System for 3D visualization and data mining of large vascular trees

Kun-Chang Yu, Erik L. Ritman, and William E. Higgins

Penn State University, University Park, PA 16802 USA

[ELR]Mayo Foundation, Rochester, MN 55905 USA

ABSTRACT

Modern micro-CT scanners produce very large 3D digital images of arterial trees. A typical 3D micro-CT image can consist of several hundred megabytes of image data, with a voxel resolution on the order of ten microns. The analysis and subsequent visualization of such images poses a considerable challenge. We describe a computer-based system for analyzing and visualizing such large 3D data sets. 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 arterial 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 stereo viewing, global and local surface rendering, sliding-thin slabs, multiplanar reformatted views, projection images, and an interactive tree map. Quantitative data can also be perused for the tree. Results are presented for 3D micro-CT images of the heart and liver.

Keywords: 3D visualization, tree analysis, data mining, vascular trees, micro-CT imaging, virtual endoscopy

1. INTRODUCTION

Modern micro-CT scanners produce very large three-dimensional (3D) digital images of vascular trees of the heart, liver, and other organs.[1–3] A typical 3D micro-CT image can consist of several hundred megabytes of image data, with a voxel resolution on the order of ten microns. The quantitative analysis and subsequent visualization of such images poses a considerable challenge. The specific problem we address is the extraction and quantitative definition of a vascular tree's structure. A typical micro-CT image captures many generations of connected branches for a typical tree. The extreme size and complexity of such images make it virtually impossible to define a tree completely and correctly, and manual definition is out of the question. Rudimentary automated techniques for definition of the vascular tree have been previously suggested.[4–10]

While these techniques can give a high percentage of correct branches, no technique guarantees the complete definition of a tree, 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, and imperfections arising in acquiring the image data during scanning.[9] 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.

Several observations need to be made on how to solve this basic problem. It is unrealistic to rely strictly on improved scanning technology and improved automated image-processing algorithms for defining an accurate tree. Yet, automated techniques, despite their imperfections, are essential in providing a complete description of a tree, and human interaction is essential for arriving at an accurate useful analysis of a tree. Our philosophy for tackling this problem, drawing upon these observations and espoused in earlier expositions of this effort,[11,12] is to use a judicious combination of automated analysis and computer-based human interaction to define a 3D tree.

For this purpose, we have devised a computer-based system, dubbed the Tree Analyzer. The basic qualitative design criteria that drove the construction of this system are as follows: (1) be acceptably computationally

Send correspondence to: weh2@psu.edu

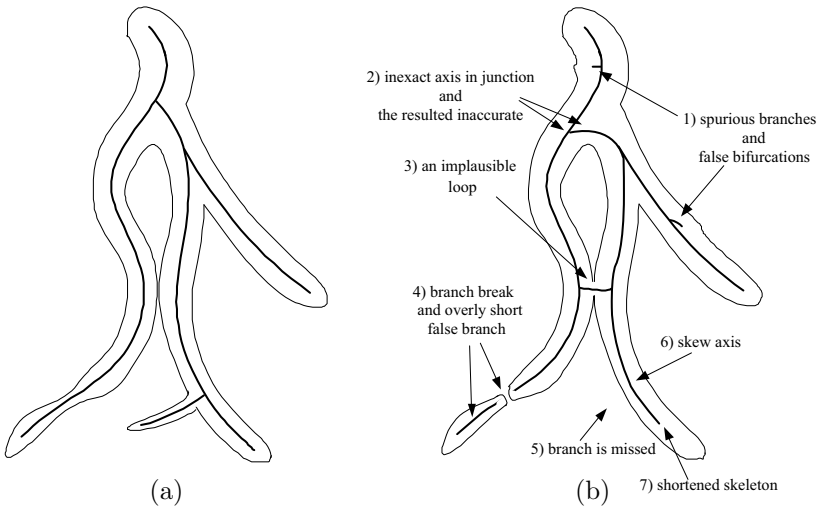


Figure 1. 2D Schematic figure illustrating the difficulties that can arise in defining the central axes of a vascular tree depicted in a 3D micro-CT image: (a) Ideal surface and central axes of tree; (b) typical output of purely automated 3D image analysis, depicting a tree with various imperfections.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

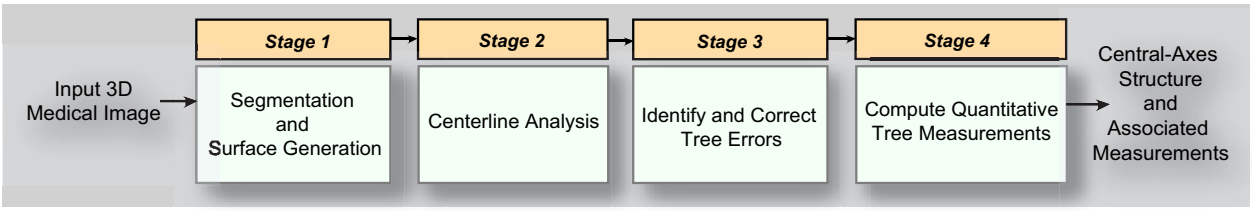


Figure 2. Four-stage top-level approach for defining the quantitative structure of a 3D vascular tree.

efficient; (2) require no more than a reasonable amount of human interaction; and (3) function over a range of anatomical and data variations. The system's software consists of a set of interactive visualization and data-mining tools, a built-in toolbox of 3D image processing operations, and a top-level graphical user interface (GUI) for analyzing a given 3D micro-CT image.[13,14]

The system is used following a 4-stage top-level approach depicted in Figure 2: (1) segment and define the surface of the tree of interest; (2) define the initial raw central axes, or centerlines, for the tree; (3) using various semi-automatic and interactive tools, correct identified tree defects; (4) perform interactive data mining to extract and examine quantitative tree data.

This paper expands upon earlier efforts in building the Tree Analyzer,[11,12] gives a complete discussion of the current system, and highlights some of the interactive data-mining capabilities. Section 2 gives a discussion of the system, Section 3 presents results, and Section 4 offers concluding comments.

2. METHODS

The Tree Analyzer is built on a Microsoft Windows XP platform and greatly expands upon an earlier system devised by Wan.[5] The software was developed using Visual Studio.Net 2003 and Visual C++. The code for managing windows, menus, and dialogue boxes, for performing basic input-output, for storing collections of data objects, and for accomplishing other functions, draws upon the standard Microsoft Foundation Class (MFC) library, the standard environment for building software systems in Microsoft Windows. Supplemental user interface components draw upon those in the Business Component Gallery (BCG), an extension library for MFC. Some of the Tree Analyzer's graphical and visualization features are derived from functions in the Visualization Toolkit (VTK).[15] The system can run on standard 32-bit Windows 2000 and XP PCs and on those running the 64-bit Windows XP x64 operating system.

