;
                end
            end
```

```
        end
      end
      for n14=1:3
          if T{No.NextPoint}{1,n14}(1,1)==NextPoint2{1,2}(1,1)
             if T{No.NextPoint}{1,n14}(1,2)==NextPoint2{1,2}(1,2)
                 if T{No.NextPoint}{1,n14}(1,3)==NextPoint2{1,2}(1,3)
                    NextPoint{1,n7}=NextPoint2{1,1};
                    No.NP=No.NextPoint2(1);
                    n7=n7+1;
                 end
             end
          end
      end
      No.NextPoint=No.NP;
    end
 end


%****************************
%reorder the step 1 NextPoint
%****************************
%finding the biggest x,y and samallest x,y points
biggestXPoint{1,1}=NextPoint{1,1};
biggestYPoint{1,1}=NextPoint{1,1};
smallestXPoint{1,1}=NextPoint{1,1};
smallestYPoint{1,1}=NextPoint{1,1};

for n15=1:(n7-2)
    if NextPoint{1,n15}(1,1)>biggestXPoint{1,1}(1,1)
        biggestXPoint{1,1}=NextPoint{1,n15};
        No.biggestXPoint=n15;
    end
    if NextPoint{1,n15}(1,2)>biggestYPoint{1,1}(1,2)
        biggestYPoint{1,1}=NextPoint{1,n15};
        No.biggestYPoint=n15;
    end
    if NextPoint{1,n15}(1,1)<smallestXPoint{1,1}(1,1)
        smallestXPoint{1,1}=NextPoint{1,n15};
        No.smallestXPoint=n15;
    end
    if NextPoint{1,n15}(1,2)<smallestYPoint{1,1}(1,2)
        smallestYPoint{1,1}=NextPoint{1,n15};
        No.smallestYPoint=n15;
    end
end

%reorder the step 1 NextPoint
n16=1;
n17=No.biggestXPoint;
for n15=1:(n7-2)
    if n17<=(n7-2)
        NextPoint5{1,n15}=NextPoint{1,n17};
        n17=n17+1;
```

```
    else NextPoint5{1,n15}=NextPoint{1,n16};
        n16=n16+1;
    end
end

for n6=1:(n7-2)
    fprintf(fid1,'%f %f
%f\n',NextPoint5{1,n6}(1,1),NextPoint5{1,n6}(1,2),NextPoint5{1,n6}(1,3));
end




%************************************************************************
NextPoint={};
NextPoint=NextPoint5;

fclose(fid);
fclose(fid1);
```

****

Code of layer by layer searching

****

```
%This code is designed to find next step point start form the root
bifurcation boundary
%from unvisited triangule. The topology of the subject is tube
%like thing.

i1=1;

for m=1:1000

    n7=1;
    n=1;
    %NextPoint{m,n}={};
    i9=0; %birfucation indicator(case 1)refer to bif2step55
    i10=0;%birfucation indicator(case 2)refer to bif1step15
    i11=0;%birfucation indicator(case 3)refer to bif1.1.1.step14

    while n<=(j-1)

        %finding the triangles with NextPoint from unvisited triangles
        %No.AssTri is the arry to store the sequence number of associated
triangle
        if n==1
           l1=1;
           No.AssTri=[];
           No.AssTriAddress=[];
           for k=1:r
               if UnvisitedTriNo(k) ~= 0
                   if T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),n}
                      No.AssTri(l1)=UnvisitedTriNo(k);
                      No.AssTriAddress(l1)=k;
                      l1=l1+1;
                  else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n}
                            No.AssTri(l1)=UnvisitedTriNo(k);
                            No.AssTriAddress(l1)=k;
                            l1=l1+1;
                      else if T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n}
                                No.AssTri(l1)=UnvisitedTriNo(k);
                                No.AssTriAddress(l1)=k;
                                l1=l1+1;
                           end
                      end
                   end
               end
           end%for k=1:r
```

```
            %finding the triangles with last NextPoint from unvisited
triangles
            %No.LAssTri is the arry to store the sequence number of
associated
            %triangles
            l2=1;
            for k=1:r
                if UnvisitedTriNo(k)~=0
                    if T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),(j-1)}
                       No.LAssTri(l2)=UnvisitedTriNo(k);
                       l2=l2+1;
                   else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),(j-1)}
                            No.LAssTri(l2)=UnvisitedTriNo(k);
                            l2=l2+1;
                       else if T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-
1),(j-1)}
                                No.LAssTri(l2)=UnvisitedTriNo(k);
                                l2=l2+1;
                            end
                       end
                    end
                end
            end%for k=1:r

            %finding the same Tri between AssTri and LAssTri

            for n1=1:(l1-1)%if existing same Tri
                for n2=1:(l2-1)
                    if No.LAssTri(n2)==No.AssTri(n1)
                        a=n1;
                    end
                end
            end


            %finding the first NextStep point which neither equal to
nextpoint
            %nor equal to last nextpoint in same Tri
               for n2=1:3
                    if T{No.AssTri(a)}{1,n2}==NextPoint{(m-1),(j-1)}
                        b1(1,1)=n2;
                    end
                end
                for n2=1:3
                    if T{No.AssTri(a)}{1,n2}==NextPoint{(m-1),n}
                        b1(1,2)=n2;
                    end
                end
                if b1(1,1)==1
                    if b1(1,2)==2
```

```
                b1(1,3)=3;
            end
        end
        if b1(1,1)==2
            if b1(1,2)==1
                b1(1,3)=3;
            end
        end
        if b1(1,1)==1
            if b1(1,2)==3
                b1(1,3)=2;
            end
        end
        if b1(1,1)==3
            if b1(1,2)==1
                b1(1,3)=2;
            end
        end
        if b1(1,1)==2
            if b1(1,2)==3
                b1(1,3)=1;
            end
        end
        if b1(1,1)==3
            if b1(1,2)==2
                b1(1,3)=1;
            end
        end
        NextPoint{m,n7}=T{No.AssTri(a)}{1,b1(1,3)};
        c=1;%reset c=1;
        for n6=1:3%if the point is on last step, if so, clear this
point
            if NextPoint{m,n7}==NextPoint{m-1,n6}
                c=2;
                break;%stop the loop of for n6=1:(j-1)
            end
            if NextPoint{m,n7}==NextPoint{m-1,j-1-n6}
                c=3;
                break;%stop the loop of for n6=1:(j-1)
            end
        end

        %*********   if c==1   ***********************

        if c==1%unequal to last step
           l3=1;%find the AssTriNo with NextPoint{m,1}
           No.AssTri2=[];
           for k=1:r
               if UnvisitedTriNo(k)~=0
                  if T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,1}
                      No.AssTri2(l3)=UnvisitedTriNo(k);
                      l3=l3+1;
                  else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,1}
```

```
                                    No.AssTri2(l3)=UnvisitedTriNo(k);
                                    l3=l3+1;
                                else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,1}
                                        No.AssTri2(l3)=UnvisitedTriNo(k);
                                        l3=l3+1;
                                    end
                                end
                            end
                        end
                    end %for k=1:r %end find the AssTriNo with NextPoint{m,1}
                    UnvisitedAssTriNo=No.AssTri;
                    r1=size(UnvisitedAssTriNo,2);
                    VisitedAssTriNo=No.AssTri(a);
                    n2=find(UnvisitedAssTriNo==No.AssTri(a));
                    aa=[];
                    ab=[];
                    for n3=1:r1
                        if n3<n2
                            aa(n3)=UnvisitedAssTriNo(n3);
                        end
                        if n3>n2
                            ab(n3-n2)=UnvisitedAssTriNo(n3);
                        end
                    end
                    UnvisitedAssTriNo=[aa,ab];
                    r1=size(UnvisitedAssTriNo,2);
                    while r1>=2 %finding the Tri which include
NextPoint1{m,n7} from AssTri
                        for n4=1:r1
                            if T{UnvisitedAssTriNo(n4)}{1,1}==NextPoint{m,n7}
                                No.AssTri1(1)=UnvisitedAssTriNo(n4);
                            else if
T{UnvisitedAssTriNo(n4)}{1,2}==NextPoint{m,n7}
                                    No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                else if
T{UnvisitedAssTriNo(n4)}{1,3}==NextPoint{m,n7}
                                        No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                    end
                                end
                            end
                        end%for n4=1:r1
                %finding the first NextStep point which neither equal to
nextpoint
                %nor equal to last nextpoint in AssTri1
                        for n2=1:3
                            if T{No.AssTri1(1)}{1,n2}==NextPoint{m,n7}
                                b1(1,1)=n2;
                            end
                        end
                        for n2=1:3
                            if T{No.AssTri1(1)}{1,n2}==NextPoint{(m-1),n}
                                b1(1,2)=n2;
```

