System for the Analysis and Visualization of Large 3D Anatomical Trees

Kun-Chang Yu,[1] Erik L. Ritman,[2] and William E. Higgins[1,*]

[1]Dept. of Electrical Engineering, Penn State University, University Park, PA 16802 USA

[2]Dept. of Physiology and Biophysics, Mayo Foundation, Rochester, MN 55905 USA

[*]Corresponding author. Fax: 1-814-863-5341. Email address: weh2@psu.edu

Abstract

Modern micro-CT and multi-detector helical CT scanners can produce high-resolution 3D digital images of various anatomical trees. The large size and complexity of these trees make it essentially impossible to define them interactively. Automatic approaches have been proposed for a few specific problems, but none of these approaches guarantee extracting geometrically accurate multi-generational tree structures. This paper proposes an interactive system for defining and visualizing large anatomical trees and for subsequent quantitative data mining. The system consists of a large number of tools for automatic image analysis, semi-automatic and interactive tree editing, and an assortment of visualization tools. Results are presented for a variety of 3D high-resolution images.

Keywords: 3D visualization, anatomical trees, micro-CT imaging, multidetector CT, arterial trees, tree analysis, virtual endoscopy, data mining.

1 INTRODUCTION

Modern micro-CT [1–5] and multi-detector CT scanners [6,7] can produce high-resolution 3D digital images of various anatomical tree structures, such as the coronary or hepatic vasculature [4, 8–10] and the airway tree [11]. For example, a typical 3D micro-CT image of the coronary arterial tree can consist of several hundred megabytes of image data, with a voxel resolution on the order of ten microns, and depict many generations of connected branches. The quantitative analysis and subsequent visualization of such images poses a considerable challenge. The sheer size and complexity of such images makes interactive manual tree definition out of the question. *We propose a system for the complete definition and quantitative analysis of anatomical trees contained in high-resolution 3D digital images.*

Rudimentary automated techniques for the definition of the vascular and hepatic trees have been previously suggested [4, 8–10, 12–15]. Also, partly in recognition of the great complexity of vascular trees depicted in large 3D images, some automated analysis work has only focused on a principal pathway, where a principal pathway is designated as one selected path from the beginning of the tree (the "root") to the end of one terminating branch [16–18]. A few recent efforts have proposed rudimentary interactive "point and click" editing for deleting unwanted branches in vascular trees [10] and deleting unwanted subtrees in cerebral vascular networks [19].

While existing techniques can give a high percentage of apparently correct branches, no technique guarantees geometrically accurate multi-generational tree structures, even to generation N,

where integer N is less than the total number of branch generations depicted in an input image. Tree errors occur because of insufficient data resolution, artifacts from improper image reconstruction, flaws in specimen preparation, and imperfections arising in acquiring the image data during scanning [4]. Figure 1 illustrates the basic problem. Various defects can occur in the defined result, such as: (a) missed branches; (b) broken branches; (c) spurious branches, resulting in extra bifurcation points and errors in branch generation indices; (d) anatomically implausible loops; (e) imprecise axes definition within a branch-point region, producing incorrect local branch geometry; and (f) improperly centered branch axes, resulting in incorrect branch measurements. These problems result in a tree that has an incorrect geometrical structure. Further, errors arising in the first few generations cause errors to propagate to all other generations.
FIGURE 1 here.

These observations lead to the basic philosophy we used in designing our proposed system: (1) it is unrealistic and counterproductive to rely strictly on improved scanning technology and improved automated image-processing algorithms for defining an accurate tree; (2) automated techniques, despite their imperfections, are essential in providing a complete description of a tree, as they can provide a high percentage of the correct tree structure; (3) judicious human interaction is essential for arriving at accurate useful analysis of a tree.

This paper describes our system, dubbed the Tree Analyzer. Section II gives an overview of the system, Section III describes the system architecture, Section IV discusses how the system is used to process a given 3D image, and Section V provides quantitative and pictorial results for various 3D micro-CT and multi-detector CT images. Finally, Section VI offers concluding comments.

2 SYSTEM OVERVIEW

The qualitative design criteria that drove the Tree Analyzer's construction are as follows: (1) be computationally efficient; (2) require only a reasonable amount of human interaction; (3) function over a wide range of anatomical and data variations. The system consists of three components:

1. A top-level graphical user interface (GUI) for performing all interactions.
2. A 3D image-processing toolbox for automatic image analysis.
3. A set of interactive tools for visualization, tree editing, and data mining.

Section 3 provides more detail on these components. The user applies the system to a given 3D image following a four-stage approach (Figure 2):

1. Apply automated image analysis to extract an initial raw tree and generate tree surface data suitable for follow-up visualization and tree editing.

2. Automatically define the initial raw central axes, or centerlines, for the extracted tree.
3. Automatically "diagnose" the tree for possible tree defects, such as broken branches, loops, etc., per Figure 1. Next, invoke various semi-automatic and interactive tools to examine and correct the identified tree defects.
4. Perform interactive data mining to extract and examine quantitative tree data.

At the end of this process, the user has a quantitative description of the desired anatomical tree. Section 4 gives detail on this process. As the Tree Analyzer has considerable breadth and capability, we cannot give complete detail on all of its functionality in this paper. References [20–24] provide supplemental details on various aspects of the system.
FIGURE 2 here.

The Tree Analyzer was built on a PC platform and greatly expands upon the earlier system of Wan [10]. The software was developed using Visual Studio.Net 2003 and Visual C++. The code for managing windows and dialogue boxes, for performing basic input-output, for storing data objects, and for accomplishing other functions, draws upon the standard Microsoft Foundation Class (MFC) library. Supplemental user interface components draw upon those in the Business Component Gallery (BCG), an extension library for MFC. Some of the Tree Analyzer's visualization features are derived from functions in the Visualization Toolkit (VTK) and OpenGL [25,26].

For a given input 3D image, all data components for a completely processed tree are stored in a central data structure referred to as the *case study* [27]. All system inputs, interactions, and outputs occur through the case study. The case study contains the following data elements:

1. Original 3D gray-scale image
2. 3D image of the segmented tree
3. Central axes of the tree
4. Surface data for rendering the tree
5. A script containing the sequence of automated image-processing operations used for performing all Stage-1 and Stage-2 (raw tree extraction and centerline analysis) analyses.
6. Text notes for documenting observations on the case
7. Quantitative data describing the axial structure of the tree
8. Viewing parameters saved for visualizing a tree during an interactive data-mining session.

The construction of the case study occurs through user interactions and command invocations with the Tree Analyzer's GUI following the four-stage processing flow of Figure 2. The case study serves as a repository for all data elements during the analysis of a 3D image, and it also enables a user to save work during different interactive sessions.

3 SYSTEM ARCHITECTURE

This section overviews the Tree Analyzer's three major components: the GUI, the 3D image-processing toolbox, and the interactive tools.

3.1 Graphical User Interface (GUI)

Figure 3 gives a sample composite view of the Tree Analyzer's graphical user interface. The default system toolbar at the top of the GUI follows the standard Microsoft software lay-out. The "File" options provide standard means for performing load and write operations on data inputs and outputs. The "Edit" options enable various tree editing functions, such as invoking automated analysis to extract the raw tree and for performing various tree modifications. The "Global" options refer to the mode of operation when multiple visualization tools are simultaneously running. The "Tools" options give a list of the various visualization tools available to the user. Finally, the "View," "Window," and "Help," options offer a number of standard aids found in Windows-based software packages. The second row of the GUI in Figure 3 provides toolbar commands that the user can pick to begin and build a case study and to invoke various tree editing functions.
FIGURE 3 here.

The bottom portion of the GUI in Figure 3 depicts, from left to right, the data elements of the currently loaded case study, the current position of the *picker*, and two panels for a *Tree Diagnostician*, which is used for automatically identifying defects in the currently loaded tree (see Section 4). The picker represents a specific 3D site of interest focused on by the system. All invoked visualization tools tend to depict views relative to the picker's location.

The remainder of the screen space is dedicated to various selected visualization tools. Figure 3 shows two rendering views; many others are possible as discussed below. Notably, Figure 3 only occupies one computer monitor. Yet, the images we have been studying in our research tend to be very large and rich in detail. Hence, to gain a fuller appreciation of the contents of the 3D image data, we have tended to use the Tree Analyzer with a display extending over two monitors. Figure 4 gives an example display for the second monitor.
FIGURE 4 here.

3.2 3D Image-Processing Toolbox

The 3D Image-Processing Toolbox is integrated as part of the kernel in the Tree Analyzer's software structure. Invoked either from the Edit option or toolbar, it enables the construction and invocation of a *script* used for performing Stage-1 and Stage-2 automated analysis. A script consist of a sequence of selected operations to perform a specified image-processing task [28]. Over 100 operations are available in the following general categories: (a) image enhancement; (b) mathematical morphology operations for shape-based image analysis; (c) topological operations for defining connected components, deleting interior cavities, extracting and manipulating central-axes structures, etc.; (d) image segmentation; (e) image manipulation operations, such as adding two images and performing simple logical combinations of images; and (f) other miscellaneous system operations, for manipulating intermediate results, input-output, and computing region surface representations.

The user interacts with the Tree Analyzer's GUI to construct a sequence of desired operations. In general a specialized script must be constructed for anatomical trees of a particular type (e.g., coronary arterial tree or airway tree) and derived from a particular scanner type (e.g., micro-CT scanner or a multi-detector CT scanner). In our research, we have constructed scripts for extracting the coronary vasculature, hepatic vasculature, and airway tree. We have used essentially the same script for all three of these problems, with the significant change being in the image segmentation method chosen. The Stage-2 centerline analysis is independent of the problem considered, provided the input is for a branching anatomical tree.

3.3 Interactive Tools

A large set of interactive tools are available for visualization, tree editing, and data mining [29,30]. Below is a list of the basic tools:

1. Global and local surface renderings of the segmented tree (3D Render tool). The global rendering shows the entire tree, while a local rendering zooms in on a particular site of interest. These views also depict the defined central axes of the tree.

   In addition, stereo glasses can be used to perceive the tree's branch structure in "true" 3D. Stereo viewing enables the user to gain a greater appreciation of a given tree's topology. We have used inexpensive stereo glasses by ProView ($6.95 per set) for stereo viewing.

2. Transverse, sagittal, and coronal slice views. These are standard 2D sections of the gray-scale data oriented in the x-y, y-z, and x-z planes, respectively [7].

3. Transverse, sagittal, and coronal projections, which give 2D orthographic projections of selected subsets of the gray-scale data along either the z, x, or y viewing directions, respectively.

4. Transverse, sagittal, and coronal thin-slab renderings, which enable more sophisticated views of subsets of the gray-scale data.
5. A bounding box feature that enables the user to view either a 3D rendering, slice, projection, or slab view focusing on a localized 3D bounding box about a selected 3D site of interest.
6. A tree map that depicts the defined axial structure of the tree in a graphical form, enables some basic tree editing, and provides local quantitative tree data.
7. A quantitative summary of an extracted tree [10].

The Tree Analyzer uses a series of VTK functions to compute 3D renderings and to perform stereo viewing [25]. Figure 3 gives examples of the 3D Render tool, while Figure 4 illustrates examples of the slice, projection, and slab tools. Thin slabs incorporating data depth and various gray-scale window settings can be computed to reveal more structural data inside a 3D volumetric image than possible with basic orthographic projections [31]. Figure 4 also gives an example of the bounding box feature. Anatomical trees can have an extremely dense 3D structure, as shown by a few examples later in this paper. The bounding-box feature can be particularly useful for focusing attention on complex interior 3D regions of interest. This feature enables the detailed study of a particular local tree region and is also useful for interactively fixing local tree defects. Figure 14, discussed later, gives an example of a quantitative summary. References [20, 21, 23, 24] give other examples and discuss other capabilities of the Tree Analyzer's tools. Most of the tools above can be found in various forms in other software packages; e.g., [7, 25, 32–34]. The Tree Analyzer, however, has an extensive number of unique capabilities for editing and performing data mining on 3D anatomical trees.

3.3.1 Interaction Modes

All tools "obey the system," per the user's movement of the computer mouse. The current system status while working with a 3D image is captured in the *view state*. The view state graphically exists as the picker and appears either as a ball or a set of cross hairs in the visualization tools. It is numerically given in the Global Status window (bottom of Figure 3, second from the left). In general, to interact with an extracted tree, the user points to either a tree rendering or some other active view. All other active views then follow the current view state. Thus, for a selected 3D site, the user gets a composite view of image data about the site through the use of the multiple tools. During interaction, the visualization tools can present data via four viewing modes:

1. Fixed — The current 3D site of the mouse appears on the view, with no other view change.
2. Move — The view becomes centered about the 3D site with no view-magnification change.

3. Detail — The view zooms in and becomes centered about the 3D site.
4. Navigation — The view shows an interior (virtual endoscopic) view of the tree branch at the selected 3D site (3D Render tool only) [35].