The system features a large number of interactive visualization and data mining tools, in addition to an integrated 3D image-processing toolbox for automated analysis. All tools are tied together through a common

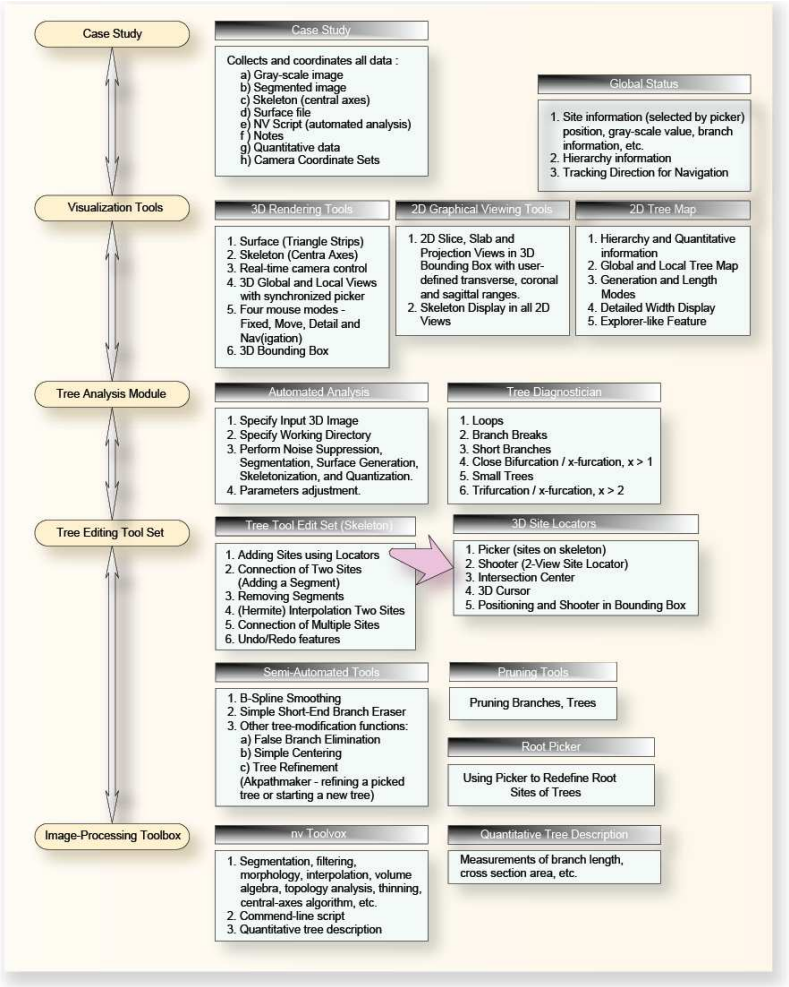


Figure 3. Complete diagram of software structure of the Tree Analyzer.

GUI. Fig. 3 gives a diagram of the entire software system. A synopsis of the various parts of the system is summarized below:

1. Case Study, for storing and managing all data objects during the analysis of an input image.
2. Visualization tools, for interacting and performing data mining operations.
3. Tree Analysis module, for performing functions specific to extracting and defining the quantitative structure of a tree.
4. Tree Editing Tool Set, for performing interactive and semi-automatic operations on a raw tree for the purpose of achieving a correct description.
5. Image Processing Toolbox NV, which serves as a general image-processing engine for computing a tree, computing quantitative tree data, and other general-purpose functions.

Further details are provided below, with the most complete discussions appearing in the references.[11,12,16]

2.1. Case Study

For a given input 3D micro-CT image, all data components for a completely processed tree are captured in a data structure referred to as the *case study*.[17] The case study contains the following data elements (Fig. 4):