```
    end
end
if b1(1,1)==1
    if b1(1,2)==2
        b1(1,3)=3;
    end
end
if b1(1,1)==2
    if b1(1,2)==1
        b1(1,3)=3;
    end
end
if b1(1,1)==1
    if b1(1,2)==3
        b1(1,3)=2;
    end
end
if b1(1,1)==3
    if b1(1,2)==1
        b1(1,3)=2;
    end
end
if b1(1,1)==2
    if b1(1,2)==3
        b1(1,3)=1;
    end
end
if b1(1,1)==3
    if b1(1,2)==2
        b1(1,3)=1;
    end
end
n7=n7+1;
NextPoint{m,n7}=T{No.AssTri1(1)}{1,b1(1,3)};
VisitedAssTriNo=[VisitedAssTriNo,No.AssTri1(1)];
n2=find(UnvisitedAssTriNo==No.AssTri1(1));
aa=[];
ab=[];
for n3=1:r1
    if n3<n2
        aa(n3)=UnvisitedAssTriNo(n3);
    end
    if n3>n2
        ab(n3-n2)=UnvisitedAssTriNo(n3);
    end
end
UnvisitedAssTriNo=[aa,ab];%finding associated tri
from unvisited tri

r1=size(UnvisitedAssTriNo,2);
%No.NextPoint=No.AssTri1(1);
n5=0;%if the nextpoint equal to last step point,
clear this point
```

```
                    for n4=1:(j-1-n)%if the nextpoint equal to last step
point, clear this point
                        if NextPoint{m,n7}==NextPoint{(m-1),(n4+n)}
                            n5=1;
                        end
                    end
                    if n5==1
                        n7=n7-1;
                        break%stop the loop of while r1>=2
                    end
                end %end while r1>=2


                for i=1:(l1-1)%clear visitedTri
                    VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                    UnvisitedTriNo(No.AssTriAddress(i))=0;
                    i1=i1+1;
                end %end for i=1:(l1-1)
                n=n+1;

            end %c==1
            %**********end if c==1  ***************************

            %**********  if c==2   ***************************
            if c==2%on last step; refer to step37
                n7=1;
                c3=0;%c3=0 finding same tri from reverse direction, c3=1
need not finding from reverse direction

          %************ start finding same tri from normal direction **
                for n8=1:4
                    if c==2%on last step
                        for i=1:(l1-1)%clear visitedTri
                            VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                            UnvisitedTriNo(No.AssTriAddress(i))=0;
                            i1=i1+1;
                        end%end for i=1:(l1-1)

                    end
                    n=n+1;
                    d1=0;%d1=0 no same tri, d1=1 exist same tri
                    l1=1;
                    for k=1:r
                        if UnvisitedTriNo(k) ~= 0
                            if T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-
1),n}
                                if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),(j-1)}
                                    No.AssTri(l1)=UnvisitedTriNo(k);
                                    No.AssTriAddress(l1)=k;
                                    l1=l1+1;
                                    d1=1;
```

```
NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,3};
                                        break %for k=1:r
                                    else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),(j-1)}
                                            No.AssTri(l1)=UnvisitedTriNo(k);
                                            No.AssTriAddress(l1)=k;
                                            l1=l1+1;
                                            d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,2};
                                            break %for k=1:r
                                        end
                                    end
                                else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n}
                                        if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),(j-1)}
                                            No.AssTri(l1)=UnvisitedTriNo(k);
                                            No.AssTriAddress(l1)=k;
                                            l1=l1+1;
                                            d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,3};
                                            break %for k=1:r
                                        else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),(j-1)}

No.AssTri(l1)=UnvisitedTriNo(k);
                                                No.AssTriAddress(l1)=k;
                                                l1=l1+1;
                                                d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,1};
                                                break %for k=1:r
                                            end
                                        end
                                    else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n}
                                            if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),(j-1)}

No.AssTri(l1)=UnvisitedTriNo(k);
                                                No.AssTriAddress(l1)=k;
                                                l1=l1+1;
                                                d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,2};
                                                break %for k=1:r
                                            else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),(j-1)}

No.AssTri(l1)=UnvisitedTriNo(k);
```

```
No.AssTriAddress(l1)=k;
                                                    l1=l1+1;
                                                    d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,1};
                                                break %for k=1:r
                                                end
                                            end
                                        end
                                    end
                                end
                            end %if UnvisitedTriNo(k) ~= 0
                        end%for k=1:r
                        if d1==1%exist same Tri
                            c=1;
                            for n6=1:3%if the point is on last step, if so,
clear this point
                                if NextPoint{m,n7}==NextPoint{m-1,n6}
                                    c=2;%on last step
                                    n7=1;
                                    break;%stop the loop of for n6=1:(j-1)
                                end
                                if NextPoint{m,n7}==NextPoint{m-1,j-1-n6}
                                    c=2;
                                    n7=1;
                                    break;%stop the loop of for n6=1:(j-1)
                                end
                            end
                        end
                        if c==1 %doesn't on last step
                            for i=1:(l1-1)%clear visitedTri
                               VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                               UnvisitedTriNo(No.AssTriAddress(i))=0;
                               i1=i1+1;
                            end%end for i=1:(l1-1)
                            l1=1;
                            for k=1:r
                                if UnvisitedTriNo(k) ~= 0
                                    if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),n}
                                        No.AssTri(l1)=UnvisitedTriNo(k);
                                        No.AssTriAddress(l1)=k;
                                        l1=l1+1;
                                    else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n}

No.AssTri(l1)=UnvisitedTriNo(k);
                                            No.AssTriAddress(l1)=k;
                                            l1=l1+1;
                                     else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n}
```

```
No.AssTri(l1)=UnvisitedTriNo(k);
                                                    No.AssTriAddress(l1)=k;
                                                    l1=l1+1;
                                                end
                                            end
                                        end
                                    end
                                end%for k=1:r
                                l3=1;%find the AssTriNo with NextPoint{m,1}
                                No.AssTri2=[];
                                for k=1:r
                                    if UnvisitedTriNo(k)~=0
                                        if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,1}
                                            No.AssTri2(l3)=UnvisitedTriNo(k);
                                            l3=l3+1;
                                        else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,1}

No.AssTri2(l3)=UnvisitedTriNo(k);
                                                l3=l3+1;
                                            else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,1}

No.AssTri2(l3)=UnvisitedTriNo(k);
                                                    l3=l3+1;
                                                end
                                            end
                                        end
                                    end
                                end %for k=1:r %end find the AssTriNo with
NextPoint{m,1}
                                UnvisitedAssTriNo=No.AssTri;
                                r1=size(UnvisitedAssTriNo,2);

                                while r1>=2 %finding the Tri which include
NextPoint1{m,n7} from AssTri
                                    for n4=1:r1
                                        if
T{UnvisitedAssTriNo(n4)}{1,1}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                            else if
T{UnvisitedAssTriNo(n4)}{1,2}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                                else if
T{UnvisitedAssTriNo(n4)}{1,3}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                                    end
                                                end
```

```
                                    end
                                end%for n4=1:r1
                                %finding the first NextStep point which
neither equal to nextpoint
                                %nor equal to last nextpoint in AssTri1
                                for n2=1:3
                                    if
T{No.AssTri1(1)}{1,n2}==NextPoint{m,n7}
                                        b1(1,1)=n2;
                                    end
                                end
                                for n2=1:3
                                    if
T{No.AssTri1(1)}{1,n2}==NextPoint{(m-1),n}
                                        b1(1,2)=n2;
                                    end
                                end
                                if b1(1,1)==1
                                    if b1(1,2)==2
                                        b1(1,3)=3;
                                    end
                                end
                                if b1(1,1)==2
                                    if b1(1,2)==1
                                        b1(1,3)=3;
                                    end
                                end
                                if b1(1,1)==1
                                    if b1(1,2)==3
                                        b1(1,3)=2;
                                    end
                                end
                                if b1(1,1)==3
                                    if b1(1,2)==1
                                        b1(1,3)=2;
                                    end
                                end
                                if b1(1,1)==2
                                    if b1(1,2)==3
                                        b1(1,3)=1;
                                    end
                                end
                                if b1(1,1)==3
                                    if b1(1,2)==2
                                        b1(1,3)=1;
                                    end
                                end
                               n7=n7+1;

NextPoint{m,n7}=T{No.AssTri1(1)}{1,b1(1,3)};

VisitedAssTriNo=[VisitedAssTriNo,No.AssTri1(1)];
                               n2=find(UnvisitedAssTriNo==No.AssTri1(1));
```

```
                                aa=[];
                                ab=[];
                                for n3=1:r1
                                    if n3<n2
                                        aa(n3)=UnvisitedAssTriNo(n3);
                                    end
                                    if n3>n2
                                        ab(n3-n2)=UnvisitedAssTriNo(n3);
                                    end
                                end
                                UnvisitedAssTriNo=[aa,ab];%finding
associated tri from unvisited tri
                                r1=size(UnvisitedAssTriNo,2);
                                %No.NextPoint=No.AssTri1(1);
                                n5=0;%if the nextpoint equal to last step
point, clear this point
                                for n4=1:(j-1-n)%if the nextpoint equal to
last step point, clear this point
                                    if NextPoint{m,n7}==NextPoint{(m-
1),(n4+n)}
                                        n5=1;
                                    end
                                end
                                if n5==1
                                    n7=n7-1;
                                break%stop the loop of while r1>=2
                                end
                            end %end while r1>=2
                            for i=1:(l1-1)%clear visitedTri
                                VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                                UnvisitedTriNo(No.AssTriAddress(i))=0;
                                i1=i1+1;
                            end %end for i=1:(l1-1)
                            n=n+1;
                            c3=1;
                            break;%stop the loop of for n6=1:4
                        end %c==1
                    end%for n8=1:4
            %*********  end finding tri from normal direction   *****