A major and novel capability is the Tree Analyzer's extensive facility for editing 3D trees. To facilitate this capability, four modes of operation exist for interacting with the central axes and 3D renderings through the picker. These modes all act within the camera space observable through a 3D Render tool and enable selection of points in 3D space:

1. Tree mode — When the picker (mouse cursor) is moved over a 3D rendering and a point is selected by clicking the mouse, the nearest currently defined central-axis point is selected.
2. 3D Site Locator — The picker acts as a "shooter" through 3D space and can be used to select any 3D point in a 3D rendering. This is done in two steps. First, a point of interest is selected on a 3D rendering from an initial view of a tree; this defines a 2D line of possible points in 3D space along the shooter's point of view. Next, the user rotates the view appropriately and selects a point on the line to complete the definition of the desired 3D point. See Figure 5.
3. Intersection Locator — The user situates the picker over a portion of a 3D rendered tree surface; this defines a line segment through which the shooter's point of view passes through the existing surface. The midpoint of this line segment is selected as the point of interest.
4. 3D Cursor — The user can move about in 3D space and confine attention to a focal plane or parallel planes in x, y, and z, as determined by the location of the picker; one axis is locked and the other two free axes can be varied as the user moves the cursor about a 3D Render view. This can bring up various slice and thin-slab views relative to the selected focal plane, as shown in Figure 6.

FIGURES 5 AND 6 here.
While considerable flexibility exists for moving about the 3D data, this is often a difficult way to glean the topological structure and connectivity of a tree. This is because the branches of a tree can twist and meander in essentially any direction in 3D. The Bounding Box feature and the Move and Detail viewing modes can help the user focus on local portions of a tree.

3.3.2 2D Tree Map

While the Bounding Box feature and the Move and Detail modes are helpful, they still cannot overcome the difficulties of perceiving the hierarchical complexities of a large tree. The 2D Tree Map is a particularly effective tool for overcoming these difficulties in the 3D perception and editing of an extracted 3D tree. It has considerable functionality for tree visualization, quantitative data mining, and tree editing, and it greatly expands upon a similar rudimentary tool integrated in the Analyze/AVW^{TM} package [36]. This section discusses the 2D Tree Map's visualization capabilities, while subsequent sections illustrate the tool's use for tree editing and data mining.

The 2D Tree Map is based around a 2D symbolic representation of a tree. It essentially "flattens" out the 3D tree into a 2D symbolic data structure, and, hence, is often much easier to manipulate than the 3D x-y-z spatial-domain representation of the tree. Technically, the 2D Tree Map presents the extracted axial structure of the anatomical tree as a hierarchical directed graph [37]. Graphs have shown much utility for data mining and for information navigation and visualization [38]. The graph representation in the form of a tree is ideal for presenting anatomical tree structure. Figure 7 gives an example of the 2D Tree Map.
FIGURE 7 here.

For the 2D Tree Map, nodes represent either branch points or branch terminations, while edges represent tree branches. In general the first branch of the tree represents the root branch, which subsequently bifurcates at a node, representing the tree's first branch point, to form child branches. Child branches in turn spawn more generations of branches. The 2D Tree Map goes well beyond display of mere branch hierarchy. Considerable quantitative data is also accessible numerically and visually from the 2D Tree Map. Branch number and generation index, branch length, average branch cross-sectional area (CSA), average branch diameter, and CSA and diameter along distinct local sites of a branch are also available, as discussed below.

The example of Figure 7a only shows generations 0 through 4, whereas the complete tree actually has 15 generations. A red node enclosing a "-" sign signifies a branch point. All other nodes terminate branches or portions of the tree and give access to other information. A solid green node denotes a terminated branch. A blue node with a "+" inside means that branch generations >4 exist at the node. The user can click on such a node to reveal generation-5 information. By further clicking on the "+" nodes, the tree expands further. In addition, the user can click with the middle mouse button on an interior tree node and spawn a separate local 2D tree map (Figure 7b). These facilities for expanding and contracting portions of a tree and for spawning separate local 2D Tree Maps are extremely useful for examining the structure of large complex trees.

In a given view, the branches are ordered from left to right, with the leftmost branch spawned by a particular branch point having the largest average diameter and the second child branch appearing to the right. Thus, the node ordering in the tree map depends on the quantitative shape attributes of tree branches. This convention gives rise to a standard branch labeling scheme, independent of the anatomy [39], that can be shown on the Tree Map if desired. The root branch is designated by the ordered pair (0,1) (generation 0, branch 1). The child branches of the root branch are labeled as follows. The largest-diameter branch at generation 1 is designated (1,1), while the remaining generation-1 branch is designated (1,2). Child branches spawned by (1,1) are labeled (2,1) and (2,2), while child branches spawned by (1,2) are labeled (2,3) and (2,4). Continuing on to generation 3, children of branch (2,1) are labeled (3,1) and (3,2), etc. If, for example, branch (2,2) has no children (i.e., it is a terminating branch), then labels (3,3) and (3,4) are skipped and labeling continues for the children of (2,3), which get labels (3,5) and (3,6), etc. Trifurcations and higher can be easily handled in this scheme.

Further, the user can click on a node to reveal a quantitative summary of the tree at that location, giving branch and generation index, branch length, and average branch cross-sectional area (CSA). Figure 7 illustrates these capabilities. Quantitative information can also be displayed visually with the tree map itself. For example, Figure 7a actually displays branches in average-diameter mode: the width of each branch in the tree map is proportional to its actual average width, as derived from the segmented tree. In Figure 7b, the 2D Tree Map is in diameter mode: the diameters of all 2D cross-sections constituting a particular branch are graphically depicted along a branch's extent in the 2D Tree Map. Finally, the local 2D Tree Map in Figure 7b is also in length mode: the length of each depicted branch is proportional to the actual branch length, with actual accumulated length given along the right-hard side of the view. Cross-sectional area can also be presented visually on a 2D Tree Map.

As with the other visualization tools, the user can point to a branch or node in the 2D Tree Map, which in turn highlights the corresponding 3D sites in the other active 2D graphical viewing tools. This often tends to be much easier and far more intuitive with the 2D Tree Map than with other tools. Several useful interactive functions of the 2D Tree Map exploit this feature.

Subsets of extracted branches that connect together to form a loop generally indicate an anomaly in the automatic segmentation of the raw tree, as such loops are seldom anatomically plausible. Loops have a special display form in the 2D Tree Map. The involved branches are displayed in yellow and the involved branch-point nodes are colored red and outlined in yellow (illustrated later in Figure 11). The user can then point to a node within the loop and examine the 3D image data about this region in other visualization tools.

4 SYSTEM USAGE

This section describes how to produce a complete tree and its associated quantitative description, per the procedure summarized in Figure 2. For Stages 1 and 2, the user loads an image to begin a case study and then runs a script to generate the raw segmented tree, surface data, and central axes. Next, in Stage 3, the Tree Diagnostician provides a list of possible tree defects, and the user applies various graphical tools to correct the defects. Finally, a quantitative summary is produced for the final tree in Stage 4. More detail appears below on these steps.

4.1 Stages 1 and 2: Automated Processing Operations

The first two stages involve automated processing operations for defining the initial raw tree and its centerline structure. These operations, all part of the 3D Image Processing Toolbox, are defined in a script and run automatically. The basic operations applied to input 3D image I are as follows:

1. Apply a 3×3×3 sigma filter to reduce noise, while preserving thin line-like structures [40].
2. Produce a raw segmented tree I_s by applying a region-growing approach.
3. Perform 3D cavity deletion to fill interior cavities in the raw segmented image. This gives image I_s consisting of solid cavity-free regions.
4. Using input image I and segmented image I_s, produce a surface representation of the tree.
5. Compute the raw central-axis structure of the segmented image, using the tree's surface representation as input. This step also produces diameter and curvature information for the tree at discrete sites along the computed central axes.
6. Compute an initial profile of quantitative information for the raw tree.

First, we have applied this basic procedure to cardiac, liver, and lung images. The one difference in the scripts for these different applications has been the choice of image segmentation algorithm for lung-image analysis.

For image segmentation, we have employed either symmetric region growing (SymRG) or a refined adaptive full-width half-maximum (FWHM) SymRG method for micro-CT image segmentation. SymRG is a form of seeded region growing that is insensitive to the locations of the initial region-growing points and that admits a memory- and computation-efficient implementation [4,41]. Key to region growing's effectiveness are the inclusion criteria used for adding new region points during the growing process. Previously proposed work using the SymRG paradigm for segmenting anatomical trees depended on ad hoc selection of a global threshold range [4]. This resulted in trial-and-error repeated runs of region growing and, ultimately, in conservative tree segmentations missing many valid branches. The adaptive FWHM SymRG method overcomes these difficulties, giving a method that does not require user trial-and-error to find adequate parameters and that is able to extract many missed branches. We briefly highlight the method below; complete detail appears in [23]. First, a locally adaptive half-maximum image I_{HM} is computed:

$$I_{HM}(x,y,z) = \text{median}\left(I(x,y,z),\sigma\right) \approx I * G_\sigma(x,y,z) \tag{1}$$

where median$(I(x,y,z),\sigma)$ represents a $\sigma \times \sigma \times \sigma$ median-filtered version of input image I. The far right-side of (1) gives a quantity that is far easier to compute than the median and represents I filtered by a $\sigma \times \sigma \times \sigma$ Gaussian $G_\sigma(\cdot,\cdot,\cdot)$ ("*" represents convolution). For each image point (x,y,z), (1) gives the local half-maximum value $I_{HM}(x,y,z)$ within a $\sigma\times\sigma\times\sigma$ neighborhood about (x,y,z). In our work, we have assumed σ denotes the nominal standard deviation of the imaging scanner's point-spread function (PSF), where the PSF is approximated by a Gaussian distribution. The half-maximum value in a local neighborhood has commonly been used by others for detecting boundaries between bright and dark regions. Using the adaptive half-maximum function I_{HM}, the next step is to create a threshold image:

$$I_{TH}(x,y,z) = \begin{cases} I_{min}, & I_{HM}(x,y,z) < I_{min}(x,y,z) \\ I_{HM}(x,y,z), & \text{if } I_{min}(x,y,z) \le I_{HM}(x,y,z) \le I_{max}(x,y,z) \\ I_{max}, & I_{HM}(x,y,z) > I_{max}(x,y,z) \end{cases} \tag{2}$$

In (2), I_{min} is either a global threshold value above which all branches are assumed to lie or it is a locally varying quantity that can be found, for example, by applying a gray-scale morphological opening $I_{min} = ((I \ominus B) \oplus B)$, where $\ominus$ is morphological erosion, $\oplus$ is morphological dilation, and structuring element B has a size sufficiently large to encompass any expected valid branches. I_{max} in (2) is selected to avoid segmenting out spurious extreme bright artifacts (such as metal clips) in the background. The threshold image I_{TH} then serves as the 3D image input to standard SymRG. Relative to the original 3D gray-scale I for the anatomical tree, I_{TH} has four attributes making it much better suited for segmenting a tree: (1) it greatly accentuates the local differences between true tree branches and background; (2) it is relatively insensitive to the absolute brightness of a branch; (3) it better separates potentially merging weaker branches: and, most importantly, (4) it removes the need for trial-and-error parameter selection in segmenting the desired branches. Figure 8 illustrates the effectiveness of adaptive FWHM SymRG over standard SymRG.
FIGURE 8 here.

Note that the raw segmented tree in I_s after step 2 may have interior cavities. This is topologically incorrect and greatly affects the resulting central-axes analysis. Hence, well-known 3D cavity deletion is applied to fill cavities in I_s [33, 41]. A cavity manifests itself in segmented image I_s as a set of 0 voxels completely surrounded in 3D by 1 voxels (voxels that belong to a segmented tree). A cavity is easily identified by locating 3D connected components of 0 voxels that do not touch the outer border of the image.

Given I and cavity-filled I_s, a surface representation of the segmented tree is next generated. The surface representation is a series of triangles covering the exterior surface of the tree and is used for later surface rendering. It is produced from a gray-scale mask image I_M using the standard Marching-Cubes algorithm implemented in VTK [25] (threshold value 0). The gray-scale mask I_M

is formed via

$$I_M = (I_S \oplus B - I_S \ominus B) \times (I - I_{TH})$$

where B is a $3 \times 3 \times 3$ structuring element, and I_{TH} is the threshold image (2) derived during image segmentation.

Stage-2 centerline analysis is next performed. Given the importance of this stage — it is the stage that actually produces the first quantitative description of a tree, we designate a separate stage for it. Before proceeding, we first introduce some notation. Define a *tree* $\mathbf{T}$ as consisting of a set of *3D sites* $\mathbf{V}$ and a set of *branches* $\mathbf{B}$:

$$\mathbf{T} = \{\mathbf{V},\ \mathbf{B}\}$$

The two sets constituting $\mathbf{T}$ are given by $\mathbf{V} = \{v_1, v_2, \ldots, v_K\}$ and $\mathbf{B} = \{b_1, b_2, \ldots, b_J\}$, where K and J are integers ≥ 1, each $v_k \in \mathbf{V}$ is a 3D site along one of tree $\mathbf{T}$'s central axes, and each $b_j \in \mathbf{B}$ is one of the central-axis branches of $\mathbf{T}$. A particular branch $b \in \mathbf{B}$ is defined by a set of contiguous 3D sites $v_b(k) \in \mathbf{V}$:

$$b = \{v_b(1), v_b(2), \ldots, v_b(E_b)\}$$

where $v_b(E_b)$ is the terminating, or end, site of branch b, integer $E_b \geq 2$ (a branch must consist of ≥ 2 3D sites), $v_b(k) \in \mathbf{V}(b)$, and $\mathbf{V}(b) \subset \mathbf{V}$ denotes a subset of 3D sites constituting branch b. All viewing sites $v \in \mathbf{V}$ belong to one and only one branch, except possibly those sites that begin and terminate branches. For example, if branch b_2 is a child branch of b_1, then

$$v_{b_2}(1) = v_{b_1}(E_{b_1}). \tag{3}$$

3D sites that do not begin or terminate a branch will be referred to as *interior* sites. 3D sites such as $v_{b_1}(E_{b_1})$ in (3) will be referred to as *branch*, or *bifurcation*, points. In general a given 3D image of interest might contain one or more trees, $\mathbf{T}_1$, $\mathbf{T}_2$, etc. These multiple trees might be valid trees — e.g., a venous and arterial tree — or they might correspond to broken trees or artifacts, produced because of difficulties in processing a given image.