1. Original 3D gray-scale image
2. 3D image of the segmented tree
3. Central axes (or skeleton) of the tree

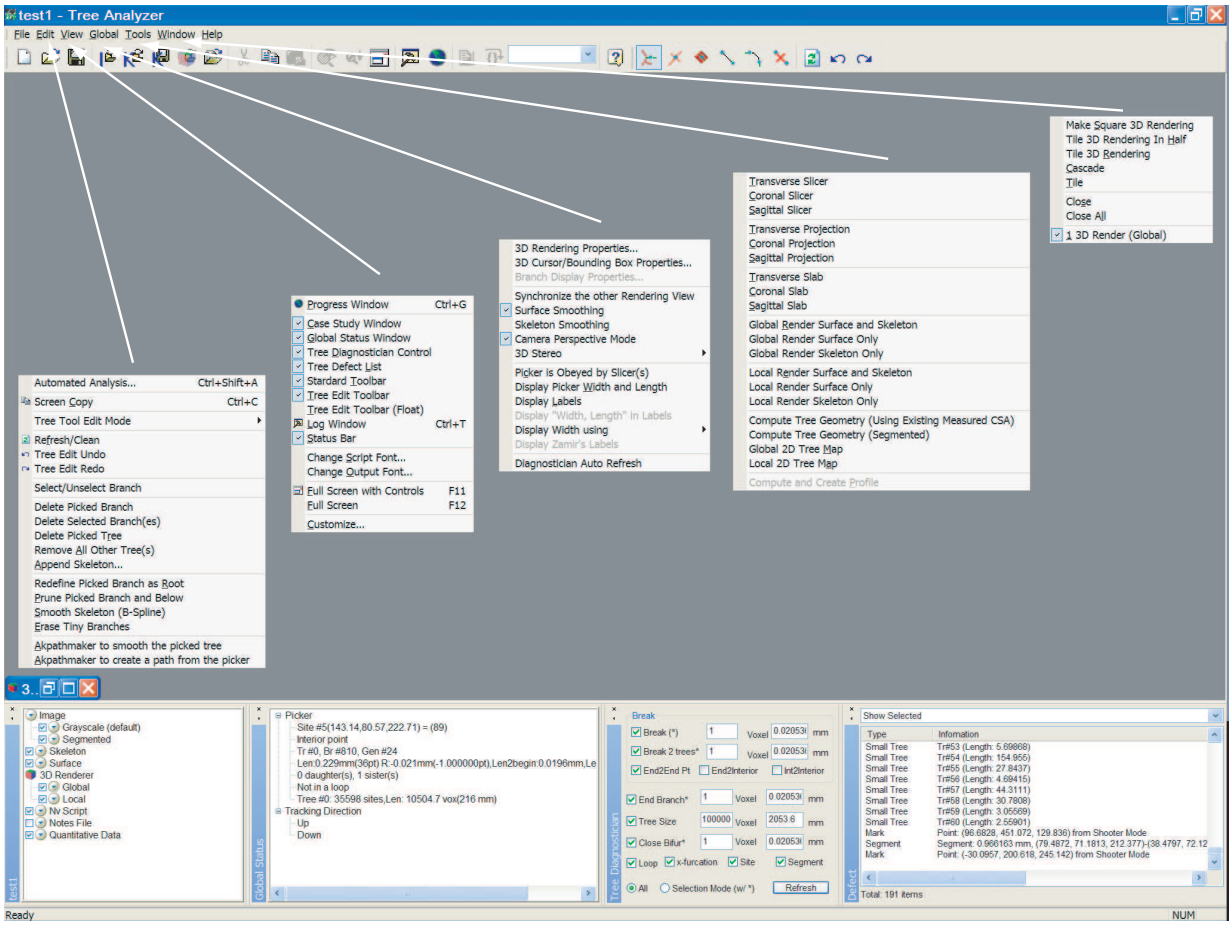


Figure 4. Top-level view of the Tree Analyzer's starting GUI configuration and a depiction of all drop-down menus. The bottom portion of this GUI configuration depicts the case study, global status (system view state), Tree Diagnostician, and the defect list identified by the Tree Diagnostician.

4. Surface data for rendering the tree
5. An automated analysis script (NV script) containing a sequence of image-processing operations for performing all Stage-1 and Stage-2 analyses.
6. User-input text notes for documenting observations on the case
7. Quantitative data describing the axial structure of the tree
8. Parameters saved for visualizing a tree during an interactive data mining session (camera coordinate sets)

A case study is built from an input 3D micro-CT data set following the four-stage processing flow of Fig. 2. NV, in conjunction with a procedure tuned to tree analysis (Tree Analysis module), is used to automatically run Stages 1 and 2. After tree defects are identified with the Tree Diagnostician (part of the Tree Analysis module), the user interacts with the Tree Editing and Visualization tools to arrive at a corrected tree.[12] Finally, the Visualization tools are used for interactive quantitative data mining.

The construction of the case study occurs through interaction with the Tree Analyzer's GUI. A synopsis of GUI functions is shown in Fig. 4. The case-study dialogue box is shown in the lower left portion of Fig. 4. The subsequent sections give further discussion of the GUI and system operation.

2.2. Visualization Tools

A large set of visualization tools are available for viewing the raw gray-scale data, in addition to viewing various types of renderings of the raw data and segmented tree (Figs. 5-6):

1. Global and local surface renderings of the segmented tree (3D Render view) (Fig. 5). The global rendering shows the entire tree, while a local rendering zooms in on a particular site of interest. These views also

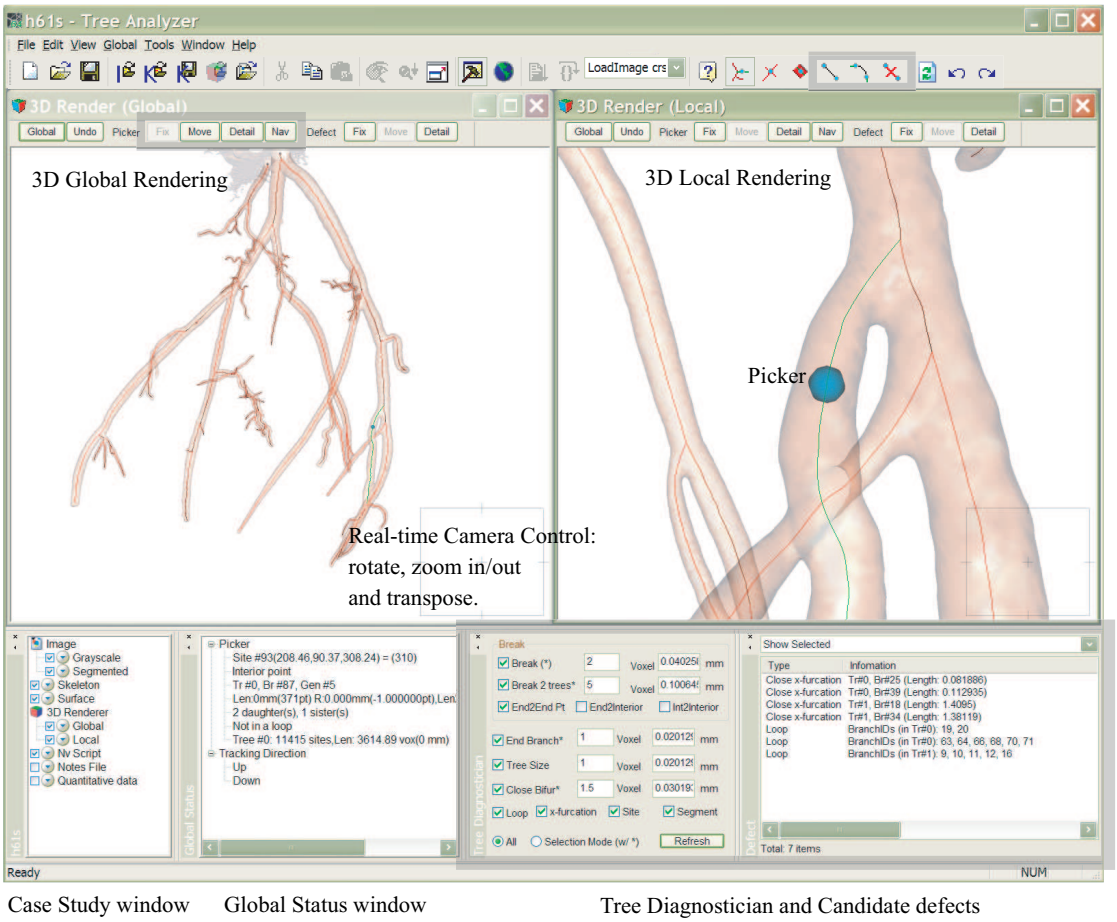


Figure 5. Composite View of Tree Analyzer.

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.