            %********* start finding tri from reverse direction *******
                    if c3==0%find same tri from reverse direction
                        for n28=1:5
                            if c==2%on last step
                                for i=1:(l1-1)%clear visitedTri
                                    VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                                    UnvisitedTriNo(No.AssTriAddress(i))=0;
                                    i1=i1+1;
                                end%end for i=1:(l1-1)
                            end
                            d1=0;%d1=0 no same tri, d1=1 exist same tri
                            l1=1;
```

```
                            for k=1:r
                                if UnvisitedTriNo(k) ~= 0
                                    if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),(j-2)}
                                        if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n28}
                                            No.AssTri(l1)=UnvisitedTriNo(k);
                                            No.AssTriAddress(l1)=k;
                                            l1=l1+1;
                                            d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,3};
                                            break %for k=1:r
                                        else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n28}

No.AssTri(l1)=UnvisitedTriNo(k);
                                                No.AssTriAddress(l1)=k;
                                                l1=l1+1;
                                                d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,2};
                                                break %for k=1:r
                                            end
                                        end
                                    else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),(j-2)}
                                            if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),n28}

No.AssTri(l1)=UnvisitedTriNo(k);
                                                No.AssTriAddress(l1)=k;
                                                l1=l1+1;
                                                d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,3};
                                                break %for k=1:r
                                            else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n28}

No.AssTri(l1)=UnvisitedTriNo(k);
                                                    No.AssTriAddress(l1)=k;
                                                    l1=l1+1;
                                                    d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,1};
                                                    break %for k=1:r
                                                end
                                            end
                                        else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),(j-2)}
                                                if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),n28}
```

```
No.AssTri(l1)=UnvisitedTriNo(k);
                                                    No.AssTriAddress(l1)=k;
                                                    l1=l1+1;
                                                    d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,2};
                                                    break %for k=1:r
                                                else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n28}

No.AssTri(l1)=UnvisitedTriNo(k);

No.AssTriAddress(l1)=k;
                                                         l1=l1+1;
                                                         d1=1;

NextPoint{m,n7}=T{UnvisitedTriNo(k)}{1,1};
                                                    break %for k=1:r
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end %if UnvisitedTriNo(k) ~= 0
                            end%for k=1:r
                            if d1==1%exist same Tri
                                c=1;
                                for n6=1:3%if the point is on last step, if
so, clear this point
                                    if NextPoint{m,n7}==NextPoint{m-1,n6}
                                        c=2;%on last step
                                        n7=1;
                                        break;%stop the loop of for
n28=1:(j-1)
                                    end
                                    if NextPoint{m,n7}==NextPoint{m-1,j-1-
n6}
                                        c=2;
                                        n7=1;
                                        break;%stop the loop of for
n28=1:(j-1)
                                    end
                                end
                            end
                            if c==1 %doesn't on last step
                                for i=1:(l1-1)%clear visitedTri

VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                                    UnvisitedTriNo(No.AssTriAddress(i))=0;
                                    i1=i1+1;
                                end%end for i=1:(l1-1)
                                l1=1;
```

```
                                j=j-1;
                                n=n28;
                                for k=1:r
                                    if UnvisitedTriNo(k) ~= 0
                                        if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),n}
                                            No.AssTri(l1)=UnvisitedTriNo(k);
                                            No.AssTriAddress(l1)=k;
                                            l1=l1+1;
                                        else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n}

No.AssTri(l1)=UnvisitedTriNo(k);
                                                 No.AssTriAddress(l1)=k;
                                                 l1=l1+1;
                                             else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n}

No.AssTri(l1)=UnvisitedTriNo(k);

No.AssTriAddress(l1)=k;
                                                      l1=l1+1;
                                                 end
                                            end
                                        end
                                    end
                                end%for k=1:r
                                l3=1;%find the AssTriNo with NextPoint{m,1}
                                No.AssTri2=[];
                                for k=1:r
                                    if UnvisitedTriNo(k)~=0
                                        if
T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,1}

No.AssTri2(l3)=UnvisitedTriNo(k);
                                            l3=l3+1;
                                        else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,1}

No.AssTri2(l3)=UnvisitedTriNo(k);
                                                 l3=l3+1;
                                            else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,1}

No.AssTri2(l3)=UnvisitedTriNo(k);
                                                      l3=l3+1;
                                                 end
                                            end
                                        end
                                    end
                                end %for k=1:r %end find the AssTriNo with
NextPoint{m,1}
                                UnvisitedAssTriNo=No.AssTri;
```

```
                                r1=size(UnvisitedAssTriNo,2);

                                while r1>=2 %finding the Tri which include
NextPoint1{m,n7} from AssTri
                                    for n4=1:r1
                                        if
T{UnvisitedAssTriNo(n4)}{1,1}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                        else if
T{UnvisitedAssTriNo(n4)}{1,2}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                            else if
T{UnvisitedAssTriNo(n4)}{1,3}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                                end
                                            end
                                        end
                                    end%for n4=1:r1
                                    %finding the first NextStep point
which neither equal to nextpoint
                                    %nor equal to last nextpoint in
AssTri1
                                    for n2=1:3
                                        if
T{No.AssTri1(1)}{1,n2}==NextPoint{m,n7}
                                            b1(1,1)=n2;
                                        end
                                    end
                                    for n2=1:3
                                        if
T{No.AssTri1(1)}{1,n2}==NextPoint{(m-1),n}
                                            b1(1,2)=n2;
                                        end
                                    end
                                    if b1(1,1)==1
                                        if b1(1,2)==2
                                            b1(1,3)=3;
                                        end
                                    end
                                    if b1(1,1)==2
                                        if b1(1,2)==1
                                            b1(1,3)=3;
                                        end
                                    end
                                    if b1(1,1)==1
                                        if b1(1,2)==3
                                            b1(1,3)=2;
                                        end
                                    end
                                    if b1(1,1)==3
```

```
                                        if b1(1,2)==1
                                            b1(1,3)=2;
                                        end
                                    end
                                    if b1(1,1)==2
                                        if b1(1,2)==3
                                            b1(1,3)=1;
                                        end
                                    end
                                    if b1(1,1)==3
                                        if b1(1,2)==2
                                            b1(1,3)=1;
                                        end
                                    end
                                   n7=n7+1;

NextPoint{m,n7}=T{No.AssTri1(1)}{1,b1(1,3)};

VisitedAssTriNo=[VisitedAssTriNo,No.AssTri1(1)];

n2=find(UnvisitedAssTriNo==No.AssTri1(1));
                                   aa=[];
                                   ab=[];
                                   for n3=1:r1
                                       if n3<n2
                                           aa(n3)=UnvisitedAssTriNo(n3);
                                       end
                                       if n3>n2
                                           ab(n3-
n2)=UnvisitedAssTriNo(n3);
                                       end
                                   end
                                   UnvisitedAssTriNo=[aa,ab];%finding
associated tri from unvisited tri
                                   r1=size(UnvisitedAssTriNo,2);
                                   %No.NextPoint=No.AssTri1(1);
                                   n5=0;%if the nextpoint equal to last
step point, clear this point
                                   for n4=1:(j-1-n)%if the nextpoint
equal to last step point, clear this point
                                       if NextPoint{m,n7}==NextPoint{(m-
1),(n4+n)}

                                           n5=1;
                                       end
                                   end
                                   if n5==1
                                       n7=n7-1;
                                   break%stop the loop of while r1>=2
                                   end
                               end %end while r1>=2
                               for i=1:(l1-1)%clear visitedTri

VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
```

```
                                    UnvisitedTriNo(No.AssTriAddress(i))=0;
                                    i1=i1+1;
                                end %end for i=1:(l1-1)
                                n=n+1;
                                c3=1;
                                break;%stop the loop of for n6=1:5
                            end %c==1
                        end%for n28=1:5
                    end%if c3==0
                  %*********  end finding tri form reverse direction *****
                else if c==3 %refer to step89
                        l2=1;
                        for k=1:r
                            if UnvisitedTriNo(k)~=0
                                if T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-
1),(j-2)}
                                   No.LAssTri(l2)=UnvisitedTriNo(k);
                                   l2=l2+1;
                                else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),(j-2)}
                                       No.LAssTri(l2)=UnvisitedTriNo(k);
                                       l2=l2+1;
                                    else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),(j-2)}
                                         No.LAssTri(l2)=UnvisitedTriNo(k);
                                         l2=l2+1;
                                       end
                                    end
                                end
                            end
                        end%for k=1:r
                        for n1=1:(l1-1)%if existing same Tri
                            if n1~=a
                               for n2=1:(l2-1)
                                   if No.LAssTri(n2)==No.AssTri(n1)
                                      a1=n1;
                                   end
                               end
                            end
                        end
                        for n2=1:3
                             if T{No.AssTri(a1)}{1,n2}==NextPoint{(m-1),(j-
1)}
                                 b1(1,1)=n2;
                             end
                        end
                        for n2=1:3
                            if T{No.AssTri(a1)}{1,n2}==NextPoint{(m-1),n}
                                b1(1,2)=n2;
                            end
                        end
                        if b1(1,1)==1
                            if b1(1,2)==2
```