In this work, we have previously used the centerline methods of Wan *et al.* and Kiraly *et al.* [10, 42]. Wan's method is based on 3D thinning and provides branches based on integer coordinates; it unfortunately gives many false branches and poorly centered branches. The method of Kiraly partially alleviates these issues by eliminating many false branches and defining 3D sites on smooth B-spline curves. Both of these approaches base their calculation of the centerlines off of the segmented image I_s. However, our current work now uses the superior approach of Yu, which bases its centerline analysis off of the more precise surface representation [21, 23]. This method, built on a differential geometry analysis of a tree's surface's curvature, gives a more robust, better-centered set of branches, with fewer false branches, better defined bifurcation regions, and the facility to separate branches that appear merged in I_s.

Figure 9 compares the raw centerlines produced by four methods at a complex junction: the three methods mentioned above and Analyze/AVW^{TM}, a well-known system for 3D medical image analysis and visualization. The central axes generated by Analyze/AVW^{TM} are not smooth, because this system represents 3D points in integer format. The Analyze/AVW tool also produces many spurious small branches [43]. While the method of Kiraly *et al.* generates substantially smoother results and fewer false branches than Wan *et al.*, it does not define central-axes structure near branch junctions plausibly. Further, it often does not center the extracted axes properly and is not able to separate branches if they touch in the input segmented image [23].
FIGURE 9 here.

After centerline analysis, a preliminary quantitative profile of the extracted raw tree structure is produced. More detail appears on these calculations in Section 4.3. To conclude, the final outputs after Stage 1 and 2 automated analysis are the surface data for all extracted trees, and central-axes sturctures $\mathbf{T}_1 = \{\mathbf{V}_1, \mathbf{B}_1\}$, $\mathbf{T}_2 = \{\mathbf{V}_2, \mathbf{B}_2\}$, $\mathbf{T}_3 = \{\mathbf{V}_3, \mathbf{B}_3\}, \ldots$ Our work to date has focused on defining one complete tree. As later results will show, however, certain trees appearing in I_s may in fact be "broken" trees that need to be reconnected to form a complete main tree.

4.2 Stage 3: Identify and Correct Tree Errors

The goal now is to convert a given imperfect tree structure, as shown in Fig. 1b, into a "correct" tree as shown in Fig. 1a. This is done by editing the tree structure. The basic process for editing a given set of raw tree data involves two phases:

1. Run the Tree Diagnostician to identify a set of possible tree defects.
2. Use various visualization tools in conjunction with a set of semi-automated and manual tree-editing functions to correct the tree.

4.2.1 Tree Diagnostician

After the raw tree is extracted and displayed using any of the tools mentioned in Section 3.3, the Tree Diagnostician identifies candidate tree defects, including broken branches, breaks between separate (but possible false) trees, overly short end branches, and others from the raw extracted central axes. An example of the Tree Diagnostician and its associated Defect List dialogue box is shown in the lower right portion of Fig. 3.

The Tree Diagnostician identifies possible defects in a set of raw trees $\mathbf{T}_1, \mathbf{T}_2, \ldots,$ extracted from input image I by performing simple distance and connectivity calculations on the 3D sites and branches constituting the trees. We give the criteria for possible defects below (all bold-faced defect names are adapted from the Tree Diagnostician example given in Figure 3):

Break: Two branches, b_1 and b_2, from any extracted tree are said to have a "break" between them – i.e., the two branches actually might constitute a single connected branch — if any pair of 3D sites belonging to the branches are within a preset minimum distance d_1 from each other:

$$||v_{b_1}(l) - v_{b_2}(m)|| \leq d_1 \tag{4}$$

where $v_{b_1}(l) \in \mathbf{V}(b_1)$ and $v_{b_2}(m) \in \mathbf{V}(b_2)$. The 3D sites considered in this calculation can be restricted to beginning and terminating ("end") branch sites (**End2EndPt** option in Figure 3) or to combinations of end sites in one branch and/or interior sites in the other (**End2Interior** or **Int2Interior** options in Figure 3).

Break2trees: Two trees, $\mathbf{T}_1 = \{\mathbf{V}_1,\ \mathbf{B}_1\}$ and $\mathbf{T}_2 = \{\mathbf{V}_2,\ \mathbf{B}_2\}$, are said to have a break between them — i.e., they actually might constitute one tree — if any pair of 3D sites belonging to the trees are within a preset minimum distance d_2 from each other:

$$||v_1(l) - v_2(m)|| \leq d_2 \tag{5}$$

where $v_1(l) \in \mathbf{V}_1$, $v_2(m) \in \mathbf{V}_2$, $l = 1, 2, \ldots, K_1$, and $m = 1, 2, \ldots, K_2$. As above, restrictions can be placed on which 3D sites are considered in this analysis.

EndBranch: A terminating branch b, which has no child branches, is deemed to be too short and possibly an artifact if

$$L(b) \leq d_3 \tag{6}$$

where

$$L(b) = E_b - 1$$

denotes the length of branch b and d_3 signifies a minimum branch length.

Tree Size: A tree $\mathbf{T} = \{\mathbf{V}, \mathbf{B}\}$ contains branches having an insufficient total branch length:

$$\sum_{b \in \mathbf{B}} L(b) \leq d_4 \tag{7}$$

where d_4 is a parameter specifying the minimum acceptable total length.

Close Bifurcation: If branch b_2 is a child branch of b_1 and b_2 also spawns child branches, then it is possible that the bifurcation points produced by branches b_1 and b_2 may be overly close. This implies that a close — and possibly false — bifurcation exists. Close bifurcations can be identified via the criterion:

$$||v_{b_1}(E_{b_1}) - v_{b_2}(E_{b_2})|| \leq d_5 \tag{8}$$

where d_5 is a parameter. Such bifurcations occur, because image noise results in false branches being formed. They also sometimes occur at junctions where trifurcations, or even higher x-furcations, actually exist, but the necessary image-sampling process produces false bifurcations. A false branch that produces a false bifurcation will cause all further generational labels below the offending branch to be falsely labeled. Thus, it is vital to identify such locations.

For this paper as shown in Figure 3, we used the following parameter values for criteria (4-8): $d_1 = 2,\ d_2 = 5,\ d_3 = 1,\ d_4 = 1,\ d_5 = 1.5$. These values essentially correspond to the minimum values that can define valid breaks, valid disconnected trees, etc. Using larger values would result in the Tree Diagnostician flagging fewer potential defects.

In addition two additional possible defects related to the topology of the tree are identified automatically by the Tree Diagnostician:

Loop: If any subset of 3D sites in a tree form a closed loop, then a constituent loop branch is flagged. (The 2D Tree Map also flags all branches constituting a loop.)

x-furcation: A branch $b \in \mathbf{T}$ is flagged as creating a possibly false x-furcation if its terminating site $v_b(E_b)$ spawns three or more child branches; i.e., for branches $b_1, b_2, \ldots, b_m \in \mathbf{T}$, $m \geq 3$,

$$v_{b_1}(1) = v_{b_2}(1) = \ldots = v_{b_m}(1) = v_b(E_b)$$

Generally, a branching anatomical tree is defined by a bifurcating structure; i.e., all parent branches have only two child branches. Sometimes, trifurcations and higher-order x-furcations can be valid. But image artifacts and undersampling can produce false higher-order branch points. Hence, x-furcations are flagged.

4.2.2 Tree Editing

After identifying potential tree defects using the Tree Diagnostician, the user then considers each candidate defect. When the user points to a defect on the defect list using the mouse, the system's global status (view state) changes to this location and all visualization tools are updated to reflect this selection. The user can then apply various manual and semi-automatic tree-editing functions to fix the defect. As defects are fixed, they are removed from the Tree Diagnostician's Defect List (see Figure 3). In addition, the quantitative representation can be updated as the defects are fixed. At any time during a session, the user can save the case study and resume editing at another time.

The tree-editing functions include several means for breaking and deleted unwanted branches and for creating new tree branches. For all functions, the user can employ any of the interaction modes discussed in Section 3.3.1. The user can manually place 3D sites at location of interest and then the sites are joined via B-spline analysis to form a new curved line segment. Figure 10 illustrates this capability. As another approach, the repair of a broken branch can be done by using this manual method in conjunction with the 3D Bounding Box [20]. As a third option, a semi-automatic function exists for creating new branches. Similar to Figure 10, the user first interactively selects two endpoints for joining. Then, automatic Hermite interpolation is applied to form a curved segment between the two points. Unlike B-spline analysis, a curve interpolated via Hermite interpolation only depends on the two endpoints [44]. All of these functions are useful for fixing broken branches and for joining separated trees.

Another manual tree-editing function involves the facility for breaking and then subsequently deleting existing branches. Again using either the shooter or 3D cursor, the user selects 3D sites on a branch, which are then deleted — this breaks the branch. The system can then delete the remnants of the branch to complete the operation. This function is also useful for deleting unwanted portions of a tree and for breaking loops. As shown in an example in the next section, it sometimes becomes necessary to redefine the root of a tree. A facility exists for doing this interactively. The user selects a desired branch to serve as the new root branch, and the currently active quantitative description of the tree is recomputed to reflect this choice.

Doing purely interactive defect correction, however, can be tedious. Thus, various semi-automatic tools are also available as invocable functions to speed up the refinement of a raw tree. These functions include the following:

Branch deletion: Individual branches are selected and automatically deleted.

Prune selected branch and below: The user selects a tree branch. Then, the branch and all of its connected ancestor branches are automatically pruned from the tree. For example, if branch (3,2) is selected, then branches (4,3), (4,4), (5,5), (5,6), (5,7), (5,8), etc., are deleted.

Delete tree(s): A disconnected small artifact tree is selected and automatically deleted. Or all trees except a selected tree are deleted.

Generation-based deletion: All branches below a prescribed generation number are deleted.

Prune short branches: All terminating branches shorter than a specified length $L(b)$ are deleted.

Line-based and sphere-based elimination: Based on geometric criteria defined fully in the central-axes analysis method of Kiraly *et al.* [42]: (1) line-based elimination — branch b_1 connected to branch b_2 is pruned if its length straddles the extent of b_2 too closely; and (2) sphere-based elimination — branch b_1 is pruned if it intersects a maximally inscribable sphere centered about the terminating site of another branch.

These functions are invoked by interacting with a global or local 3D Render tool.

The 2D Tree Map enables the pruning of undesired branches or subtrees by pointing to them. Also, Figure 11 gives an example of correcting a tree loop using the 2D Tree Map and other tools. In addition, a tree's root branch can be redefined by pointing to a desired branch. After redefining the root branch, all branch labels are recomputed to reflect the new order. Redefinition of the root becomes necessary when two anatomically separate trees initially appear connected when they should be considered separately (e.g., venous and arterial trees appear connected); after the two trees are separated, at least one of them will need to have their true root branch defined.
FIGURES 10 and 11 here.

As edits are made, the Tree Analyzer gives an updated tree structure and associated quantitative data. The user can decide to undo changes or keep changes. When changes are kept, the stored tree structure is updated to reflect the corrections.

4.3 Stage 4: Compute Quantitative Tree Measurements

A quantitative profile can be computed at anytime after a tree description is generated (after Stage 2). Quantitative summaries at the generation, branch, and 3D site level are available. They can be popped up as dialog boxes in the main Tree Analyzer GUI or saved as ASCII text files. The Results section gives examples.

The quantitative measurements computed by the Tree Analyzer are the same as those proposed by Wan *et al.* [10]. A few major differences are pointed out below. Paralleling Wan *et al.*, the following definitions illustrate some of the quantitative data that can be derived from a tree representation **T**:

$c_{j,k}$	k^{th} cross-section of branch b_j (one at each 3D site $v_{b_j}(k)$)
$L(b_j)$	length of branch b_j
$\mu_A(b_j)$	average cross-sectional area (CSA) of branch b_j
g_i	i^{th} tree generation
$N^b(g_i)$	number of branches in generation g_i
$\mu_L(g_i)$	average length of branches at generation g_i
$\mu_A(g_i)$	average CSA of of branches at generation g_i

Other measurements, not discussed here, can also be computed related to surface area, diameter, and volume [10, 23]. Wan based all measurements off of the segmentation I_s. This tends to give a conservative and less-precise structure for basing quantitative measurements. In our current work, all values are derived from the more precise surface representation of a tree. Further, Wan approximated branches as a sequence of connected line segments, each of length 1 voxel. Our current effort imposes no such restriction on the spacing of 3D sites v. In addition 3D sites are situated at sub-voxel spacing along curves defined by either a B-spline of Hermite polynomial.