2. Transverse, sagittal, and coronal slice views (Fig. 6). These are standard 2D sections of the gray-scale data in the x-y, y-z, and x-z planes.
3. Transverse, sagittal, and coronal projection views (Fig. 6). These views give 2D projections of selected subsets of the original gray-scale data along either the z, x, or y directions.
4. Transverse, sagittal, and coronal thin-slab renderings, which enable more sophisticated views of subsets of the gray-scale data. Thin slabs incorporating data depth and various windowings can be computed to reveal more structural data inside of a 3D volume.[18]
5. A bounding box feature that enables the user to compute either a slice, projection, or slab view focusing on a specified 3D bounding box about a selected 3D site of interest (Fig. 6).
6. A tree map that depicts the defined axial structure of the tree in a graphical form (Figs. 11 and 15).
7. Quantitative summaries of an extracted tree (Fig. 16).

All tools "obey the system," per the user's movement of the computer mouse. The current interactive status of the system is captured in the *view state*, represented in the Global Status window (bottom of Fig. 4, second from the left) and the picker (Fig. 5).[12] In general, to interact with an extracted tree, the user points to a global rendering of the tree, and all other active views follow the movement of the mouse. During interaction, four viewing modes are possible:

1. Fixed — The current 3D site of the mouse appears on the view, with no other change in the view.

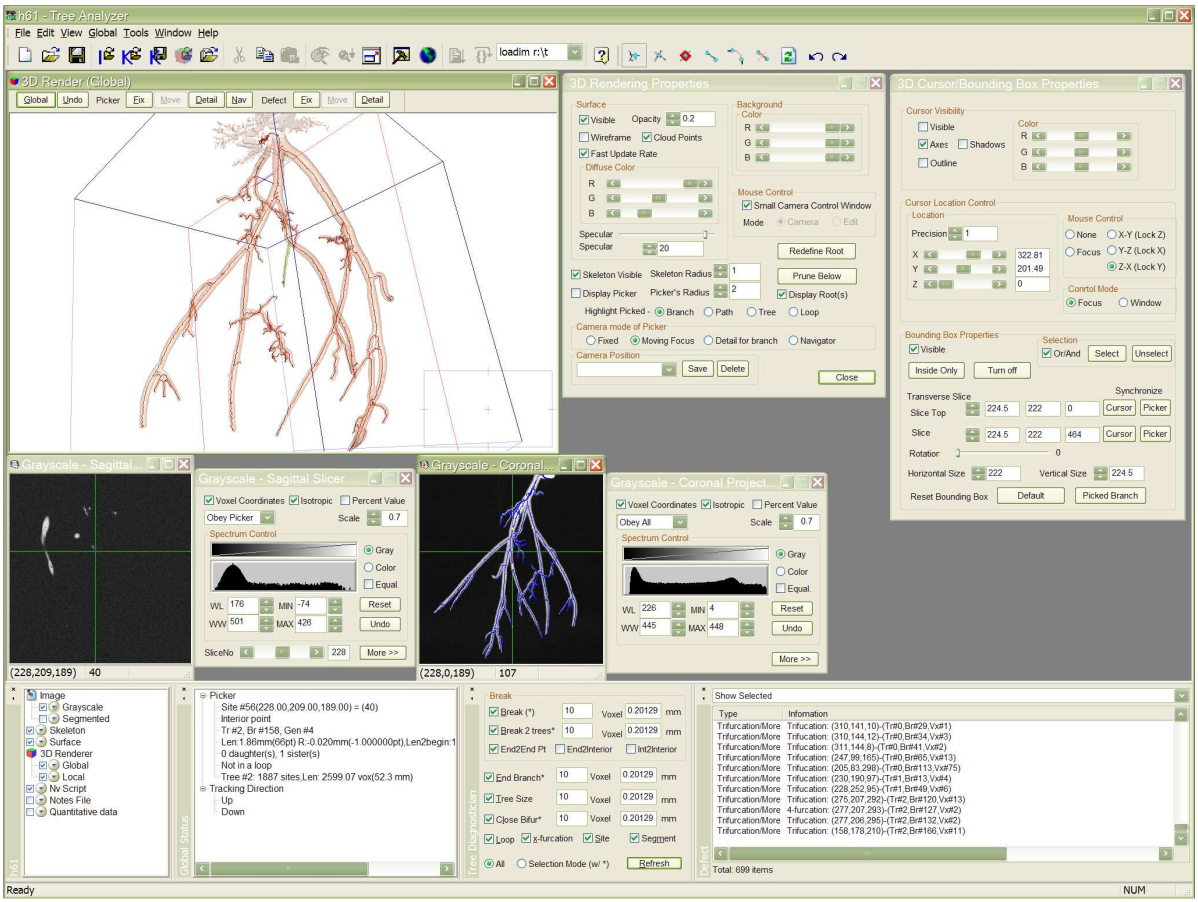


Figure 6. A second example lay-out for the Tree Analyzer. A bounding box is shown in the 3D Render window. You can use its control dialog (the top middle window) to adjust the position of the bounding box. A red square in the bounding box of the left top window indicates that the 2D view of the transverse slice 91 is displayed in the far-left center window. The third window from the left in the center is a sagittal projection of the bounding box.

2. Move — The view moves to the 3D site pointed to by the mouse. The magnification of the view doesn't change, just the center.
3. Detail — The views zooms in about (magnifies) the location pointed to by the mouse.
4. Navigation — The view, if possible, shows an interior view of the tree branch at the selected 3D site.

Thus, for a selected 3D site, the user gets a composite view of image data about the site through the use of the multiple viewers. Fig. 5 shows examples the 3D Render tool, while Fig. 6 depicts slice and projection views restricted by the bounding box feature.

2.3. NV - General Image-Processing Toolbox

The NV toolbox, integrated as part of the kernel of the Tree Analyzer's software structure, contains a large number of general purpose 3D image-processing functions. Currently, 104 total functions are available in the following general categories:

1. image-enhancement filters, for noise reduction and general filtering
2. mathematical morphology operations, for shape-based image analysis
3. topological operations, for defining connected components, deleting interior cavities, and other related operations
4. image segmentation operations, for region extraction
5. skeleton operations, for processing axial and digital-skeleton representations

6. image manipulation operations, such as adding two images, performing simple logical combinations of images, and others
7. workspace and system operations, for manipulating intermediate processed results and input-output
8. 3D visualization functions, for computing region surface representations of data components

NV enables the construction of general scripts, which consist of sequences of selected operations to perform a specified image-processing task automatically. The Tree Analysis dialogue, discussed below, is based on an underlying NV script, specific to extracting 3D vascular trees. The references give a general manual for NV's functions and capabilities.[16]

2.4. Tree Analysis Module

Note that the Tree Analyzer has considerable general capability that makes it potentially useful for other 3D medical imaging problems. The Tree Analysis module focuses on the specific operations for extracting a raw tree and for diagnosing potential tree defects.