```
                        b1(1,3)=3;
                    end
                end
                if b1(1,1)==2
                    if b1(1,2)==1
                        b1(1,3)=3;
                    end
                end
                if b1(1,1)==1
                    if b1(1,2)==3
                        b1(1,3)=2;
                    end
                end
                if b1(1,1)==3
                    if b1(1,2)==1
                        b1(1,3)=2;
                    end
                end
                if b1(1,1)==2
                    if b1(1,2)==3
                        b1(1,3)=1;
                    end
                end
                if b1(1,1)==3
                    if b1(1,2)==2
                        b1(1,3)=1;
                    end
                end
                NextPoint{m,n7}=T{No.AssTri(a1)}{1,b1(1,3)};
                l3=1;%find the AssTriNo with NextPoint{m,1}
                No.AssTri2=[];
                for k=1:r
                    if UnvisitedTriNo(k)~=0
                        if T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,1}
                            No.AssTri2(l3)=UnvisitedTriNo(k);
                            l3=l3+1;
                        else if
T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,1}
                                No.AssTri2(l3)=UnvisitedTriNo(k);
                                l3=l3+1;
                            else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,1}

No.AssTri2(l3)=UnvisitedTriNo(k);
                                    l3=l3+1;
                                end
                            end
                        end
                    end
                end %for k=1:r %end find the AssTriNo with
NextPoint{m,1}
                UnvisitedAssTriNo=No.AssTri;
                r1=size(UnvisitedAssTriNo,2);
```

```
                        VisitedAssTriNo=No.AssTri(a);
                        n2=find(UnvisitedAssTriNo==No.AssTri(a));
                        aa=[];
                        ab=[];
                        for n3=1:r1
                            if n3<n2
                                aa(n3)=UnvisitedAssTriNo(n3);
                            end
                            if n3>n2
                                ab(n3-n2)=UnvisitedAssTriNo(n3);
                            end
                        end
                        UnvisitedAssTriNo=[aa,ab];
                        r1=size(UnvisitedAssTriNo,2);
                        while r1>=3 %finding the Tri which include
NextPoint1{m,n7} from AssTri
                            for n4=1:r1
                                if
T{UnvisitedAssTriNo(n4)}{1,1}==NextPoint{m,n7}
                                    No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                else if
T{UnvisitedAssTriNo(n4)}{1,2}==NextPoint{m,n7}
                                        No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                    else if
T{UnvisitedAssTriNo(n4)}{1,3}==NextPoint{m,n7}

No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                        end
                                    end
                                end
                            end%for n4=1:r1
                %finding the first NextStep point which neither equal to
nextpoint
                %nor equal to last nextpoint in AssTri1
                            for n2=1:3
                                if T{No.AssTri1(1)}{1,n2}==NextPoint{m,n7}
                                    b1(1,1)=n2;
                                end
                            end
                            for n2=1:3
                                if T{No.AssTri1(1)}{1,n2}==NextPoint{(m-
1),n}
                                    b1(1,2)=n2;
                                end
                            end
                            if b1(1,1)==1
                                if b1(1,2)==2
                                    b1(1,3)=3;
                                end
                            end
                            if b1(1,1)==2
                                if b1(1,2)==1
                                    b1(1,3)=3;
```

```
                    end
                end
                if b1(1,1)==1
                    if b1(1,2)==3
                        b1(1,3)=2;
                    end
                end
                if b1(1,1)==3
                    if b1(1,2)==1
                        b1(1,3)=2;
                    end
                end
                if b1(1,1)==2
                    if b1(1,2)==3
                        b1(1,3)=1;
                    end
                end
                if b1(1,1)==3
                    if b1(1,2)==2
                        b1(1,3)=1;
                    end
                end
                n7=n7+1;
                NextPoint{m,n7}=T{No.AssTri1(1)}{1,b1(1,3)};
                VisitedAssTriNo=[VisitedAssTriNo,No.AssTri1(1)];
                n2=find(UnvisitedAssTriNo==No.AssTri1(1));
                aa=[];
                ab=[];
                for n3=1:r1
                    if n3<n2
                        aa(n3)=UnvisitedAssTriNo(n3);
                    end
                    if n3>n2
                        ab(n3-n2)=UnvisitedAssTriNo(n3);
                    end
                end
                UnvisitedAssTriNo=[aa,ab];%finding associated
tri from unvisited tri
                r1=size(UnvisitedAssTriNo,2);
                %No.NextPoint=No.AssTri1(1);
                n5=0;%if the nextpoint equal to last step point,
clear this point
                for n4=1:(j-1-n)%if the nextpoint equal to last
step point, clear this point
                    if NextPoint{m,n7}==NextPoint{(m-1),(n4+n)}
                        n5=1;
                    end
                end
                if n5==1
                    n7=n7-1;
                    break%stop the loop of while r1>=2
                end
            end %end while r1>=2
```

```
                    for i=1:(l1-1)%clear visitedTri
                        VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                        UnvisitedTriNo(No.AssTriAddress(i))=0;
                        i1=i1+1;
                    end %end for i=1:(l1-1)
                    n=n+1;
                    j=j-1;
                end%else if c==3

            end %if c==2
        end %if n=1

        if n>1
           l1=1;
           No.AssTri=[];
           for k=1:r
               if UnvisitedTriNo(k)~=0
                  if T{UnvisitedTriNo(k)}{1,1}==NextPoint{(m-1),n}
                      No.AssTri(l1)=UnvisitedTriNo(k);
                      No.AssTriAddress(l1)=k;
                      l1=l1+1;
                  else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{(m-1),n}
                           No.AssTri(l1)=UnvisitedTriNo(k);
                           No.AssTriAddress(l1)=k;
                           l1=l1+1;
                      else if T{UnvisitedTriNo(k)}{1,3}==NextPoint{(m-1),n}
                               No.AssTri(l1)=UnvisitedTriNo(k);
                               No.AssTriAddress(l1)=k;
                               l1=l1+1;
                          end
                      end
                  end
               end
            end %for k=1:r
            if l1==2
                VisitedTriNo=[VisitedTriNo,No.AssTri(1)];
                UnvisitedTriNo(No.AssTriAddress(1))=0;
                i1=i1+1;
            end%if l1==2
            if l1>2
                VisitedAssTriNo=[];
                UnvisitedAssTriNo=No.AssTri;
                r1=size(UnvisitedAssTriNo,2);
                while r1>=2 %finding the Tri which include NextPoint1{n7}
from AssTri
                    for n4=1:r1
                        if T{UnvisitedAssTriNo(n4)}{1,1}==NextPoint{m,n7}
                            No.AssTri1(1)=UnvisitedAssTriNo(n4);
                            i8=n;
```

```
                        else if
T{UnvisitedAssTriNo(n4)}{1,2}==NextPoint{m,n7}
                                No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                i8=n;

                            else if
T{UnvisitedAssTriNo(n4)}{1,3}==NextPoint{m,n7}
                                    No.AssTri1(1)=UnvisitedAssTriNo(n4);
                                    i8=n;

                                end
                            end
                        end
                    end
                    if (n-i8)>8
                        i7=i8;
                        i9=1;
                    end
                    if i9==1
                        if i8==n
                            i10=1;
                            fprintf('There is a birfucation on step
%d\n',m-1)
                            break%while r1>=2
                        end
                    end


                    %finding the point which neither equal to nextpoint
                    %nor equal to last step nextpoint in AssTri1
                    for n2=1:3
                        if T{No.AssTri1(1)}{1,n2}==NextPoint{m,n7}
                            b1(1,1)=n2;
                        end
                    end
                    for n2=1:3
                        if T{No.AssTri1(1)}{1,n2}==NextPoint{(m-1),n}
                            b1(1,2)=n2;
                        end
                    end
                    if b1(1,1)==1
                        if b1(1,2)==2
                            b1(1,3)=3;
                        end
                    end
                    if b1(1,1)==2
                        if b1(1,2)==1
                            b1(1,3)=3;
                        end
                    end
                    if b1(1,1)==1
                        if b1(1,2)==3
                            b1(1,3)=2;
```

```
                    end
                end
                if b1(1,1)==3
                    if b1(1,2)==1
                        b1(1,3)=2;
                    end
                end
                if b1(1,1)==2
                    if b1(1,2)==3
                        b1(1,3)=1;
                    end
                end
                if b1(1,1)==3
                    if b1(1,2)==2
                        b1(1,3)=1;
                    end
                end
                n7=n7+1;
                NextPoint{m,n7}=T{No.AssTri1(1)}{1,b1(1,3)};
                VisitedAssTriNo=[VisitedAssTriNo,No.AssTri1(1)];
                n2=find(UnvisitedAssTriNo==No.AssTri1(1));
                aa=[];
                ab=[];
                for n3=1:r1
                    if n3<n2
                        aa(n3)=UnvisitedAssTriNo(n3);
                    end
                    if n3>n2
                        ab(n3-n2)=UnvisitedAssTriNo(n3);
                    end
                end
                UnvisitedAssTriNo=[aa,ab];%finding associated tri from
unvisited tri
                r1=size(UnvisitedAssTriNo,2);
                n5=0;
                for n4=1:(j-1-n)%if the nextpoint equal to last step
nextpoint, clear this point
                    if NextPoint{m,n7}==NextPoint{(m-1),(n4+n)}
                       n5=1;
                    end
                end
                if n5==1
                    n7=n7-1;
                    break
                end
                c1=1;%whether equal to current step point
                if n7>100
                    for r4=1:10
                        if NextPoint{m,n7}==NextPoint{m,r4}
                            c1=2;
                            break
                        end
                    end
```