5 RESULTS

This section presents a variety of results illustrating the effectiveness of various aspects of the Tree Analyzer. Other results using the Tree Analyzer appear in [20, 21, 23, 24].

Fig. 12 shows output for micro-CT image h61. The image depicts a cast of a mouse coronary arterial tree, where the cast was attached to the lid of a jar using clay (the rendered mass at the top of a rendering is the clay). Figure 12a displays the result using the previously proposed axial-extraction method by Kiraly [42], while Figure 12b presents the corresponding result using Yu's centerline method. The clay situated at the top of the scan originally produced a mass of invalid branches around the tree root after Stage 2 for both centerline methods. Thus, Figures 12a-b actually depict the two trees after simple tree editing was performed in the vicinity of the root to delete the mass of distracting branches. This editing required (1) breaking each valid tree from the mass and (2) deleting the disconnected mass of branches. These simple operations required approximately one minute of interaction to produce the results of Figures 12a-b. In both cases, two valid disconnected trees were produced.

Using the Tree Diagnostician, no loops were found using Kiraly's method, since the method automatically breaks loops. But three loops existing in the segmentation were improperly broken by the method, and an obvious branch was missing (Figure 12a). Tree editing on this result required 20 minutes of interaction, as the improperly broken loops greatly affected the raw tree. The Stage-2 result of Yu, on the other hand, resulted in a much better starting point for tree editing (Figures 12b and 13a). Three loops were detected in this result. The tree editing time, which involved breaking the loops and joining the two separate trees, required only 5 minutes. Note that no quantitative information is available for the root-node region created for the final edited global tree, because of the clay. This resulted in an ill-defined root region that produced many defects identified by the Tree Diagnostician. We cut the tree in this region, as stated above, and then connected the two "valid" sub-trees; this then gave a new root and generation-1 set of branches.

Quantitative measurements were computed for h61, before and after tree editing, as shown in Figure 14. The tables show that the generational description for h61's tree is far more sensible (and correct!) after tree editing than before. To verify the correctness of these numbers, Figure 15 gives a comparison of the measured branch lengths for h61 between ground-truth manual measurements made by a skilled human operator (done using a microscope) [20] and by Kiraly's method and Yu's centerline method (both after tree editing). Figure 15 shows that the Yu results have a better linear regressive slope (0.9848 versus 0.9724) and a better R-squared value (0.9855 versus 0.9718). The R-squared value, an indicator ranging from 0 to 1, reveals how closely the estimated results for the linear regressive line correspond to the actual data. Even after we manually fixed the defects in

Kiraly's raw result, the output is still more scattered due to the improper position of branch points, as shown in Fig. 15. All h61 results indicate the importance of an effective Stage-2 automated centerline analysis method. The remainder of the presented results use Yu's centerline method.
FIGURES 12-15 here.

Figures 16-17 present results for a 3D image of a rat's left circumflex arterial tree (lca_146), first used in [10]. The figures present a comparison of the results produced by Wan's 2002 system to the current Tree Analyzer. The current system introduces three major improvements over the earlier system: (1) the more robust adaptive FWHM SymRG is used for image segmentation, which extracts more branches; (2) the improved centerline method of Yu is employed, which gives smooth, better-centered branches; and (3) the new extensive Stage-3 tree-editing capabilities are used. Interactive tree editing required 17 minutes to resolve 8 branch breaks, 7 short branches, 9 close x-furcations, and 6 loops. A comparison of the final quantitative results produced by the Tree Analyzer and Wan's system (Figure 17) reveals that the current system gives a more extensive analysis that is plausible to a higher-generation index.

Figure 18a presents an example of applying the Tree Analyzer to a human 3D MDCT chest image. All Stage-1 and Stage-2 automated processing operations discussed in Section 4.1 were used for this example, with the exception that a segmentation method tailored for airway trees was used [45]. After Stage-1 and Stage-2 analysis, two minutes of tree editing corrected 1 short branch, 1 loop, and 2 close bifurcations. The final tree contained one plausible trifurcation.
FIGURES 16-17 here.

Figure 18b depicts results, free of defects, for a micro-CT image of the rat-liver vasculature. This example depicts a tree that nominally consists of 75 generations. Unlike the standard bifurcating patterns of the airway tree and coronary vasculature, the liver's vascular tree exhibits a mode-3 branching pattern, as discussed by Zamir, in that a main trunk extends down the length of the organ, and smaller branch generations are spawned off the trunk [46]. Thus, branches at high-generation levels have more in common in terms of their quantitative and mechanical properties than certain branches at their corresponding generation level. While the Tree Analyzer has considerable capability for processing, visualizing, and editing such trees, further work is necessary to arrive at a more proper and useful quantitative description for such an anatomical tree.
FIGURE 18 here.

6 DISCUSSION

The Tree Analyzer is a complex system containing a large variety of tools for general 3D automated analysis, 3D visualization, data mining, and quantitative analysis. It is particularly suited to analyzing images containing large branching anatomical trees. Our main focus has been aimed at analyzing the vascular structure of organs imaged by a micro-CT scanner, but we have also applied the Tree Analyzer to 3D MDCT images of the chest and airway tree. In the interactive part of the system, we have introduced many functions that allow the user to efficiently diagnose and repair various problems in raw extracted trees. The system also provides a series of interactive editing tools that not only help solve problems, but also provide a more reliable tree structure. Semi-automatic tools have also been devised to support the inadequacies of purely interactive methods. This makes it possible to generate more trustworthy trees and associated quantitative measurements.

The capabilities of the system are extremely extensive and cannot all be described here. We refer the reader to the references for further information [20–24]. Note that the Tree Analyzer has considerable general capability that makes it potentially useful for other 3D medical imaging problems. Further work can be done to apply the system to the separation of venous and arterial trees [5]. In addition, effort can be made to propose more suitable representations for trees, such as the liver's vasculature, that do not abide by a standard bifurcating structure.

7 ACKNOWLEDGEMENTS

This work was partially supported by grants #EB000305, #CA074325 and #CA091534 from the NIH. Early versions of some of this work were presented at the conferences [20, 21, 24]. We also thank Mike Graham for generating the matching results for Figure 15 and Diane Eaker for providing assistance with some of the micro-CT data.

References

[1] S. M. Jorgensen, O. Demirkaya, and E. L. Ritman, "Three dimensional imaging of vasculature and parenchyma in intact rodent organs with x–ray micro–CT," *Am J. Physiol (Heart, Circ Physiol 44)*, vol. 275, pp. H1103–H1114, 1998.

[2] A. Garcia-Sanz, A. Rodriguez-Barbero, M. D. Bentley, E. L. Ritman, and J. C. Romero, "Three-dimensional microcomputed tomography of renal vasculature in rats," *Hypertension*, vol. 31, no. Part 2, pp. 440–444, 1998.

[3] R. H. Johnson, H. Hu, S. T. Haworth, P. S. Cho, C. A. Dawson, and J. H. Linehan, "Feldkamp and circle-and-line conebeam reconstruction for 3D micro-CT of vascular networks," *Physics in Medicine and Biology*, vol. 43, no. 4, pp. 929–940, 1998.

[4] S. Y. Wan, A. P. Kiraly, E. L. Ritman, and W. E. Higgins, "Extraction of the hepatic vasculature in rats using 3D micro-CT images," *IEEE Transactions on Medical Imaging*, vol. 19, no. 9, pp. 964–971, Sept. 2000.

[5] M. Kretowski, Y. Rolland, J. Bezy-Wendling, and J.-L. Coatrieux, "Physiologically based modeling of 3-D vascular networks and CT scan angiography," *IEEE Transactions on Medical Imaging*, vol. 22, no. 2, pp. 248–257, Feb. 2003.

[6] W. Kalender, *Computed Tomography: Fundamentals, System Technology, Image Quality, Applications*, Publicis MCD Verlag, Munich, 2000.

[7] N. C. Dalrymple, S. R. Prasad, M. W. Freckleton, and K. N. Chintapalli, "Introduction to the language of three-dimensional imaging with multidetector CT," *Radiographics*, vol. 25, no. 5, pp. 1409–1428, Sept.-Oct. 2005.

[8] F. K. H. Quek and C. Kirbas, "Vessel extraction in medical images by wave-propagation and traceback," *IEEE Transactions on Medical Imaging*, vol. 20, no. 2, pp. 117–131, Feb. 2001.

[9] D. Selle, P. Preim, A. Schenk, and H.-O. Peitgen, "Analysis of vasculature for liver surgical planning," *IEEE Transactions on Medical Imaging*, vol. 21, no. 11, pp. 1344–1357, Nov. 2002.

[10] S. Wan, E. Ritman, and W. Higgins, "Multi-generational analysis and visualization of the vascular tree in 3D micro-CT images," *Computers in Biology and Medicine*, vol. 32, no. 2, pp. 55–71, Feb 2002.

[11] E. A. Kazerooni, "High resolution CT of the lungs," *Am. J. Roentgenology*, , no. 3, pp. 501–519, Sept. 2001.

[12] J. Williams and L. Wolff, "Analysis of the pulmonary vascular tree using differential geometry based vector vields," *Computer Vision and Image Understanding*, vol. 65, no. 2, pp. 226–236, Feb. 1997.

[13] C. C. Hanger, S. T. Haworth, R. A. Molthen, and C. A. Dawson, "Semi-automated skeletonization of the pulmonary arterial tree in micro-CT images," *SPIE Medical Imaging 2001: Physiology and Funct. from Multidim. Images, A. Clough and C. T. Chen, eds.*, vol. 4321, pp. 510–516, Feb. 18-20, 2001.

[14] L. Antiga, B. Ene-Iordache, and A. Remuzzi, "Computational geometry for patient-specific reconstruction and meshing of blood vessels from MR and CT angiography," *IEEE Transactions on Medical Imaging*, vol. 22, no. 5, pp. 674–684, May 2003.

[15] I. Volkau, W. Zheng, R. Baimouratov, A. Aziz, and W. L. Nowinski, "Geometric modeling of the human normal cerebral arterial system," *IEEE Transactions on Medical Imaging*, vol. 24, no. 4, pp. 529–539, April 2005.

[16] R. H. Johnson, K. L. Karau, R. C. Molthen, , and C. A. Dawson, "Exploiting self–similarity of arterial tree branches to reduce the complexity of analysis," *SPIE Medical Imaging 1999: Physiology and Function from Multidimensional Images*, vol. 3660, pp. 351–361, C. T. Chen and A. V. Clough (ed.), 1999.

[17] R. H. Johnson, K. L. Karau, R. C. Molthen, S. T. Haworth, and C. A. Dawson, "Micro-ct image-derived metrics to quantify arterial wall distensibility reduction in a rat model of pulmonary hypertension," in *Proceedings of SPIE Medical Imaging 2000: Physiology and Function from Multidimensional Images*, A. Clough and C.T Chen, eds., 2000, vol. 3978.

[18] K. L. Karau, R. C. Molthen, R. H. Johnson, A. H. Dhyani, S. T. Haworth, and C. A. Dawson, "Pulmonary arterial remodeling revealed by microfocal X-ray tomography," *SPIE Medical Imaging 2001: Physiology and Function from Multidimensional Images*, vol. 4321, pp. 18–20, Feb. 2001.

[19] E. Bullitt, S. Aylward, K. Smith, S. Mukherji, M. Jiroutek, and K. Muller, "Symbolic description of intracerebral vessels segmented from magnetic resonance angiograms and evaluation by comparison with X-ray angiograms," *Medical Image Analysis*, vol. 5, pp. 157–169, 2001.

[20] K. C. Yu, E. L. Ritman, and W. E. Higgins, "Toward reliable multi-generational analysis of anatomical trees in 3D high-resolution CT images," *SPIE Medical Imaging 2003: Physiology and Function — Methods, Systems, and Applications*, vol. 5031, pp. 178–186, 2003.

[21] K.C. Yu, E. L. Ritman, and W. E. Higgins, "Graphical tools for accurate definition of 3D arterial trees," in *SPIE Medical Imaging 2004: Physiology, Function, and Structure from Medical Images*, A. Amini and A. Manduca, eds., 2004, vol. 5369, pp. 485–495.

[22] K.C. Yu, E. L. Ritman, and W. E. Higgins, "3D model-based vasculature analysis using differential geometry," in *IEEE Int. Symp. on Biomedical Imaging*, Arlington, VA, 15-18 April 2004, pp. 177–180.

[23] K.C. Yu, *Multi-Generational Analysis of Anatomical Trees in High-Resolution 3D Images*, Ph.D. thesis, The Pennsylvania State University, 2005.

[24] K. C. Yu, E. L. Ritman, and W. E. Higgins, "System for 3D visualization and data mining of large vascular trees," in *SPIE Optics East 2005: Three-Dimensional TV, Video, and Display IV*, B. Javadi, F. Okano, and J. Son, Eds., 2005, vol. 6016, pp. 60160B–1 — 60160B–15.