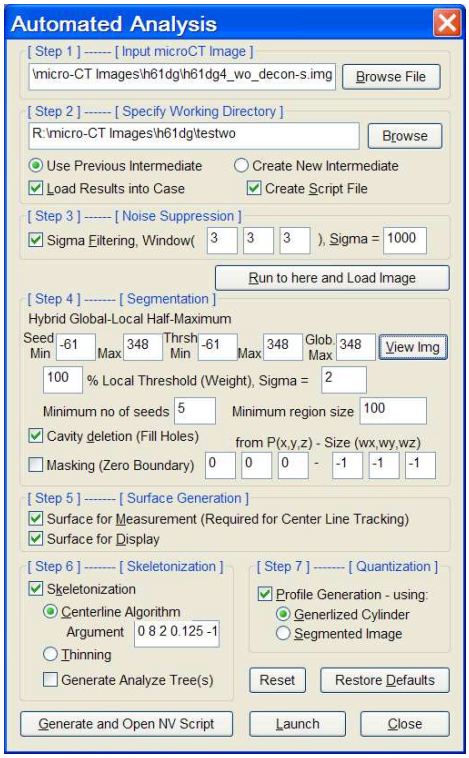


Figure 7. Tree Analyzer dialogue used for setting up an automated analysis run of Stages 1 and 2 on an input micro-CT image.

The sequence of specific operations to segment the raw tree, define the tree's surface data (used for later rendering) and extract the raw central axes of the tree (Stages 1 and 2, per Fig. 2) are depicted in the system dialogue box of Fig. 7. This operation sequence is performed by setting parameters in this dialogue (which in turn internally modify the built-in NV script for tree analysis) and by then invoking the specified analysis. The specific image-processing functions captured in this dialogue are all contained within the general toolbox NV. A block diagram of this sequence of functions is shown in Fig. 8. The basic operations are: (1) segment the raw tree to generate the image I_s; (2) define the surface data for later rendering using I_s and the original raw gray-scale image I_o; (3) extract the central axes of the tree using the tree surface; and (4) compute the quantitative tree measurements. The references give a detailed discussion of this sequence.[11,12,16]

During Stage-3, possible tree defects are identified automatically with the Tree Diagnostician. Examples of the Tree Diagnostician and associated Defect List dialogue boxes are shown in the lower right portion of Fig. 4. Candidate defects, such as possibly broken branches, breaks between separate (but possibly false) trees, overly short end branches, and others are easily identified from the raw extracted central axes.

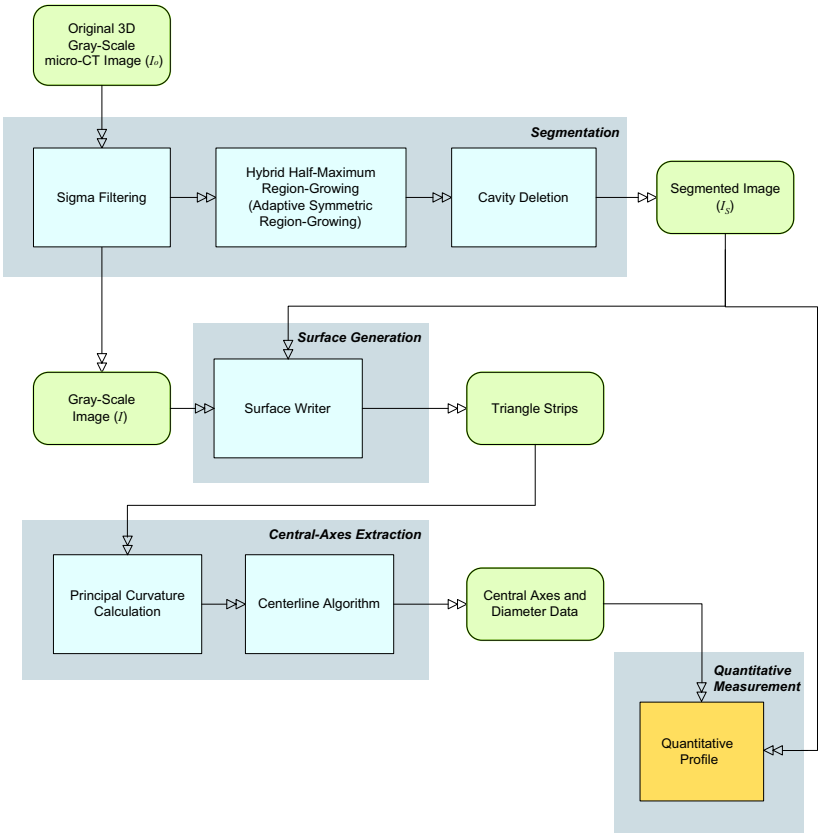


Figure 8. Diagram of all automated operations used for segmenting the raw tree, computing the initial set of central axes, computing the data used for interactive rendering, and for making the quantitative measurements of the tree.

2.5. Tree Editing Toolset

In general, the user cycles through each candidate defect identified by the Tree Diagnostician. When the user selects a defect on the list, the system's global status (view state) is moved to this position and all visual tools are updated to reflect this position. The user can then apply the various tree editing tools to fix the defect.

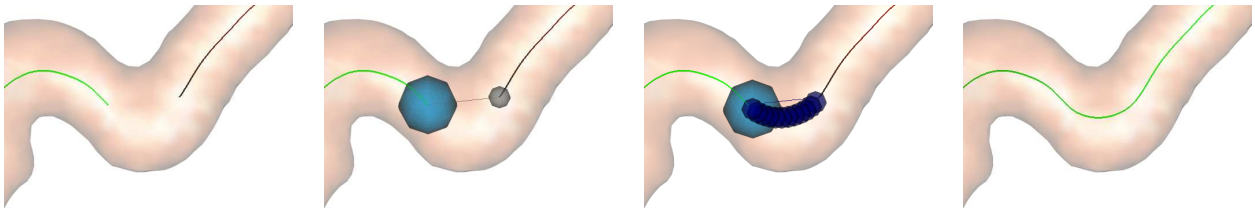
The Tree Editing Tool Set includes axis editing tools, semi-automated tools, and simple automated tools. The axis editing tools allow the user to add new 3D sites using locator tools[11]; connect two sites by adding a segment or a series of sites and their associated segments using Hermite interpolation (Fig. 9 gives an example); and remove a segment to break two connected sites. Our previous work[11] gave an example demonstrating the repair of a broken branch (the tree editing tools were used in conjunction with 3D Bounding Box).