```
                    if c1==2%equal to the current step point
                        break%stop the loop of the for r4=1:10
                    end
                end%if n7>10
            end %end while r1>=2
            if i10==1
                i11=1;
                break% stop while n<=(j-1) loop
            end

            for i=1:(l1-1)%clear visitedTri
                VisitedTriNo=[VisitedTriNo,No.AssTri(i)];
                UnvisitedTriNo(No.AssTriAddress(i))=0;
                i1=i1+1;
            end%end for i=1:(l1-1)
        end %if l1>2
        if n7>100%if equal to the first point
            if c1==2
                 r6=1;
                 for r5=1:(n7-1)
                     if r5>=r4
                         NextPoint5{1,r6}=NextPoint{m,r5};
                         r6=r6+1;
                     end
                 end
                 for r5=1:10
                     NextPoint{m,n7+1-r5}={};
                 end
                 for r5=1:(r6-1)
                     NextPoint{m,r5}= NextPoint5{1,r5};
                 end
                 n7=r5;
                break%stop the loop of while n<=(j-1)
            end
        end
        n=n+1;
   end%if n>1
end %while n<=(j-1);
if i11==1
   Bif1={};
    Bif2={};
    d2=i7;
    d3=i8-1;
    for i4=1:i7
        Bif1{1,i4}=NextPoint{(m-1),i4};%birfucation 1
    end
    for i5=1:(j-1-i8+2)
        Bif1{1,(i4+i5)}=NextPoint{(m-1),d3};%birfucation 1
        d3=d3+1;
    end
    d3=1;
    for i6=(i7+1):(i8-2)
        Bif2{1,d3}=NextPoint{m-1,i6};%birfucation 2
```

```
            d3=d3+1;
        end

        j=i7+j-1-i8+3;
        break % stop the loop of m=15:15
    end

    if NextPoint{m,n7}==NextPoint{m,1}
       NextPoint{m,n7}={};
       n7=n7-1;
    end

    %****** if the last or last last point equal to some first 10 points %%
refer to bif1.1.1.step13******
    connectedNo4=0;
    connectedNo5=0;
    for n6=1:3
        if NextPoint{m,n7-1}==NextPoint{m,n6}
            connectedNo5=n6;
        end
    end
    for n6=1:3
        if NextPoint{m,n7}==NextPoint{m,n6}
            connectedNo4=n6;
        end
    end
    if connectedNo5>0
       for k3=1:(connectedNo5-1)%clear visited Tri
            for k=1:r
                if UnvisitedTriNo(k) ~= 0
                    if T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,k3}
                        VisitedTriNo=[VisitedTriNo,UnvisitedTriNo(k)];
                        UnvisitedTriNo(k)=0;
                        i1=i1+1;
                    else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,k3}
                            VisitedTriNo=[VisitedTriNo,UnvisitedTriNo(k)];
                            UnvisitedTriNo(k)=0;
                            i1=i1+1;
                        else if T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,k3}

VisitedTriNo=[VisitedTriNo,UnvisitedTriNo(k)];
                                UnvisitedTriNo(k)=0;
                                i1=i1+1;
                            end
                        end
                    end
                end
            end
        end

        for k3=1:(n7-connectedNo5)
            NextPoint{m,k3}=NextPoint{m,(k3+connectedNo5)};
        end
```

```
        for k3=1:connectedNo5
            NextPoint{m,n7+1-k3}={};
        end
        n7=n7-connectedNo5;

    else if connectedNo4>0
            for k3=1:(connectedNo4-1)%clear visited Tri
                for k=1:r
                    if UnvisitedTriNo(k) ~= 0
                        if T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,k3}
                            VisitedTriNo=[VisitedTriNo,UnvisitedTriNo(k)];
                            UnvisitedTriNo(k)=0;
                            i1=i1+1;
                        else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,k3}

VisitedTriNo=[VisitedTriNo,UnvisitedTriNo(k)];
                                UnvisitedTriNo(k)=0;
                                i1=i1+1;
                            else if
T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,k3}

VisitedTriNo=[VisitedTriNo,UnvisitedTriNo(k)];
                                    UnvisitedTriNo(k)=0;
                                    i1=i1+1;
                                end
                            end
                        end
                    end
                end

                for k3=1:(n7-connectedNo4)
                    NextPoint{m,k3}=NextPoint{m,(k3+connectedNo4)};
                end
                for k3=1:connectedNo4
                    NextPoint{m,n7+1-k3}={};
                end
                n7=n7-connectedNo4;
            end
    end

    %******* end equal to some point ***************************


    %****** cut off these points below to connected point  *******
    connectedNo1=0;
    connectedNo2=0;
    connectedNo3=0;
    connectedNo4=0;
    for n6=1:3
        l1=1;
        for k=1:r
```

```
    if UnvisitedTriNo(k) ~= 0
      if T{UnvisitedTriNo(k)}{1,1}==NextPoint{m,n6}
          atri{1,n6}(1,l1)=UnvisitedTriNo(k);
          atriaddress{1,n6}(1,l1)=k;
          l1=l1+1;
      else if T{UnvisitedTriNo(k)}{1,2}==NextPoint{m,n6}
               atri{1,n6}(1,l1)=UnvisitedTriNo(k);
               atriaddress{1,n6}(1,l1)=k;
               l1=l1+1;
          else if T{UnvisitedTriNo(k)}{1,3}==NextPoint{m,n6}
                   atri{1,n6}(1,l1)=UnvisitedTriNo(k);
                   atriaddress{1,n6}(1,l1)=k;
                   l1=l1+1;
              end
          end
      end
    end
end%for k=1:r
for k=1:(l1-1)
      if T{atri{1,n6}(1,k)}{1,1}==NextPoint{m,n7}
          connectedNo1=n6;
      else if T{atri{1,n6}(1,k)}{1,2}==NextPoint{m,n7}
              connectedNo1=n6;
          else if T{atri{1,n6}(1,k)}{1,3}==NextPoint{m,n7}
                  connectedNo1=n6;
              end
          end
      end
      if T{atri{1,n6}(1,k)}{1,1}==NextPoint{m,n7-1}
          connectedNo2=n6;
      else if T{atri{1,n6}(1,k)}{1,2}==NextPoint{m,n7-1}
              connectedNo2=n6;
          else if T{atri{1,n6}(1,k)}{1,3}==NextPoint{m,n7-1}
                  connectedNo2=n6;
              end
          end
      end
      if T{atri{1,n6}(1,k)}{1,1}==NextPoint{m,n7-2}
          connectedNo3=n6;
      else if T{atri{1,n6}(1,k)}{1,2}==NextPoint{m,n7-2}
              connectedNo3=n6;
          else if T{atri{1,n6}(1,k)}{1,3}==NextPoint{m,n7-2}
                  connectedNo3=n6;
              end
          end
      end
      if T{atri{1,n6}(1,k)}{1,1}==NextPoint{m,n7-3}
          connectedNo4=n6;
      else if T{atri{1,n6}(1,k)}{1,2}==NextPoint{m,n7-3}
              connectedNo4=n6;
          else if T{atri{1,n6}(1,k)}{1,3}==NextPoint{m,n7-3}
                  connectedNo4=n6;
              end
```

```
                    end
                end
            end%k=1:(l1-1)
            catri(1,n6)=l1-1;
        end%for n6=1:10
        if connectedNo4>0
           for k1=1:(connectedNo4-1)
                for i=1:catri(1,k1)%clear visitedTri
                    if UnvisitedTriNo(atriaddress{1,k1}(1,i))~=0
                       VisitedTriNo=[VisitedTriNo,atri{1,k1}(1,i)];
                       UnvisitedTriNo(atriaddress{1,k1}(1,i))=0;
                       i1=i1+1;
                    end
                end %end for i=1:catri(1,k1)
            end
            for k1=1:catri(1,connectedNo4)%clear visitedTri
                if T{atri{1,connectedNo4}(1,k1)}{1,1}==NextPoint{m,n7-2}
                    VisitedTriNo=[VisitedTriNo,atri{1,connectedNo4}(1,k1)];
                    UnvisitedTriNo(atriaddress{1,connectedNo4}(1,k1))=0;
                    i1=i1+1;
                  else if T{atri{1,connectedNo4}(1,k1)}{1,2}==NextPoint{m,n7-2}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo4}(1,k1)];
                        UnvisitedTriNo(atriaddress{1,connectedNo4}(1,k1))=0;
                        i1=i1+1;
                    else if
T{atri{1,connectedNo4}(1,k1)}{1,3}==NextPoint{m,n7-2}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo4}(1,k1)];

UnvisitedTriNo(atriaddress{1,connectedNo4}(1,k1))=0;
                            i1=i1+1;
                        end
                    end
                end
            end
            k1=1;
            for k2=1:n7%move the sequence
                if k2>=connectedNo4
                    NextPoint{m,k1}=NextPoint{m,k2};
                    k1=k1+1;
                end
            end
            for k2=1:(n7-k1+4)
                NextPoint{m,n7-k2+1}={};
            end
            n7=k1-4;
        else if connectedNo3>0
                for k1=1:(connectedNo3-1)
                    for i=1:catri(1,k1)%clear visitedTri
                        if UnvisitedTriNo(atriaddress{1,k1}(1,i))~=0
                           VisitedTriNo=[VisitedTriNo,atri{1,k1}(1,i)];
                           UnvisitedTriNo(atriaddress{1,k1}(1,i))=0;
```