[25] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit, 2nd. Ed.*, Prentice Hall, Upper Saddle River, New Jersey, 1998.

[26] R.S. Wright, Jr., and B. Lipchak, *OpenGL Super Bible, 3rd. Ed.*, SAMS Publishing, 2005.

[27] A. J. Sherbondy, A. P. Kiraly, A. L. Austin, J. P. Helferty, S. Wan, J. Z. Turlington, E. A. Hoffman, G. McLennan, and W. E. Higgins, "Virtual bronchoscopic system combining 3D CT and endoscopic video," *SPIE Medical Imaging 2000: Physiology and Function from Multidimensional Images*, vol. 3978, pp. 104–116, A. Clough and C.T. Chen, eds., 2000.

[28] G Sundaramoorthy, J D Hoford, E A Hoffman, and W E Higgins, "IMPROMPTU: A system for automatic 3D medical image analysis," *Comp Med Imaging and Graphics*, vol. 19, no. 1, pp. 131–143, Jan.-Feb. 1995.

[29] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kauffman, San Francisco, 2001.

[30] D. Keim, "Information visualization and visual data mining," *IEEE Trans. Visual. Comp. Graph.*, vol. 8, no. 1, pp. 1–8, Jan.-Mar. 2002.

[31] J.Z. Turlington and W.E. Higgins, "New techniques for efficient sliding thin-slab volume visualization," *IEEE Transactions in Medical Imaging*, vol. 20, no. 8, pp. 823–835, Aug. 2001.

[32] R. A. Robb and D. P. Hanson, "ANALYZE: A software system for biomedical image analysis," in *Proc. of the First Conference on Visualization in Biomedial Computing, Atlanta, GA*, May 1990, pp. 507–518.

[33] W E Higgins, R A Karwoski, W J T Spyra, and E L Ritman, "System for analyzing true three-dimensional angiograms," *IEEE Transactions on Medical Imaging*, vol. 15, no. 3, pp. 377–385, June 1996.

[34] K Ramaswamy and W E Higgins, "Interactive dynamic navigation for virtual endoscopy," *Computers in Biology and Medicine*, vol. 29, no. 5, pp. 303–331, Sept.-Oct. 1999.

[35] P. Rogalla, J. Van Scheltinga, and B. Hamm, *Virtual Endoscopy and Related 3D Techniques*, Springer-Verlag, Berlin, 2002.

[36] R. A. Robb, *Three-dimensional Biomedical Imaging: Principles and Practice*, VCH Publishers, New York, 1994.

[37] Chung Laung Liu, *Elements of Discrete Mathematics, 2nd Ed.*, McGraw-Hill International Editions, New York, 1985.

[38] I. Herman, G. Melancon, and M. S. Marshall, "Graph visualization and navigation in information visualization: a survey," *IEEE Trans. Visualization Comp. Graphics*, vol. 6, no. 1, pp. 24–43, Jan.-Mar. 2000.

[39] B. Duan and M. Zamir, "Pressure peaking in pulsatile flow through arterial tree structures," *Annals of Biomedical Engineering*, vol. 23, no. 6, pp. 794–803, 1995.

[40] J.S. Lee, "Digital image smoothing and the sigma filter," *Computer Vision, Graphics, and Image Processing*, vol. 24, no. 2, pp. 255–269, Nov. 1983.

[41] Shu-Yen Wan and William E. Higgins, "Symmetric region growing," *IEEE Transactions on Image Processing*, vol. 12, no. 9, pp. 1007–1015, Sep. 2003.

[42] A. P. Kiraly, J. P. Helferty, E. A. Hoffman, G. McLennan, and W. E. Higgins, "3D path planning for virtual bronchoscopy," *IEEE Transactions on Medical Imaging*, vol. 23, no. 11, pp. 1365–1379, Nov. 2004.

[43] R. A. Robb, *AVW Reference Manual*, Biomedial Imaging Resource, Mayo Foundation, Rochester, MN, 1995.

[44] D. Hearn and M. P. Baker, *Computer Graphics with OpenGL*, Pearson Prentice Hall, Upper Saddle River, NJ, 3rd. edition, 2004.

[45] A. P. Kiraly, E. A. Hoffman, G. McLennan, W. E. Higgins, and J. M. Reinhardt, "3D human airway segmentation for virtual bronchoscopy," *SPIE Medical Imaging 2002: Physiology and Funct. from Multidim. Images, A. Clough and C. T. Chen, eds.*, vol. 4683, pp. 16–29, 2002.

[46] M. Zamir, "Optimality principles in arterial branching," *J. theor. Biol.*, pp. 227–251, 1976.

[47] S. T. Witt, C. H. Riedel, M. Goessl, M. S. Chmelik, and E. L. Ritman, "Point spread function deconvolution in 3D micro-CT angiography for multiscale vascular tree separation," *SPIE Medical Imaging 2003: Visualization, Image-Guided Procedures, and Display*, vol. 5030, pp. 720–727, 2003.

[48] M. Graham and W.E. Higgins, "Globally optimal model-based matching of anatomical trees," in *SPIE Medical Imaging 2006: Image Processing*, J. M. Reinhardt and J. P. W. Pluim, Eds., 2006, vol. 6144, pp. 373–388.

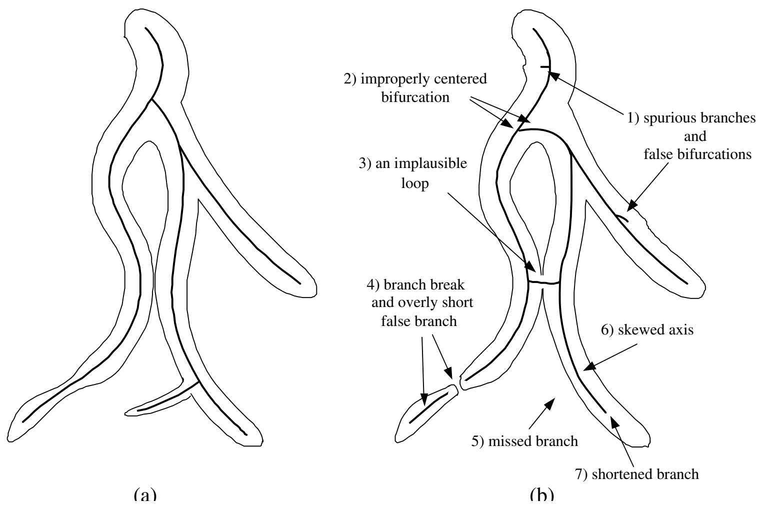

2) improperly centered bifurcation
1) spurious branches and false bifurcations
3) an implausible loop
4) branch break and overly short false branch
6) skewed axis
5) missed branch
7) shortened branch
(a)
(b)

Figure 1: 2D Schematic figure illustrating the difficulties that can arise in defining the central axes of a vascular tree depicted in a 3D image: (a) Ideal surface and central axes of tree; (b) typical output of purely automated 3D image analysis, depicting a tree with various imperfections. In both figures, the outer outline represents the tree's surface, and the interior lines represent the tree's axial structure.

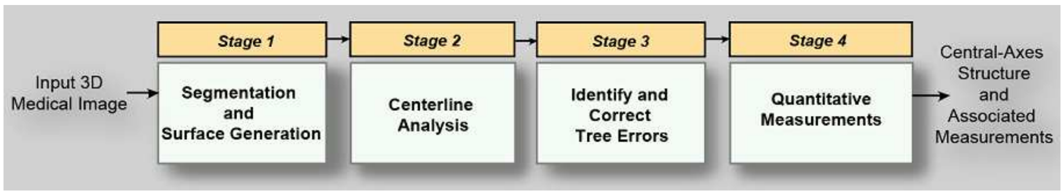
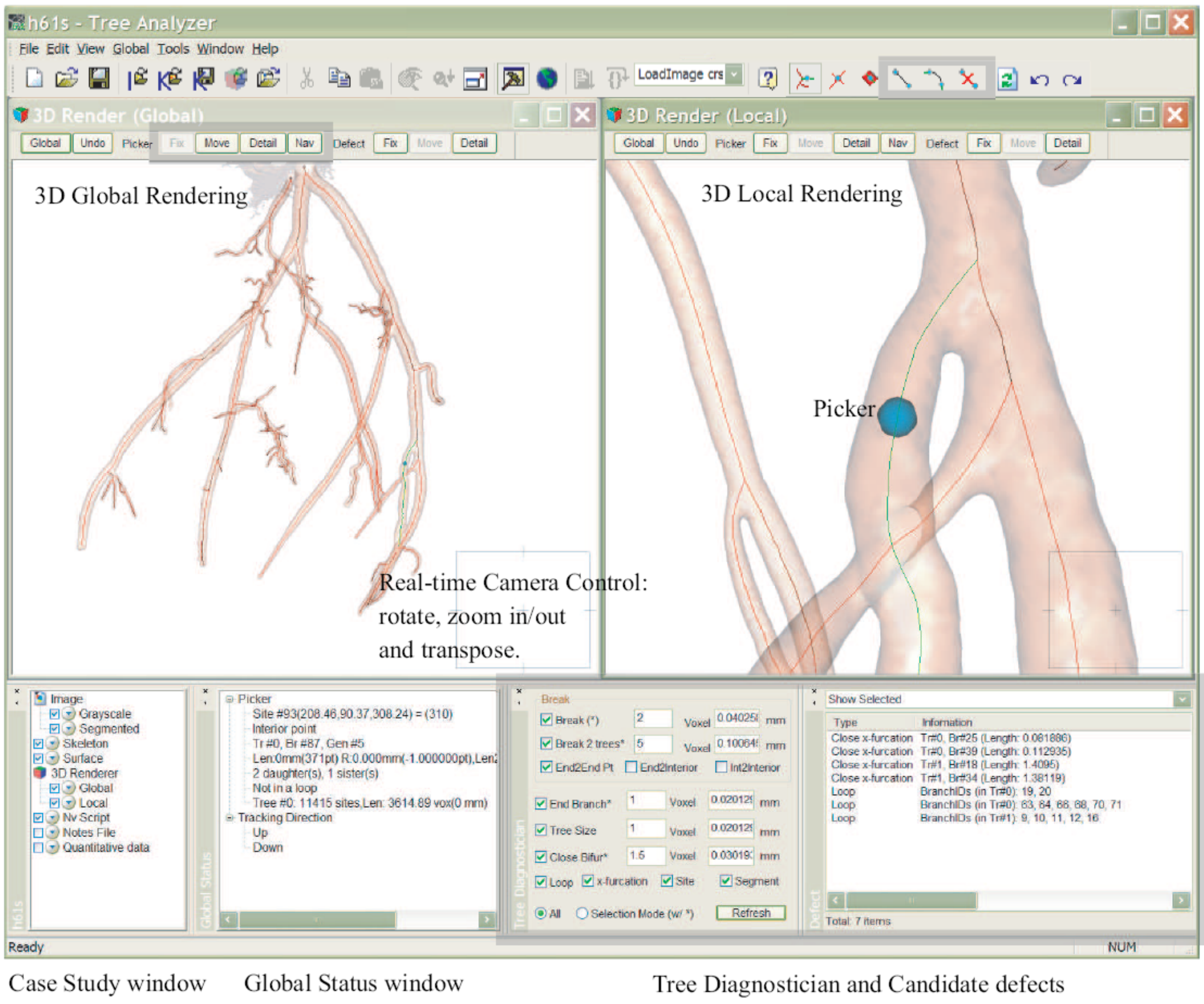

Input 3D Medical Image
Stage 1
Segmentation and Surface Generation
Stage 2
Centerline Analysis
Stage 3
Identify and Correct Tree Errors
Stage 4
Quantitative Measurements
Central-Axes Structure and Associated Measurements

Figure 2: Four-stage approach for defining the quantitative structure of a 3D anatomical tree.


h61s - Tree Analyzer
File Edit View Global Tools Window Help
LoadImage crs
3D Render (Global)
Global Undo Picker Fix Move Detail Nav Defect Fix Move Detail
3D Render (Local)
Global Undo Picker Fix Move Detail Nav Defect Fix Move Detail
3D Global Rendering
3D Local Rendering
Picker
Real-time Camera Control: rotate, zoom in/out and transpose.
h61s
Global Status
Tree Diagnostician
Defect
Ready
NUM
Case Study window
Global Status window
Tree Diagnostician and Candidate defects

```
Image
  ☑ Grayscale
  ☑ Segmented
☑ Skeleton
☑ Surface
3D Renderer
  ☑ Global
  ☑ Local
☑ Nv Script
☐ Notes File
☐ Quantitative data
```

```
⊟ Picker
   Site #93(208.46,90.37,308.24) = (310)
   Interior point
   Tr #0, Br #87, Gen #5
   Len:0mm(371pt) R:0.000mm(-1.000000pt),Len
   2 daughter(s), 1 sister(s)
   Not in a loop
   Tree #0: 11415 sites,Len: 3614.89 vox(0 mm)
⊟ Tracking Direction
   Up
   Down
```

```
Break
☑ Break (*)         2     Voxel  0.04025  mm
☑ Break 2 trees*    5     Voxel  0.10064  mm
☑ End2End Pt   ☐ End2Interior   ☐ Int2Interior

☑ End Branch*       1     Voxel  0.02012  mm
☑ Tree Size         1     Voxel  0.02012  mm
☑ Close Bifur*      1.5   Voxel  0.03019  mm
☑ Loop   ☑ x-furcation   ☑ Site   ☑ Segment
◉ All   ○ Selection Mode (w/ *)   Refresh
```

```
Show Selected
Type                Information
Close x-furcation   Tr#0, Br#25 (Length: 0.081886)
Close x-furcation   Tr#0, Br#39 (Length: 0.112935)
Close x-furcation   Tr#1, Br#18 (Length: 1.4095)
Close x-furcation   Tr#1, Br#34 (Length: 1.38119)
Loop                BranchIDs (in Tr#0): 19, 20
Loop                BranchIDs (in Tr#0): 63, 64, 66, 68, 70, 71
Loop                BranchIDs (in Tr#1): 9, 10, 11, 12, 16
Total: 7 items
```

Figure 3: Example composite view of the Tree Analyzer. The depicted surface renderings are for a tree extracted from micro-CT image h61, a 450×445×465 image of a rat coronary arterial tree with spatial resolution $\Delta x = \Delta y = \Delta z = 20.13$ μm (9.06mm × 8.96 mm × 9.36mm real volume extent), reconstructed without applying deconvolution [47]. The small box in the lower right corner of each 3D Render tool enables viewpoint control (rotation, translation, scaling). The lower portion of this composite view depicts the case study, picker status, and the Tree Diagnostician. (Used by courtesy of SPIE.)