Semi-automatic tree-editing tools are selectable as a pop-up menu from the 3D Render tool (Fig. 10). It provides many axis editing functions (the fourth group in the menu) such as "Delete Picked Branch" to delete a manually selected branch, etc. The sixth group in the menu lists semi-automated tools and simple automated tools. For example, the user can redefine the root of a tree and prune all branches below a selected site. The semi-automated tools also provide B-spline smoothing that helps smooth the existing axes by adjusting the positions of sites, and they provide pruning functions such as the "short-end-branch eraser," which helps remove spurious branches as shown in Fig. 1b.

All of these tools are used to correct the candidate defects identified by the Tree Diagnostician. As edits are made to the tree structure, the user can decide to undo changes or keep changes. When changes are kept, the stored tree structure is updated to reflect the corrections.

2.6. Data Mining and the Complete System

After the tree is fixed, the quantitative tree description is computed. Fig. 11 gives a snapshot of the data computed for an interior tree position. This figure shows the rendered tree, axes, and corresponding position of



(a) (b) (c) (d)

Figure 9. Example of using the axis editing tool: (a) a branch showing a gap between two separated axes; (b) the two sites are connected by using the mouse; (c) a series of sites are generated by Hermite interpolation; (d) resulting connected axis.

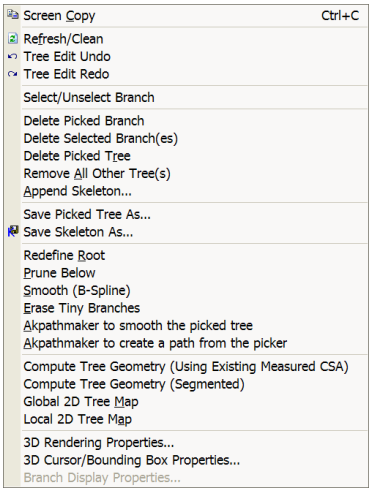


Figure 10. Semi-automatic functions for tree editing, selectable from the 3D Render tool.

the tree map.

Figs. 5 and 6 illustrate some the extensive visual and data mining capabilities of the system. Fig. 6 depicts a use of the bounding box feature to focus on a site of interest, where slice and projection views focused within the bounding-box region depict supplemental views. The next section gives pictorial and quantitative results for the system.

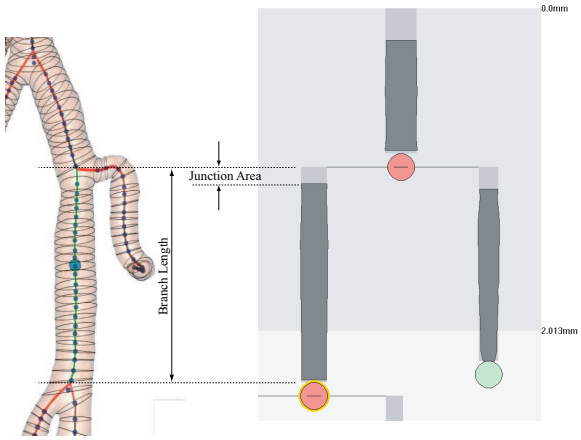
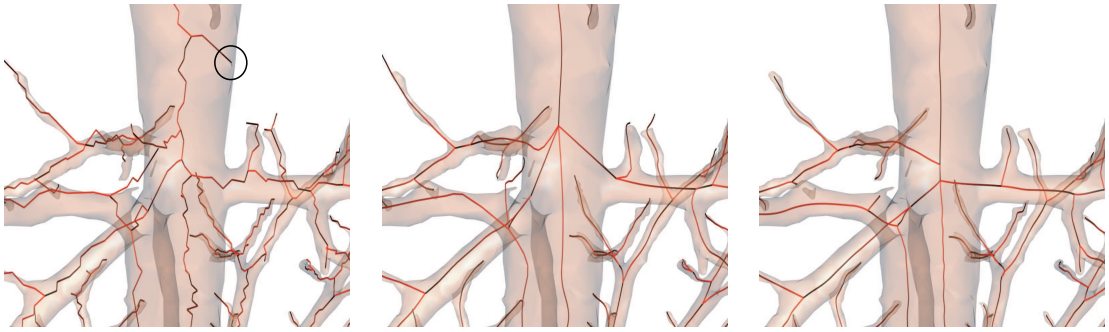
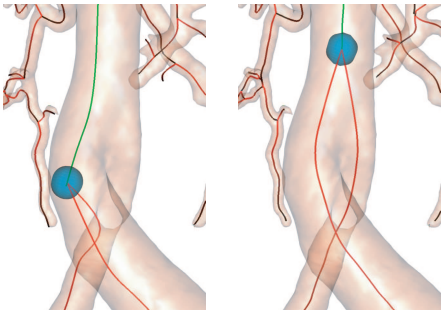


Figure 11. A partial depiction of how quantitative tree information is depicted visually. Left side: a portion of the 3D rendered tree showing the axes (centerlines). Right side: corresponding excerpt of Tree Maps, where the dark gray bars indicate diameter along the branch, the circles indicate branch points, and the numbers in mm indicate branch length.



(a) (b) (c)

Figure 12. Results for a complex junction, with five adjacent branches: (a) Wan,[5] (b) Kiraly *et al.*,[19] (c) Tree Analyzer.



(a) (b)

Figure 13. Results for a situation where two twisting branches touch each other: (a) Kiraly *et al.* - the axes are incorrect[19]; (b) Tree Analyzer result giving the correct axes.

3. RESULTS

Figs. 12 and 13 give examples of results for our system at a complex junction and for a pair of overlapping branches. It is clear that the axial structures produced by the Tree Analyzer are far more accurate than the previous approaches.