```
                        i1=i1+1;
                    end
                end %end for i=1:catri(1,k1)
            end
            for k1=1:catri(1,connectedNo3)%clear visitedTri
                if T{atri{1,connectedNo3}(1,k1)}{1,1}==NextPoint{m,n7-1}
                    VisitedTriNo=[VisitedTriNo,atri{1,connectedNo3}(1,k1)];
                    UnvisitedTriNo(atriaddress{1,connectedNo3}(1,k1))=0;
                    i1=i1+1;
                  else if
T{atri{1,connectedNo3}(1,k1)}{1,2}==NextPoint{m,n7-1}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo3}(1,k1)];

UnvisitedTriNo(atriaddress{1,connectedNo3}(1,k1))=0;
                          i1=i1+1;
                      else if
T{atri{1,connectedNo3}(1,k1)}{1,3}==NextPoint{m,n7-1}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo3}(1,k1)];

UnvisitedTriNo(atriaddress{1,connectedNo3}(1,k1))=0;
                              i1=i1+1;
                          end
                      end
                end
            end
            k1=1;
            for k2=1:n7%move the sequence
                if k2>=connectedNo3
                    NextPoint{m,k1}=NextPoint{m,k2};
                    k1=k1+1;
                end
            end
            for k2=1:(n7-k1+3)
                NextPoint{m,n7-k2+1}={};
            end
            n7=k1-3;
        else if connectedNo2>0
                for k1=1:(connectedNo2-1)
                    for i=1:catri(1,k1)%clear visitedTri
                        if UnvisitedTriNo(atriaddress{1,k1}(1,i))~=0
                           VisitedTriNo=[VisitedTriNo,atri{1,k1}(1,i)];
                           UnvisitedTriNo(atriaddress{1,k1}(1,i))=0;
                           i1=i1+1;
                        end
                    end %end for i=1:catri(1,k1)
                end
                for k1=1:catri(1,connectedNo2)%clear visitedTri
                    if T{atri{1,connectedNo2}(1,k1)}{1,1}==NextPoint{m,n7}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo2}(1,k1)];
                        UnvisitedTriNo(atriaddress{1,connectedNo2}(1,k1))=0;
```

```
                        i1=i1+1;
                    else if
T{atri{1,connectedNo2}(1,k1)}{1,2}==NextPoint{m,n7}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo2}(1,k1)];

UnvisitedTriNo(atriaddress{1,connectedNo2}(1,k1))=0;
                            i1=i1+1;
                        else if
T{atri{1,connectedNo2}(1,k1)}{1,3}==NextPoint{m,n7}

VisitedTriNo=[VisitedTriNo,atri{1,connectedNo2}(1,k1)];

UnvisitedTriNo(atriaddress{1,connectedNo2}(1,k1))=0;
                                i1=i1+1;
                            end
                        end
                    end
                end


                k1=1;
                for k2=1:n7%move the sequence
                    if k2>=connectedNo2
                        NextPoint{m,k1}=NextPoint{m,k2};
                        k1=k1+1;
                    end
                end
                for k2=1:(n7-k1+2)
                    NextPoint{m,n7-k2+1}={};
                end
                n7=k1-2;
            else if connectedNo1>0
                    for k1=1:(connectedNo1-1)
                        for i=1:catri(1,k1)%clear visitedTri
                            if UnvisitedTriNo(atriaddress{1,k1}(1,i))~=0
                               VisitedTriNo=[VisitedTriNo,atri{1,k1}(1,i)];
                               UnvisitedTriNo(atriaddress{1,k1}(1,i))=0;
                               i1=i1+1;
                            end
                        end %end for i=1:catri(1,k1)
                    end
                    k1=1;
                    for k2=1:n7%move the sequence
                        if k2>=connectedNo1
                            NextPoint{m,k1}=NextPoint{m,k2};
                            k1=k1+1;
                        end
                    end
                    for k2=1:(n7-k1+1)
                        NextPoint{m,n7-k2+1}={};
                    end
                    n7=k1-1;
```

```
                end
            end
        end
    end



%***********  end cut off  ************************

%***********  whether this step is closed  *********
    connect1=0;
    r7=size(VisitedTriNo,2);
    for k=1:r7
        if T{VisitedTriNo(k)}{1,1}==NextPoint{m,1}
            if T{VisitedTriNo(k)}{1,2}==NextPoint{m,n7}
                connect1=1;
                break %for k=1:r
            else if T{VisitedTriNo(k)}{1,3}==NextPoint{m,n7}
                    connect1=1;
                    break %for k=1:r
                end
            end
        else if T{VisitedTriNo(k)}{1,2}==NextPoint{m,1}
                if T{VisitedTriNo(k)}{1,1}==NextPoint{m,n7}
                    connect1=1;
                    break %for k=1:r
                else if T{VisitedTriNo(k)}{1,3}==NextPoint{m,n7}
                         connect1=1;
                         break %for k=1:r
                    end
                end
            else if T{VisitedTriNo(k)}{1,3}==NextPoint{m,1}
                    if T{VisitedTriNo(k)}{1,1}==NextPoint{m,n7}
                        connect1=1;
                        break %for k=1:r
                    else if T{VisitedTriNo(k)}{1,2}==NextPoint{m,n7}
                                connect1=1;
                                break %for k=1:r
                        end
                    end
                end
            end
        end
    end%for k=1:r7
    if connect1==0
        fprintf('step %d is not closed\n',m)
    end
%************  end checking whether is closed  *********************
```

```
%************  birfucation checking   ****************************
    for i2=1:n7
        for i3=1:n7
            if i2~=i3
                if NextPoint{m,i2}==NextPoint{m,i3}
                    if (i3-i2)>10
                        fprintf('There is a birfucation on step %d\n',m)
                        break %for i3=1:n7
                    end
                end
            end
        end
        if i3~=n7
            d2=i2;
            d3=i3;
            break %for i2=1:n7
        end
    end
    if i2~=n7
        Bif1={};
        Bif2={};
        for i4=1:i2
            Bif1{1,i4}=NextPoint{m,i4};%birfucation 1
        end
        for i5=1:(n7-i3)
            Bif1{1,(i4+i5)}=NextPoint{m,d3+1};%birfucation 1
            d3=d3+1;
        end
        d3=i3;
        for i6=1:(i3-i2)
            Bif2{1,i6}=NextPoint{m,d2};%birfucation 2
            d2=d2+1;
        end
        j=i2+n7-i3+1;
        break% "for m= " roop
    end
%************  end birfucation checking   ************************

    VisitedTri{(m-1)}=VisitedTriNo;
    VisitedTriNo=[];
    PointCounter(m)=n7;
    j=n7+1;