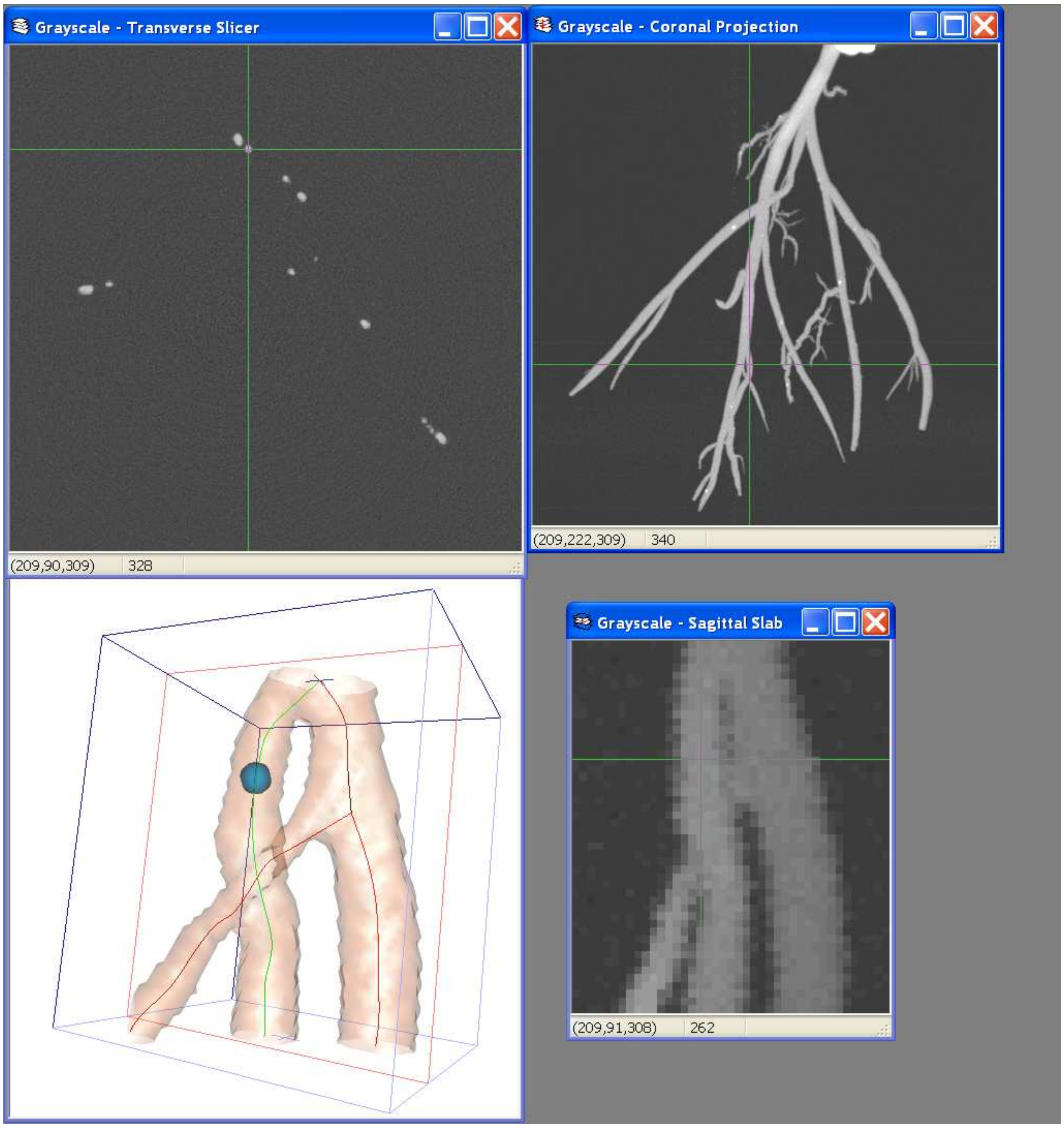

Grayscale - Transverse Slicer
(209,90,309) 328
Grayscale - Coronal Projection
(209,222,309) 340
Grayscale - Sagittal Slab
(209,91,308) 262

Figure 4: Examples of visualization tools depicted on a second monitor for the same situation as Figure 3. The top left view shows a 2D transverse slice, and the top right view is a coronal maximum-intensity projection. The lower two views illustrate the bounding box feature. The lower left view depicts a localized 3D surface rendering about a site and bounding box of interest; the bounding box is the inscribed parallelepiped shown in the figure. Bounding box dimensions: top center point = [204,81,296] and bottom center point = [197,99,339] determine the height; length and width = 20 (all dimensions in terms of voxels). The picker is at location [209,90,309]. The lower right view shows a corresponding depth-weighted maximum sagittal slab for data within the bounding box (WL=217, WW = 752, slab thickness = 40, vision = 60, view at 7.5X magnification).

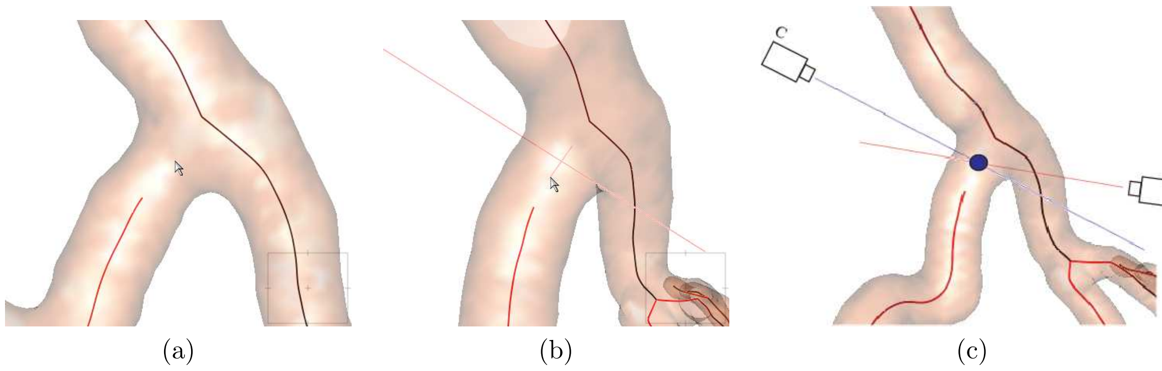


Figure 5: Example of the 3D shooter feature for selecting points in 3D space. The user points to a location on a 3D Render view (a) — this defines a 2D line in 3D Space (b). The user then rotates the rendering and points to a location on the previously defined 2D line — this specifies a 3D site, as shown by the ball (c). The two lines (and mock cameras) on the view of part (c) indicate the points of view taken during the two-step selection process.

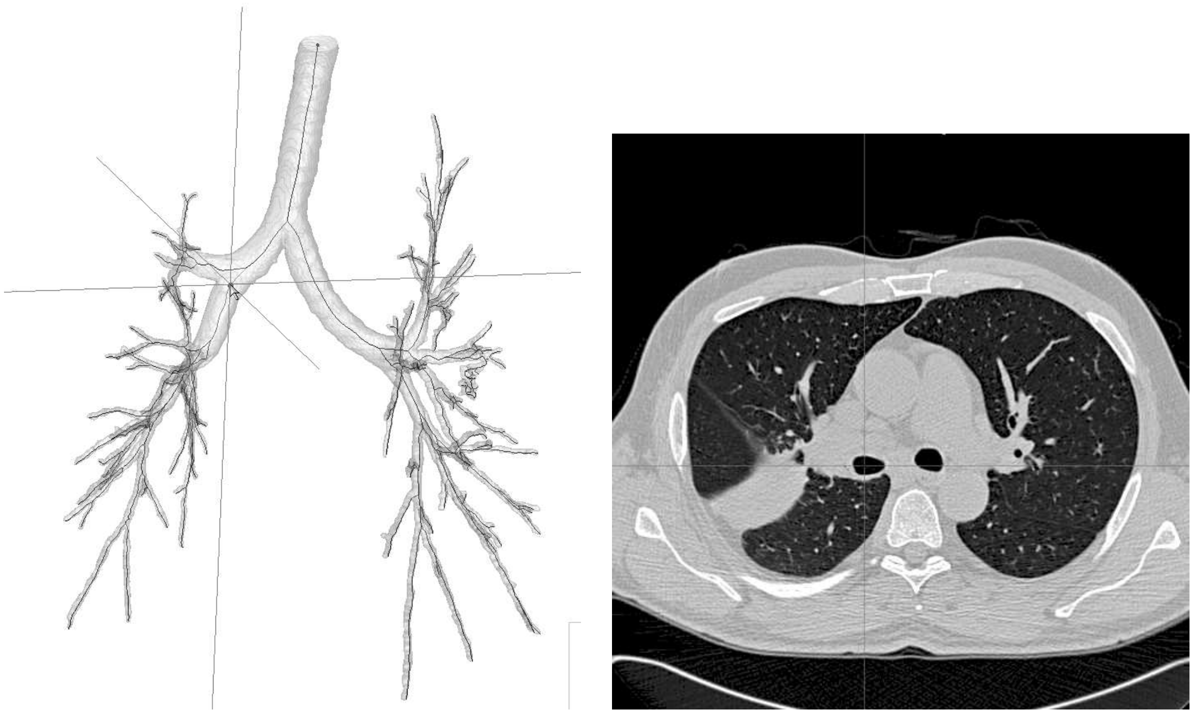
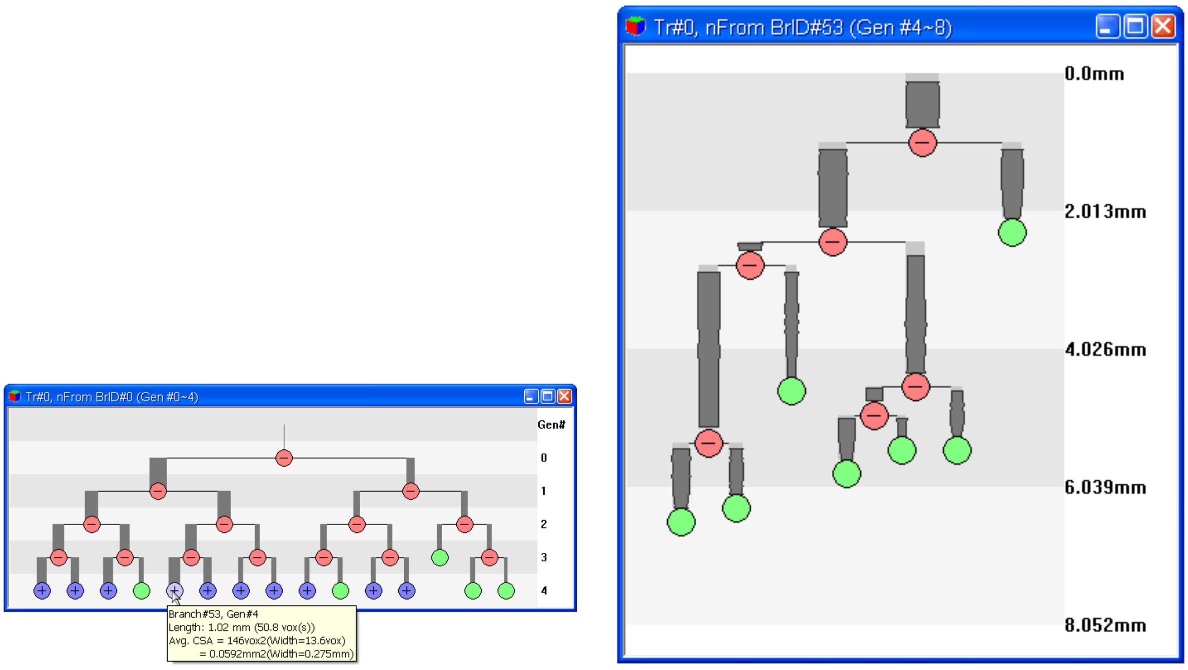