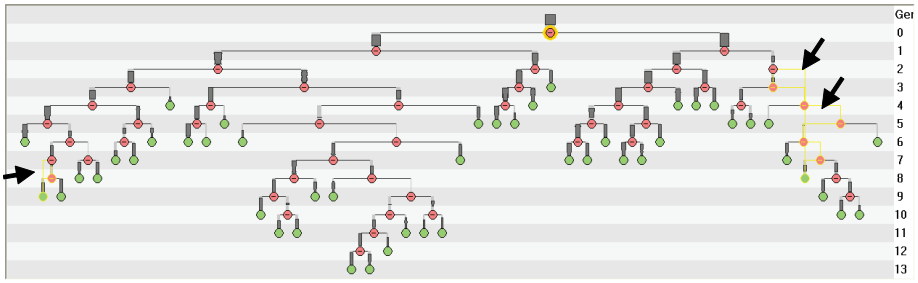
Fig. 14 shows results of central-axes analysis on *h61*, a typical micro-CT coronary artery image reconstructed without applying deconvolution[20–22] (image characteristics: dimensions 450 × 445 × 465 voxels = 9.06mm × 8.96 mm × 9.36mm real volume extent; $\Delta x = \Delta y = \Delta z = 20.13\mu$m). The image depicts a cast of a mouse coronary arterial tree, where the cast was attached to the lid of a jar using clay. Fig. 14a displays the result of using a previously proposed axial-extraction method by Kiraly *et al.*[19] Fig. 14b presents the result of using the Tree Analyzer before tree editing. Some branches missing in the Kiraly *et al.*'s result appear in the result of Tree Analyzer. Also, Kiraly *et al.*'s method cannot resolve overlapping branches, which usually create a loop (Fig. 13).

Using the Tree Diagnostician, no loops were found using Kiraly *et al.*'s method, since loops are automatically discarded in the method. We found three 3D handle loops, however, in the central axes generated by Tree Analyzer before fixing. Any other potential loops, due to overlapping branches in the segmented tree, are resolved properly by the Stage-2 algorithm. The handle loops can be manually fixed by using the Tree Editing tool set. For h61, two valid trees were generated and appear separated, because of the clay situated at the tree root. We manually merged the two trees into one tree. The resulting tree is shown in Fig. 14c. It took a human operator five minutes to remove the defects. Fig. 15a shows a 2D Tree Map before manually connecting the two separated trees. After manual editing (Figure 14c), fifteen generations were found in the final axes. Figure 15b displays the 2D tree map of the final axes in generation mode.

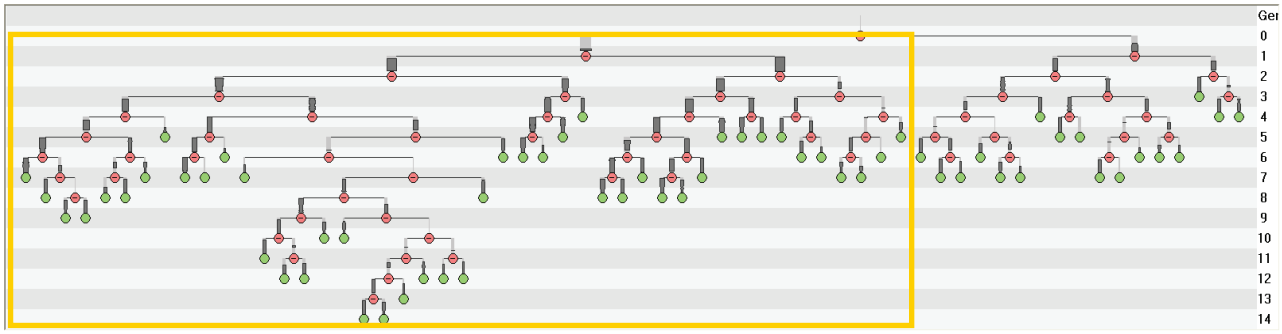
Quantitative measurements were also computed for h61, before and after tree editing, as shown in Fig. 16. In this figure, "GenID" refers to generation index, "NumBR" refers to the number of branches existing at the indicated generation, "AvgBrLen" refers to average branch length in microns, "AvgCSA" refers to the average cross-sectional area in $(\text{microns})^2$ for branches at the indicated generation, "DevCSA" refers to the standard

(a) (b) (c)

Figure 14. Tree analysis results for micro-CT scan h61: (a) Kiraly *et al.*,[19] (b) Tree Analyzer showing two separate valid trees(before tree editing), (c) Tree Analyzer (after tree editing). Extra mass at the top of these figures represents the clay used to mount the arterial tree on the jar lid.



(a)



(b)

Figure 15. 2D Tree Maps of h61 (Tree Analyzer result). (a) The major tree (of two separated trees) before tree editing as shown in Fig. 14b. Arrows indicate loops (yellow labeled branches) in the tree map. (b) Final tree, joining the two main trees, having fifteen generations after tree editing (Fig. 14c). The yellow rectangle area is the major tree before the editing.

GenID	NumBr	AvgBrLen	NumGC	AvgCSA	DevCSA	AvgBrCSA	DevBrCSA	AvgSurf	DevSurf	AvgVol	DevVol	Avg2Root	Dev2Root
0	1	4.92	2	340.51	0.81	340.51	0	235.2	0	1224.26	0	0	0
1	2	110.63	70	213.83	3.24	215.5	5.3	5108.86	2235.01	21017.92	8983.2	4.92	0
2	4	54.64	65	106.63	3.59	94.82	51.86	1515.96	1038.57	4586.27	3424.78	115.55	43.13
3	7	87.07	223	69.29	2.81	68.37	42.87	2227.32	1589.32	5532.94	4439.87	167.69	22.67
4	12	59.79	248	45.49	2.29	40.17	29.27	1244.27	1047.85	2491.88	2349.56	260.1	40.87
5	14	41.01	208	39.98	2	28.68	23.2	782.08	1122.35	1496.8	2647.13	289.63	68.58
6	16	32.09	182	35.16	1.62	17.89	18.76	586.7	931.91	1065.83	1947.45	320.43	66.99
7	12	25.06	117	21.41	1.56	15.12	10.78	335.25	321.41	457.47	523.34	356.28	86
8	7	18.39	54	16.35	1.3	15.75	6.35	199.24	143.28	234.22	221.08	326.88	75.54
9	8	15.01	44	11.15	1.55	9.66	6.6	134.18	111.75	138.04	143.52	334.9	65.86
10	4	10.36	13	18.64	1.01	14.83	3.84	122.43	142.07	149.12	189.2	354.18	15.84
11	6	8	25	6.96	0.9	8.81	5.06	48.87	21.94	36.77	15.62	354.1	15.67
12	2	16.6	13	12.11	1.15	10.7	6.13	159.29	92.82	166.29	125.34	346.59	0
13	2	17.37	14	11.34	1.31	10.96	2.62	175.91	44.85	168.56	61.52	367.08	0