end%for m=2:3

%*****************************************
```

References

Amenta, N., Bern, M. and Kamvysselis, M. 1998. A new Voronoi-based surface reconstruction algorithm. *SIGGRAPH 98*, 415-421.

Amenta, N., Choi, S., and Kolluri R. K. 2001. The power crust, Proc. 6th Annu. Sympos. Solid Modelling Applications, 249-260.

Antiga, L., Ene-Iordache, B., and Remuzzi, A. 2003. Computational geometry for patient-specific reconstruction and meshing of blood vessels from MR and CT angiography, *IEEE Transactions on Medical Image*. Vol. 22 (5), p.674 – 684.

Asgharian, B., and Price, O. T. 2006. Airflow distribution in the human lung and its influence on particle deposition. *Inhalation Toxicology*. 18:795-801.

Assimakopoulos, P. A., Boyd, D. P., Jaschke, W., and Lipton M. J. 1986. Spatial resolution analysis of computed tomographic images, *Invest. Radiol.,* vol. 21, pp. 260-271, 3.

Aykac, D., Hoffman, E. A., McLennan, G. and Reinhardt, J. M. 2003. Segmentation and analysis of the human airway tree from 3D X-Ray CT images, *IEEE Trans. Medical Imaging*.

Balásházy, I. and Hofmann, W. 1993. Particle deposition in airway bifurcations - I. Inspiratory flow. *Journal of Aerosol Science.* 24:745-772.

Balashazy, W. Hofmann and T. Heistracher, 1999. Computation of local enhancement factors for the quantification of particle deposition patterns in airway bifurcations, *Journal of Aerosol Science* 30, pp. 185–203.

Balashazy, I., Hofmann, W., and Heistracher, T. 2003. Local particle deposition patterns may play a key role in the development of lung cancer. *J. Appl. Physiol*. 94:1719–1725.

Ballard, D. H., and Brown, C. M. 1982. *Computer Vision*. Prentice Hall PTR.

Bashein, G., and Detmer, P. R. Graphics Gems IV. AP Professional, 1994. ch.1. Centroid of a Polygon, pp. 3-6.

Beare, R. 2006. A Locally Constrained Watershed Transform. *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 28, NO. 7.

Bilgen, D. 2000. Segmentation and analysis of the human airway tree from 3D X-ray CT images, Master's thesis, The University of Iowa, IA, USA.

Bloch, I. 1993. Fuzzy connectivity and mathematical morphology, *Pattern Recognition Letters*, vol. 14, pp. 483-488, 6.

Boissonnat, J. D. 1984. Geometric structures for three dimensional shape representation, *ACM Transact*. On Graphics 3(4), 266-286.

Boyden, E. A. 1955. Segmental anatomy of the lungs. McGraw-Hill.

Brown, R.H., Pearse D. B., Pyrgos, G., Liu, M. C., Togias, A., and Permutt, S. 2006. The structural basis of airways hyperresponsiveness in asthma. *J Appl Physiol* 101:30-39.

Burrowes, K. S., Hunter, P. J. and Twahai, M. H. 2005. Anatomically based finite element models of the human pulmonary arterial and venous trees including supernumerary vessels, *Journal of Applied Physiololgy*, Vol. 99, p.731 – 738.

Calay, R. K., Kurujareon, J. and Holdo, A. E. 2002. Numerical simulation of respiratory flow patterns within human lung. *Respiration Physiology.* 130:201-221.

Carraghan, R. and Pardalos, P. M. 1990. An exact algorithm for the maximum clique problem, *Operations Research Letters*, vol. 9, pp. 375-382, 11.

Carroll, N., J. Elliot, A. Morton and A. James. 1993. The structure of large and small airways in nonfatal and fatal asthma. Am Rev Respir Dis 147:405-410.

Carvalho, B. M., Gau C. J., Herman, G. T., and Kong, T. Y. 1999. Algorithms for Fuzzy Segmentation, *Pattern Analysis & Applications*, vol. 2, pp. 73-81.

Cefalu, W. 2003. New Options in Exogenous Insulin Therapy, Diabetes Roundtable, viewed 22 May, 2006. <http://www.diabetesroundtable.com/courses/update/options.asp>

Chang, H.K. 1989. Flow dynamics in the respiratory tract. *Respiratory Physiology*, an Analytical Approach, eds. H.K. Chang and M. Paiva. New York: Dekker.

Chang, H. K., and El Masry, O. A. 1982. A model study of flow dynamics in human central airways. Part I: Axial velocity profiles. *Respiration Physiology.* 49:75-95.

Choi, B.K. 2003. Editorial Part 1, *Computer Aided Design*, Vol. 35 (2), pp.125-126.

Choi, L.T., Tu, J.Y., Li, H.F., and Thien, F. 2007. Flow and particle deposition patterns in a realistic human double bifurcation airway model. *Inhal. Toxicol.* 19:117-131.

Comer, J.K., Kleinstreuer, C., and Zhang, Z. 2000a. Flow structures and particle deposition patterns in double-bifurcation airway models. Part 1. Air flow field. *J. Fluid Mech.* 435:25–54.

Comer, J.K., Kleinstreuer, C., and Zhang, Z. 2000b. Flow structures and particle deposition patterns in double-bifurcation airway models. Part 2. Aerosol transport and deposition. *J. Fluid Mech*. 435:25–54.

Corieri, P. 1994. Experimental and numerical investigation of flows in bifurcations within lung airways. PhD Thesis. University of Libre de Bruxelles.

Ebina, M., Takahashi, T., Chiba. and Motomiya, M. 1993. Cellular hypertrophy and hyperplasia of airway smooth muscles underlying bronchial asthma. A 3-D morphometric study. Am Rev Respir Dis 148:720–726.

Eck, M. and Hoppe, H. 1996. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type, *SIGGRAPH*, (New Orleans), p. 325—334.

Edelsbrunner, H. and M¨ucke, E. P. 1994. Three-dimensional alpha shapes, *ACM Trans. Graphics*, 13, 43–72.

Ertbruggen, C. V., Hirsch C., and Paiva, M. 2004. Anatomically based three-dimensional model of airways to simulate flow and particle transport using computational fluid dynamics, *Journal of Applied Physiololgy*. Vol. 98, p.970 – 980.

Fetita, C. I., Preteux F., Beigelman-Aubry C., and Grenier P. 2004. Pulmonary airways: 3-D reconstruction from multislice CT and clinical investigation, *IEEE Transactions on Medical Image*. Vol. 12 (11), p. 1353 – 1364.

Fouke JM, Wolin AD, Strohl KP, Galbraith GM. 1989 Feb. Elastic characteristics of the airway wall. *J Appl. Physiol*. 66(2):962-7.

Hahn, I., P.W. Scherer and M.M. Mozell. 1993. Velocity profiles measured for airflow through a large-scale model of the human nasal cavity. *J Appl. Physiol*. 75(5):2273-2287.

Haskin, P.H. and Goodman, L.R. 1982. Normal tracheal bifurcation angle: A reassessment. *American Journal Roentgen* 139:879-882.

Hassan, T., Timofeev, E. V., Saito, T., Shimizu, H., Ezura, M., Tominaga, T., Takahashi, A. and Takayama, K. 2004. Computational replicas: anatomic reconstructions of cerebral vessels as volume numerical grids at three-dimensional angiography. *American Journal of Neuroradiology*, Vol. 25, p.1356 – 1365.

Hollister, S., Levy, R., Chu, T., Hollaran, J. and Feinberg, S. 2000. An image based approach for designing and manufacturing of craniofacial scaffolds, *Int J Oral Maxillofacial Surg* 29 (2000), pp. 67–71.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1992. Surface reconstruction from unorganized points, *SIGGRAPH 92*, 71-78.

Horsfield, K., Gladys, D., Olson, D. E. Finlay, G. F., and Cumming, G. 1971. Models of the human bronchial tree. *Journal of Applied Physiology.* 31:207-217.

Ibanez, L., Schroeder, W., Ng, L., and Cates, J. 2007. The ITK Software Guide: The Insight Segmentation and Registration Toolkit. Technical report, Kitware, Inc, Available online:http://www.itk.org/ItkSoftwareGuide.pdf.

Isabey, D. and H.K. Chang. 1981. Steady and unsteady pressure-flow relationships in central airways. *J. Appl. Physiol*. 51:1338-1348.

Isabela, C., Silva, S., Colby, T.V., and Müller, N.L. 2004. Asthma and Associated Conditions: High-Resolution CT and Pathologic Findings. AJR 183.

ITK.Org. http://www.itk.org/

Kaiser Santa Rosa.org. http://www.kaisersantarosa.org/medicalimaging

Karabulut, N. 2005. CT Assessment of tracheal carinal angle and its determinants. *British Journal of Radiology* 78:787-790.

Katz, I. M., and Martonen, T. 1996. Three-dimensional fluid particle trajectories in the human larynx and trachea. *Journal of Aerosol Medicine-Deposition Clearance and Effects in the Lung*. 9:513-520.

Kim, C. S., Hu, S. C., Dewitt, P., and Gerrity, T. R. 1996. Assessment of regional deposition of inhaled particles in human lungs by serial bolus delivery method. *Journal of Applied Physiology.* 81:2203-2213.

Koikkalainen, J., and Lotjonen, J. 2004. Reconstruction of 3D head geometry from digitized point sets: an evaluation study, *IEEE Transactions on Information Technology in Biomedicine*, Vol. 8(3), p.377 – 385.

Kuwano, K., Bosken, C.H., Pare, P.D., Bai, T.R., Wiggs, B.R., and Hogg, J.C. 1993. Small airways dimensions in asthma and in chronic obstructive pulmonary disease. *Am Rev Respir Dis* 148:1220-1225.

Lambert, R.K., Wiggs, B.R., Kuwano, K., Hogg, J.C., and Pare, P.D. 1993. Functional significance of increased airway smooth muscle in asthma and COPD. *J Appl Physiol* 74:2771-2781.

Liu Y, R.M.C. So and Zhang C.H., 2002. Modeling the bifurcating flow in a human lung airway, *Journal of Biomechanics* 35, pp. 465–473.

Liu Y. R.M.C. So and Zhang C.H., 2003. Modeling the bifurcating flow in an asymmetric human lung airway, *Journal of Biomechanics* 35, pp. 951–959.

Long, Q., Ariff, B., Zhao, S. Z., Thom, S.A., Hughes, A. D., and Xu, X. Y. 2003. Reproducibility study of 3D geometrical reconstruction of tgeh human carotid bifurcation from magnetic resonance images, *Magnet Resonance in Medicine*, Vol. 49, p.665-674.