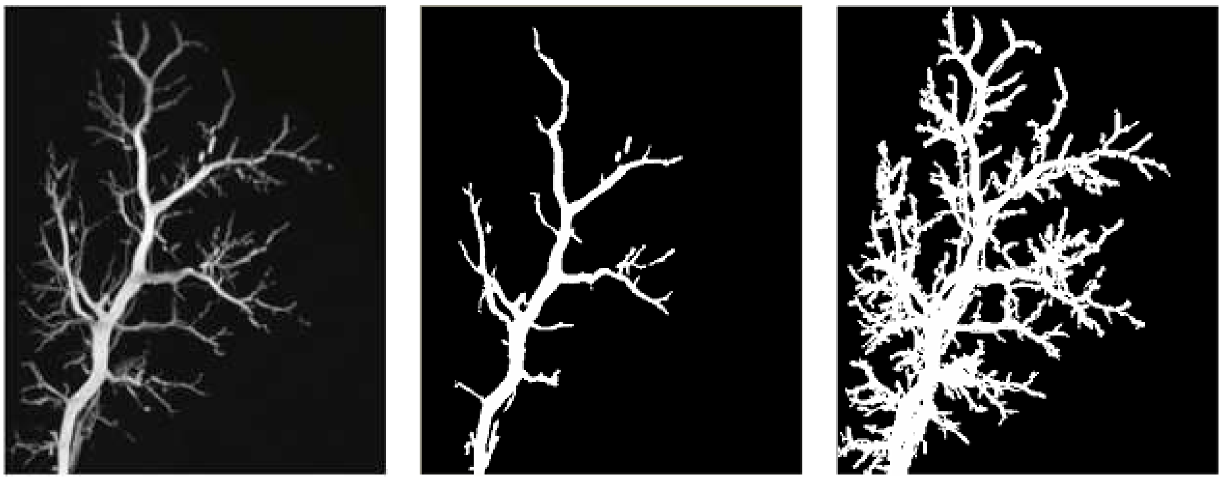
Figure 6: Example of using the 3D cursor to select a 2D slice. Human MDCT chest image h006_512_85 is considered (512×512×574 3D image, sampling intervals: Δx = Δy =0.724mm, Δz =0.60mm): (a) 3D Render view with the three axes of the 3D cursor depicted (z axis is locked); (b) 2D transverse (x-y plane) slice for indicated 3D cursor location (gray-scale window width [WW] = -350 and window level [WL] =1600 for the depicted slice [lung window]).



Branch#53, Gen#4
Length: 1.02 mm (50.8 vox(s))
Avg. CSA = 146vox2(Width=13.6vox)
= 0.0592mm2(Width=0.275mm)

(a) (b)

Figure 7: Examples of the 2D Tree Map for the tree extracted from h61: (a) Global 2D Tree Map in average-diameter mode for the first four generations; (b) local 2D Tree Map in diameter and length modes spawned from selected fourth-generation branch #53 in (a).



(a) (b) (c)

Figure 8: Sample segmentation results comparing SymRG [4, 41] and adaptive FWHM SymRG for image control2 (originally used in [4]; image of a single hepatic lobe of a rat liver, 400×400×375 image, voxel size = $(21\mu\text{m})^3$). (a) Coronal weighted-sum projection of original gray-scale data. (b) Segmentation result using SymRG. (c) Segmentation result using adaptive FWHM SymRG.

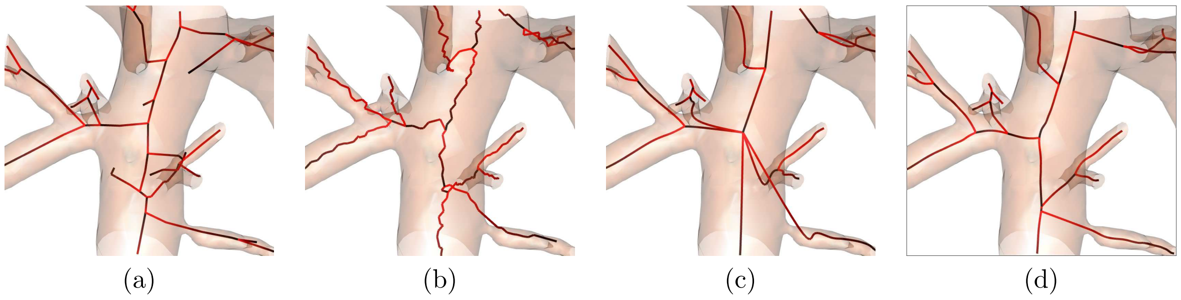


Figure 9: Stage-2 centerline analysis results for a complex junction in 3D image r216_psf020826 ([rat liver micro-CT image] dimensions 620 $\times$ 500 $\times$ 1000 [real volume dimensions: 12.57mm $\times$ 10.13mm $\times$ 20.27mm], $\Delta x = \Delta y = \Delta z = 20.27\mu$m): (a) result produced by the tree tool in Analyze/AVW^{TM} [43];(b) Wan 3D-thinning-based approach [4,10]; (c) method of Kiraly *et al.* [42]; (d) 3D centerline method of Yu and Higgins [21,23].

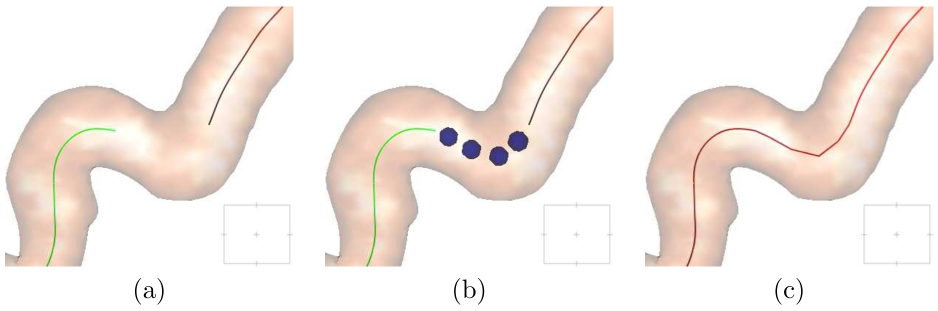
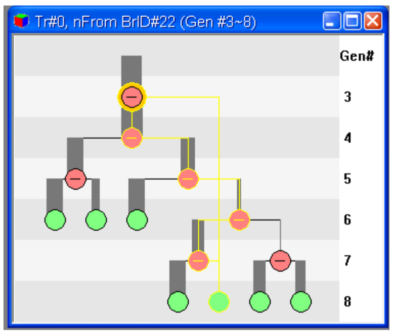
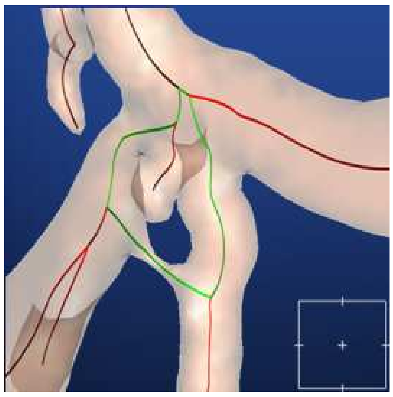


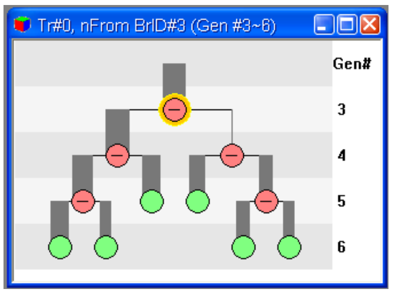
Figure 10: Connecting a broken branch using manually defined 3D points: (a) a broken branch showing a gap between two separated parts; (b) points placed manually by the user with the picker in Intersection-Locator mode — thus, the selected points are situated along the center of the segmented tree; (c) resulting connected axis after B-spline interpolation.



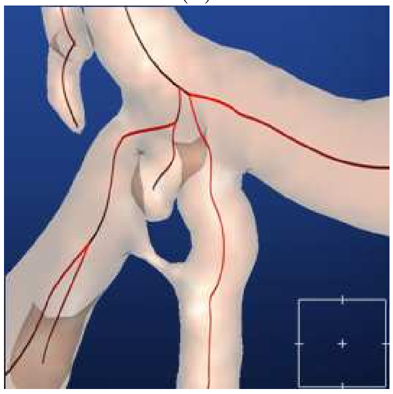
(a)



(b)



(c)



(d)

Figure 11: Fixing a 2D Tree Map section indicating a loop (h61): (a) local 2D Tree Map of a loop involving branch generations 3-7, with involved branches depicted in yellow; (b) local 3D rendering of (a), with loop indicated in green; (c) local 3D rendering after repairing the loop; (d) corresponding new local 2D Tree Map about repaired region (only generations 3-6 now exist about the region).

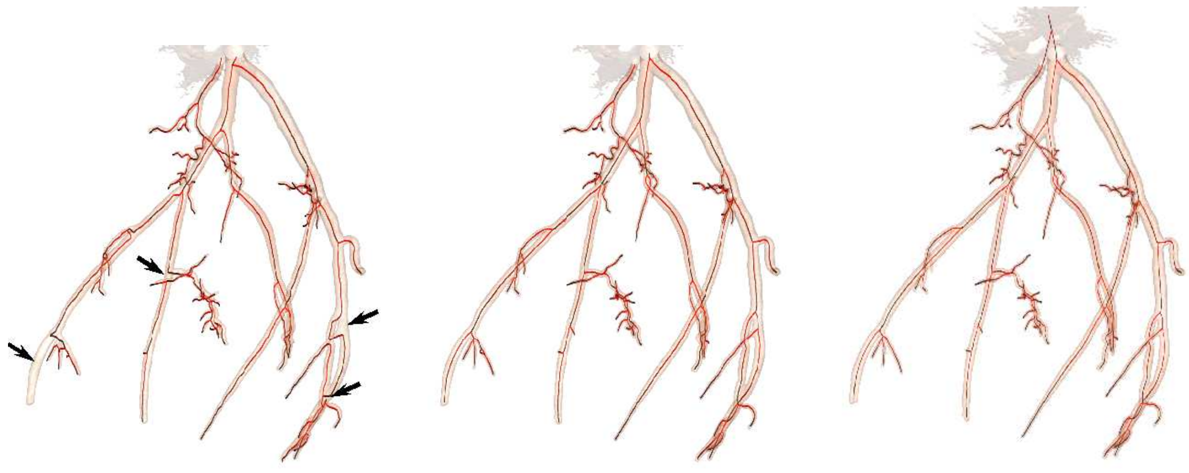
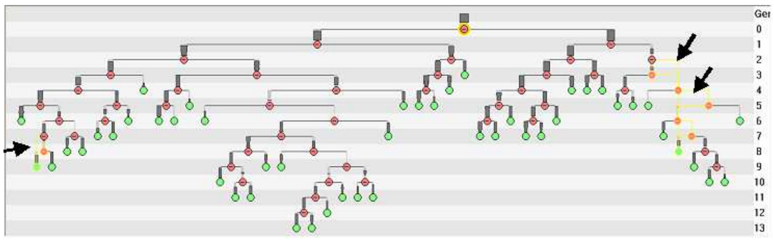
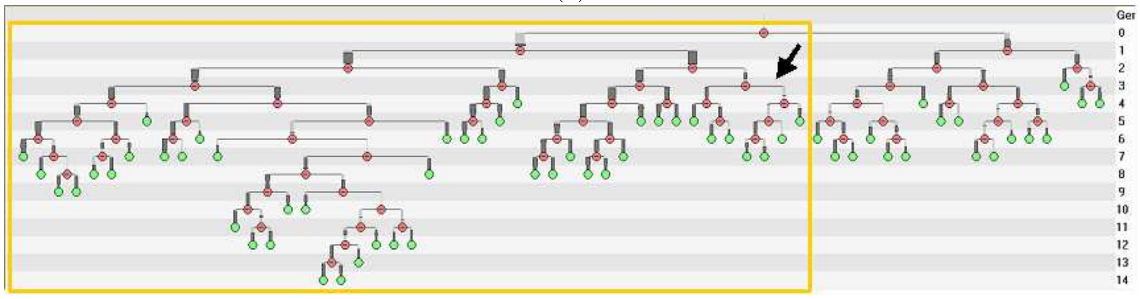

(a) (b) (c)

Figure 12: Tree analysis results for micro-CT scan h61. (a) Results after using Stage-2 centerline method of Kiraly *et al.*; lower left black arrow indicates a missing branch, while the other three black arrows point to incorrect branch regions resulting from improperly broken loops. (b) Results after Stage-2 centerline method of Yu. (c) Results after Stage-3 tree editing on part (b); one correct tree now exists. The extra mass rendered at the top of these figures represents the clay used to mount the arterial tree cast on a jar lid.



(a)



(b)

Figure 13: 2D Tree Maps of h61. (a) Tree Map of Stage-2 centerline results for the major tree depicted in the 3D rendering of Figure 12b before tree editing; arrows indicate loops (yellow labeled branches). (b) Final tree map, showing one correct tree after joining the two main trees and correcting other defects; this tree map corresponds to the rendering shown in Figure 12c; the yellow rectangle area is the major tree of part (a) after editing and the arrow points to the tree section that had two loops previously; the added portion to the right of the box arises from the smaller tree joined during tree editing.

(a)

g_i	$N^b(g_i)$	$\mu_L(g_i)$	$\mu_A(g_i)$
0	1	4.92	340.51
1	2	110.63	213.83
2	4	54.64	106.63
3	7	87.07	69.29
4	12	59.79	45.49
5	14	41.01	39.98
6	16	32.09	35.16
7	12	25.06	21.41
8	7	18.39	16.35
9	8	15.01	11.15
10	4	10.36	18.64
11	6	8.00	6.96
12	2	16.6	12.11
13	2	17.37	11.43
14	-	-	-

(b)

g_i	$N^b(g_i)$	$\mu_L(g_i)$	$\mu_A(g_i)$
0	1	16.58	—
1	2	59.34	113.35
2	4	72.90	174.65
3	8	45.86	74.76
4	14	57.50	59.14
5	20	42.03	40.96
6	22	28.31	38.27
7	20	29.76	31.84
8	10	32.78	20.47
9	4	24.02	16.84
10	4	14.37	12.32
11	4	10.36	18.64
12	6	8.00	6.96
13	2	16.60	12.11
14	2	17.37	11.34

Figure 14: Quantitative measurements for h61. (a) Results for the major tree before tree editing, corresponding to the major tree in Figure 12b and the tree map in Figure 13a. (b) Results for the complete tree after tree editing (two trees joined), per Figures 12c and 13b. "-" means a measurement can't be made; e.g., the final tree's root branch is not a real branch. Units for $\mu_L(g_i)$: voxels; for $\mu_A(g_i)$: (voxels)^2.

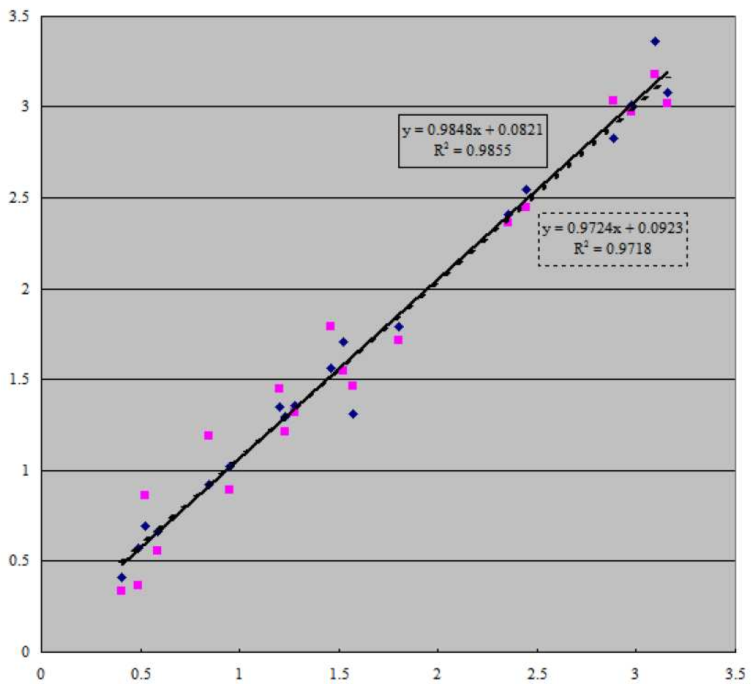
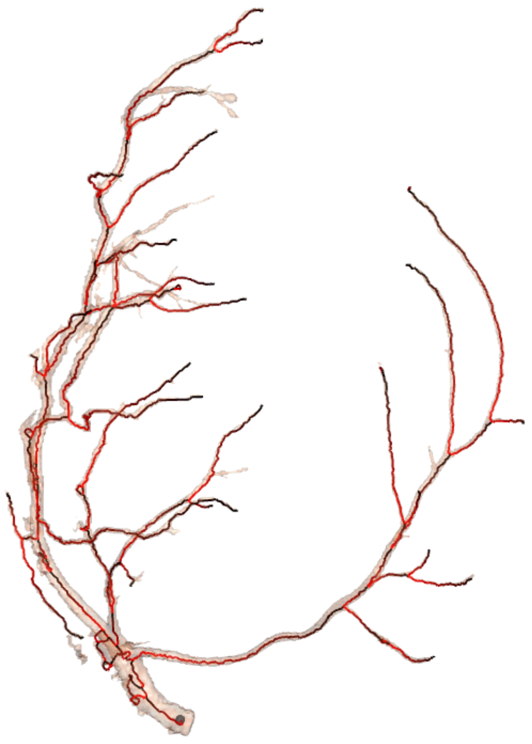


Figure 15: Comparison of h61 measured branch lengths for Kiraly versus Yu. Y axis represents Kiraly's and Yu's measurements in mm. The X axis represents the ground-truth branch lengths in mm. Diamonds are Yu's measurements and ground-truth correspondences. Squares indicate Kiraly's measurements and corresponding ground-truth measurements. The solid line is the linear regression line of Yu's measurements ($y = 0.9848x + 0.0821$), while the dotted line is the regression line of Kiraly's meausrements ($y = 0.9724x + 0.0923$). The matching branches between two techniques were found via a tree-matching algorithm [48].


(a)

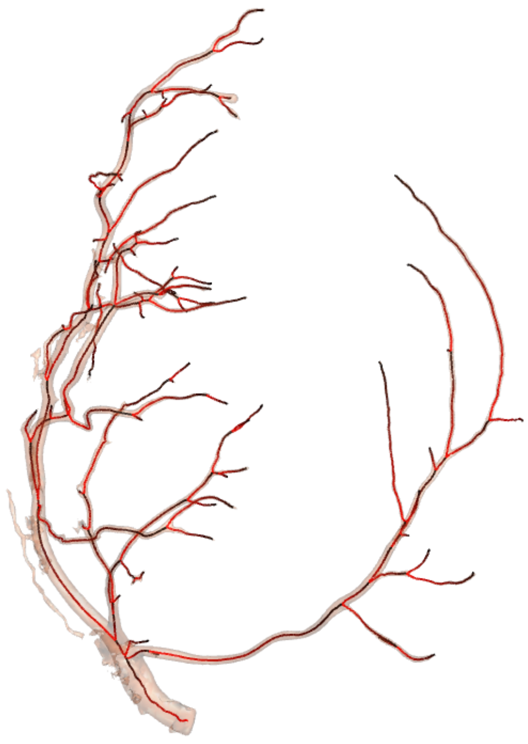

(b)

Figure 16: 3D renderings of trees produced for 3D micro-CT image lca_146 (401×267×490 image; voxel size = 20.8μm^3): (a) Via methods of Wan [10]; many missing branches are apparent from comparing the rendering and centerlines. (b) Tree Analyzer results using Yu's centerline method; a false attachment is edited out in this case.

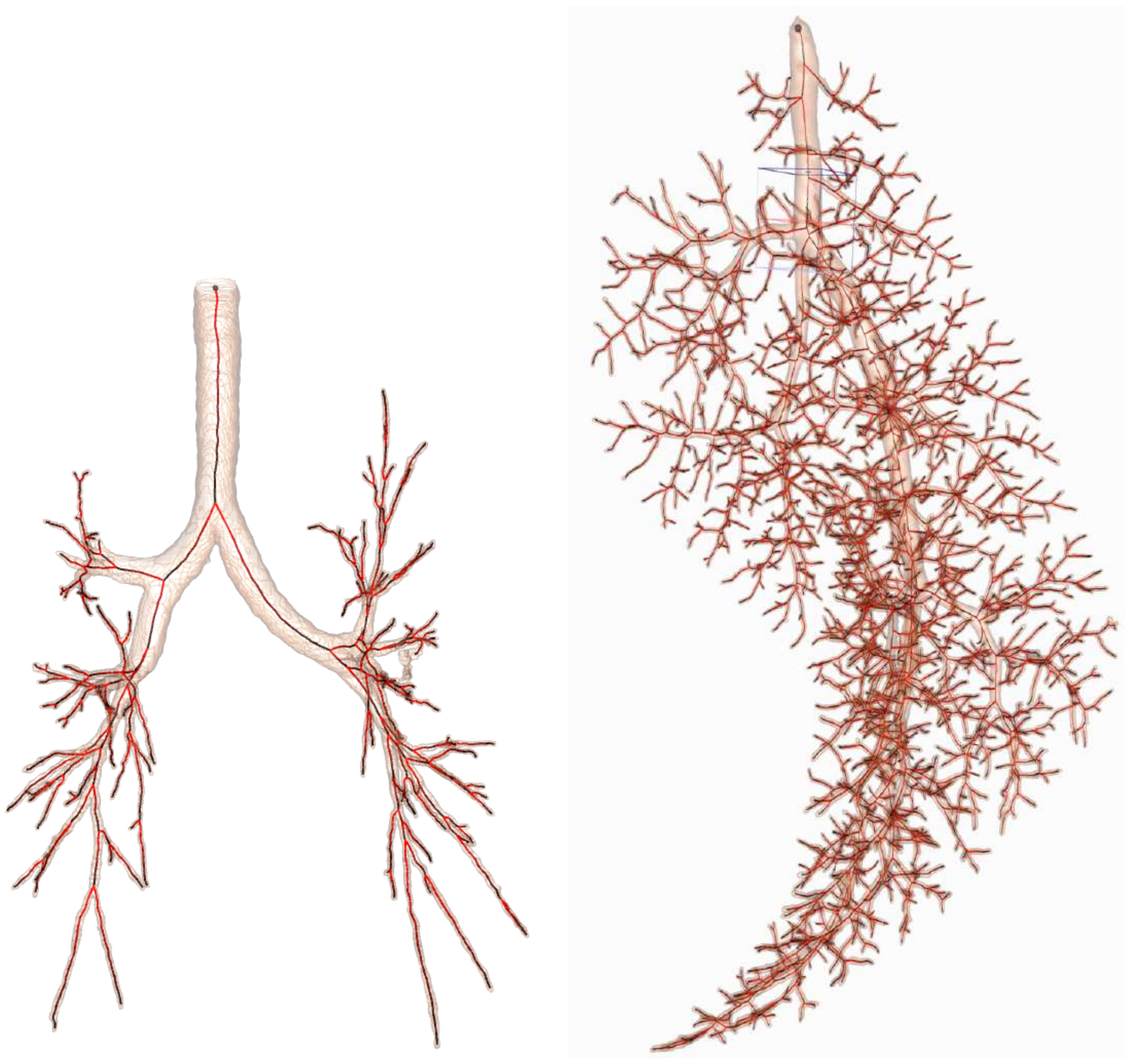
(a) Wan [10]

g_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$N^b(i)$	1	2	4	6	6	2	2	4	4	2	4	6	2	2	2	-	-	-

(b) Tree Analyzer

g_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$N^b(i)$	1	2	4	4	8	12	14	14	16	8	10	12	8	6	8	4	4	2

Figure 17: Quantitative results for lca_146, based on final trees depicted in Figure 16. The table depicts the number of generations and branches per generation extracted by the two approaches. A "-" means that no branches were extracted at generation g_i.



(a) (b)

Figure 18: Final Tree Analyzer results for two other cases: (a) human MDCT image h006_512_85 used in Figure 6; (b) r216_psf020826 considered in Figure 9.

SUMMARY

Modern micro-CT and multi-detector helical CT scanners can produce high-resolution 3D digital images of various anatomical trees, such as the coronary or hepatic vasculature and the airway tree. The sheer size and complexity of these trees make it essentially impossible to define them interactively, and the analysis and subsequent visualization of such images poses a considerable challenge. For example, a typical 3D micro-CT image can consist of several hundred megabytes of image data, with a voxel resolution on the order of twenty microns containing a tree having ten or more generations. Automatic approaches have been proposed for a few specific problems, but none of these approaches can guarantee extracting geometrically accurate multi-generational tree structures. This paper proposes a system for defining and visualizing large anatomical trees and then performing subsequent quantitative data mining of the extracted tree.

The system, dubbed the Tree Analyzer, processes an image in four major stages. In the first two stages, a series of automated 3D image-processing operations are applied to an input 3D digital image to produce a raw anatomical tree and several supplemental data structures describing the tree (central-axis structure, surface rendering polygonal data, quantitative description of all tree branches). Next, the human interacts with the system to visualize and correct potential defects in the extracted raw tree. A series of sophisticated 3D editing tools and automated operations are available for this step. Finally, the corrected tree can be visualized and manipulated for data mining, using a large number of graphics-based rendering tools, such as 3D global and local surface rendering, stereo viewing, sliding-thin slabs, multiplanar reformatted views, projection images, and a novel interactive tree map. The tree map, based a 2D directed graph representation, particularly offers many novel capabilities for interacting with and editing a complex 3D tree. Quantitative data can also be perused for the tree.

Results are presented for 3D micro-CT and human MDCT images. A quantitative comparison to human ground truth reveals the efficacy of the system. The total processing time for both the automated operations and subsequent interactive editing operations typically are done in less than 15 minutes for a typical 3D image.