Longest, P.W., S. Vinchurkara and T. Martonen. 2006. Transport and deposition of respiratory aerosols in models of childhood asthma. *Aerosol Science* 37:1234-1257.

Lorenson, W. E., and Cline, H. E. 1987. Marching cubes: a high resolution 3D surface construction algorithm, *Comput Graphics 21*, pp. 163–169.

Lundblad, L.K., Thompson-Figueroa, J., Allen, G.B., Rinaldi, L., Norton, R.J., Irvin, C.G., and Bates, J.H. 2007. Airways hyperresponsiveness in allergically inflamed mice: the role of airway closure. *Am J Resp Crit Care Med* 175:768-774.

Mankovich, N. J., Robertson, D. R., and Cheeseman, A. M. 1990. Three-dimensional image display in medicine, *J Digit Imaging* 3 (2), pp. 69–80.

Martonen T.B., Guan X. and Schreck R.M., 1994. Effects of carinal ridge shapes on lung airstreams, *Aerosol Science and Technology* 21, pp. 119–136.

Materialise Inc., http://www.materialise.com/mimics

Menon, A. S., Weber, M. E., and Chang, H. K. 1984. Model study of flow dynamics in human central airways. Part III: Oscillatory velocity profiles. *Respiration Physiology.* 55:255-275.

Mitchell, H.W. and Gray, P.R. 1999. Assessment of the dynamic relationship between external diameter and lumen flow in isolated bronchi. *Respir Physiol* 116:67-76.

Nowak, N., P.P Kakade and A.V. Annapragada. 2003. Computational fluid dynamics simulation of airflow and aerosol deposition in human lungs. *Annals of Biomedical Engineering* 31:374-390.

Pascual, R.M. and Peters, S.P. 2005. Airway remodeling contributes to the progressive loss of lung function in asthma: an overview. *J Allergy Clin Immunol*. 116:477-486.

Palágyi, K., Tschirren, J., Hoffman, E. A., Sonka, M. 2006. Quantitative analysis of pulmonary airway tree structures, *Computeres in Biology and Medicine*, Vol. 36, p.974 – 996.

Pedley, T.J., Schroter R.C. and Sudlow M.F. 1977. Gas flow and mixing in the airways. In Bioengineering Aspects of the Lung,, ed. J. West. New York: Dekker.

Permutt, S. 2007. The role of the large airways on smooth muscle contraction in asthma. *J Appl Physiol*; doi:10.1152/japplphysiol.00568.

Piegl, L.A. and Tiller, W. 1995. The NURBS book. New York, NY: Springer-Verlag.

Quatember, B., and Muhlthaler, H. 2003. Generationof CFD meshes from biplane angiograms: an example of image-based mesh generation and simulation. *Applied Numbercial Mathematics*, Vol. 46, p.379 – 397.

Raindrop Inc. 2004. GeoMagic Studio User Manual, Research Triangle, NC, USA.

Schlesinger, R. B., Gurman, J. L., and Lippmann, M. 1982. Particle deposition within bronchial airways: Comparisons using constant and cyclic inspiratory flows. *Annals of Occupational Hygiene.* 26:47-64.

Schorter, R. C., and Sudlow, M. F. 1969. Flow patterns in models of the human bronchial airways. *Respiration Physiology*. 7:341-355.

Schroeder, W., Martin, K. and Lorensen, B. 1996. The visualization toolkit: an object-oriented approach to 3-D graphics. Prentice Hall, New Jersey.

Schroeder, W., Avila, L., and Hoffman, W. 2000. Visualizing with VTK: a tutorial. *IEEE Computer Graphics and Applications*, 20(5):20–27.

Shannon, J. M., and Deterding R. R. 1997. Epithelial-mesenchymal interactions in lung development. In: *Lung Growth and Development*. McDonald JA. New York: Dekker. p.81-118.

Slutsky, A.S., Berdine G. G. and Drazen J.M. 1981. Oscillatory flow and quasi-steady behavior in a model of human central airways. *J. Appl. Physiol*. 50(6):1293-1299.

Sorkness, R. 2007. Commentary on The role of the large airways on smooth muscle contraction in asthma. *J Appl Physiol* 103:1464-doi:1410.1152/japplphysiol.00702.02007.

Sullivan, K.J., and Chang, H.K. 1991. Steady and oscillatory trans-nasal pressure-flow relationships in healthy adults. *J. Appl. Physiol*. 71:983-992.

Sun, W., Darling, A., Starly, B., and Nam, J. 2004. Computer-aided tissue engineering: overview, scope and challenges. *J Biotechnol Appl Biochem* 39 (1), pp. 29–47.

Swift, D.L. and Proctor, D.F. 1977. Access of air to the respiratory tract. In Respiratory Defence Mechanisms, eds. J.D. Brain, D.F. Proctor and L.M. Reid. New York, NY: Marcel Dekker.

Tawhai, M. H., Hunter, M. H., Tschirren, J., Reinhardt, J., McLennan, G. and Hoffman, E. A. 2004. CT-based geometry analysis and finite element models of the human and ovine bronchial tree. *Journal of Applied Physiololgy* . Vol. 97, p.2301 – 2321.

Tgavalekos, N. T., Tawhai, M., Harris, R. S., Mush, G., Vidal-Melo, M., Venegas, J. G. and Lutchen, K.R. 2005. Identifying airways responsible for hetetogeneous ventilation and mechanical dysfunction is asthma: an image function modelling approach, *Journal of Applied Physiololgy*. Vol. 99, p.2388 – 2397.

Thompson, B R., King, G., and Harding, R. 2007. Commentary on The role of the large airways on smooth muscle contraction in asthma. *J Appl Physiol* 103:1465 - doi:1410.1152/japplphysiol.00696.02007

Travis, W.E., Colby, T.V., Koss, M.N., Rosado de Christenson, M.L., and Müller, N.L. 2002. Non-neoplastic disorders of the lower respiratory tract. Washington, DC: Armed Forces Institute of Pathology:381-471.

Tschirren, J., McLennan, G., Palágyi, K., Hoffman, E. A., and Sonka, M. 2005. Matching and anatomical labeling of human airway tree, *IEEE Transactions on Medical Imaging*. Vol. 24(12): 1540–1547.

Tschirren, Juerg., Hoffman, E. A., McLennan Geoffrey and Sonka Milan. 2004. Segmentation and quantitative analysis of intrathoracic airway trees in low-dose CT scans, Medical Imaging*, IEEE Transactions on*. Submitted.

Van Ertbruggen, C., C. Hirsch and M. Paiva. 2005. Anatomically based three-dimensional model of airways to simulate flow and particle transport using computational fluid dynamics. *J. Appl. Physiol*. 98:970-980.

Vial, L., Perchet, D., Fodil, R., Caillibotte, G., Fetita, C., Preteux, F., Beigelman-Aubry, C., Grenier, P., Thiriet, M., Isabey, D., and Sbirlea-Apiou, S. 2005. Airflow modeling of steady inspiration in two realistic proximal airway trees reconstructed from human thoracic tomodensitometric images. *Computer Methods in Biomechanics and Biomedical Engineering.* 8:267-277.

Viceconti, M., Zannoni, C., Testi, D., and Capello, A. 1999. CT data sets surface extraction for biomechanical modeling of long bones, *Comput Methods Programs Biomed* 59 pp. 159–166.

Vignola, A.M., Kips, J., and Bousquet, J. 2000. Tissue remodeling as a feature of persistent asthma. *J Allergy Clin Immunol* 105:1041-1053.

VIDA Diagnostic Inc., http://www.vidadiagnostics.com

Volkau, I., Zheng, W., Baimouratov, R., Aziz, A., and Nowinski, W. L. 2005. Geometric modelling of the human normal cerebral arterial system. *IEEE Transactions on Medical Imaging*. Vol. 24 (4), p.529-539.

VTK.Org. http://www.vtk.org/

Wagers, S., Haverkamp, H.C., Bates, J.H.T., Norton, R.J., Thompson-Figueroa, J.A., Sullivan, M.J., and Irvin, C.G. 2007. Intrinsic and antigen-induced airway hyperresponsiveness are the result of diverse physiological mechanisms. 102: 221–230. *J Appl Physiol* 102:221-230.

Weibel, E.R. 1963. Morphometry of the human lung. New York, US: Academic Press.

Wonkyu Park, Eric A. Hoffman, and Milan Sonka 1998. Segmentation of Intrathoracic Airway Trees: A Fuzzy Logic Approach. *IEEE Transactions on medical imaging, Vol. 17, No. 4*

Wright, G. 1997. Magnetic resonance imaging. *IEEE Signal Processing Magazine*, 14(1):56–66.

Yang, X.L., Liu, L., So, R.M.C., and Yang, J.M. 2006. The effect of inlet velocity profile on the bifurcation COPD airway flow. *Computers in Biology and Medicine* 36(2):181-194.

Yoo, T., Ackerman, M., Lorensen, W., Schroeder, W., Chalana, V., Aylward, S., Metaxes, D., and Whitaker, R. 2002. Engineering and algorithm design for an image processing API: A technical report on ITK - the Insight Toolkit. In Proceedings of Medicine Meets Virtual Reality, pages 586–592.

Zhang, L., Asgharian, B., and Anjilvel, S. 1997. Inertial deposition of particles in the human upper airway bifurcations. *Aerosol Science and Technology.* 26:97-110.

Zhang, Z., C. Kleinstreuer, C.S. Kim and A.J. Hickey. 2002. Aerosol transport and deposition in a triple bifurcation bronchial airway model with local tumours. *Inhalation Toxicology* 14:1111-1133.

Zhang, Z., and Kleinstreuer, C. 2002. Transient airflow structures and particle transport in a sequentially branching lung airway model. *Physics of Fluids.* 14:862-880.

Zhang, Y. and Finlay W. H., 2005, Measurement of the effect of cartilaginous rings on particle deposition in a proximal lung bifurcation model. *Aerosol Sci*. Tech. 39:394–399.

Zhang, Z., C. Kleinstreuer, J.F. Donohue and C.S. Kim. 2005. Comparison of micro- and nano-size particle depositions in a human upper airway model. *J. Aerosol Sci*. 36:211-233.

Zhang, Z., C. Kleinstreuer and C.S. Kim. 2002a. Cyclic micron-size particle inhalation and deposition in a triple bifurcation lung airway model. *Aerosol Sci*. 33:257–281.

Zhang, Z., C. Kleinstreuer and C.S. Kim. 2002b. Gas-solid two-phase flow in a triple bifurcating lung airway model. Int. *J. Multiphase Flow* 28:1021-1046.

Zhao, Y., and Lieber, B. B. 1994. Steady inspiratory flow in a model symmetric bifurcation. *Journal of Biomechanical Engineering.* 116:488-496.

Zhou, Y. and Y.S. Cheng. 2005. Particle deposition in a cast of human tracheobronchial airways. *Aerosol Sci*. Tech. 39:492-500.