(a)

GenID	NumBr	AvgBrLen	NumGC	AvgCSA	DevCSA	AvgBrCSA	DevBrCSA	AvgSurf	DevSurf	AvgVol	DevVol	Avg2Root	Dev2Root
0	1	16.58	0	-	-	-	-	-	-	-	-	-	-
1	2	59.34	16	113.35	2.19	210.70	129.81	821.91	586.72	2,399.50	1,175.24	16.58	-
2	4	72.90	90	174.65	2.55	126.24	89.34	2,831.02	2,777.23	10,986.61	11,877.05	75.92	26.77
3	8	45.86	109	74.76	2.80	59.82	50.92	1,052.21	920.59	2,736.47	3,082.78	148.82	33.22
4	14	57.50	277	59.14	2.19	42.35	40.13	1,287.90	1,471.43	2,974.79	4,055.12	195.73	33.40
5	20	42.03	290	40.96	1.96	30.38	25.87	812.61	975.56	1,566.46	2,148.02	273.32	51.65
6	22	28.31	222	38.27	1.71	21.83	21.08	508.82	963.79	966.45	2,224.56	292.46	78.73
7	20	29.76	214	31.84	1.59	17.04	16.68	511.91	851.99	897.92	1,777.77	334.51	84.26
8	10	32.78	131	20.47	1.70	17.56	9.78	441.92	307.63	588.47	514.78	428.69	64.14
9	4	24.02	41	16.84	1.43	16.55	8.27	281.67	134.47	338.33	241.09	404.20	36.50
10	4	14.37	22	12.32	1.20	10.03	9.16	144.29	112.79	162.46	168.62	386.95	6.92
11	4	10.36	13	18.64	1.01	14.83	3.84	122.43	142.07	149.12	189.20	398.42	15.84
12	6	8.00	25	6.96	0.90	8.81	5.06	48.87	21.94	36.77	15.62	398.34	15.67
13	2	16.60	13	12.11	1.15	10.70	6.13	159.29	92.82	166.29	125.34	390.83	-
14	2	17.37	14	11.34	1.31	10.96	2.62	175.91	44.85	168.56	61.52	411.32	-

(b)

Figure 16. Quantitative measurements for h61 (Tree Analyzer result). (a) Before tree editing: Two trees were generated in the segmented image as shown in Fig. 14b; the measurements are computed for the major tree of the two separate trees. (b) The measurements are computed for the merged tree after tree editing, per Fig. 14c. "-" indicates the measurements cannot be performed; e.g., a tiny branch is within a junction.

deviation in cross-sectional area in microns for branches at a given generation, etc. A complete discussion of these quantities is given in the references.[5,16] The tables show that the generational description for h61's tree is far more sensible (and correct!) after tree editing than before, at least up to generation 5.

To verify the correctness of the numbers above, Fig. 17 gives a comparison of the measured branch lengths for h61 between manual measurements made by a skilled human operator(done using a microscope),[11] Kiraly *et al.*'s method, and the Tree Analyzer (after tree editing). Fig. 17 shows that the comparison result for Tree Analyzer has a better linear regressive slope (0.985 versus 0.972 for Kiraly *et al.*), and it also has a better R-squared value ($R^2 = 0.986$ versus 0.972 for Kiraly *et al.*). After we manually fixed most of the defects in Kiraly *et al.*'s result, the output is still more scattered due to the improper position of branch points, as shown in Fig. 17. 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. A regressive line is most reliable when its R-squared value is at or near 1.

Fig. 18 gives results for a second micro-CT image of the vasculature of a rat liver (image r216_psf020826, reconstructed with a deconvolution algorithm; dimensions $620 \times 500 \times 1000$ or 12.57mm $\times$ 10.13mm $\times$ 20.27mm in real volume; $\Delta x = \Delta y = \Delta z = 20.27\mu$m). Fig. 18a shows the axes produced by using a semi-automated branch-extraction technique found in the Analyze^{TM} package.[23,24] Fig. 18b shows the central axes produced by the Tree Analyzer. The Analyze^{TM} technique produces axes approximated to integer coordinates only, while the Tree Analyzer gives smooth real-valued (floating point) coordinates for the axes. Figs. 18c and 18d focus on a particular complex junction of this tree, situated near the middle right part of the tree. In Fig. 18c, the axes generated using Analyze^{TM} are not smooth in the main trunk due to the integer format. Many spurious

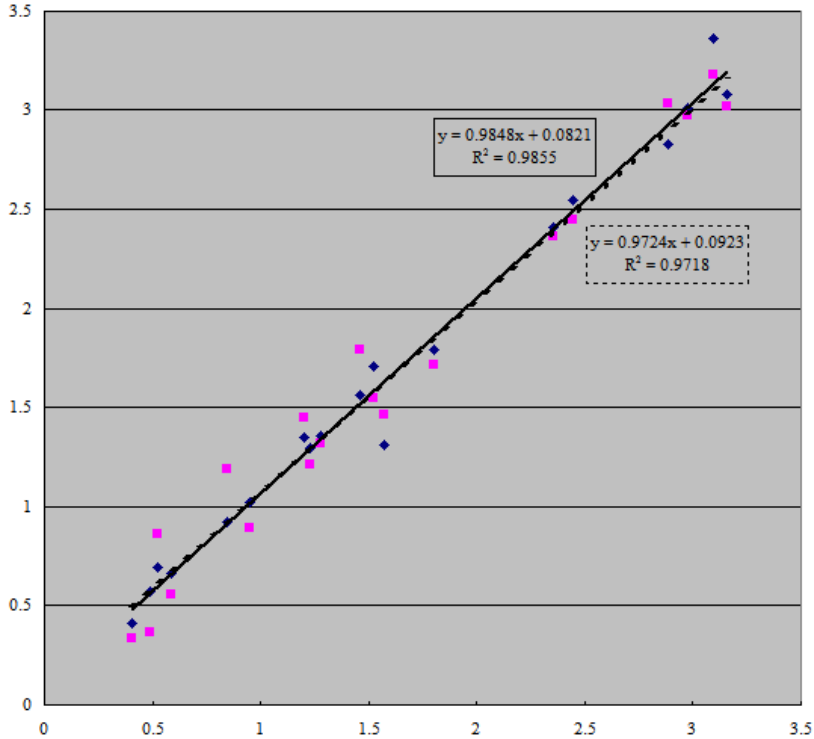


Figure 17. Comparison of results: measured branch lengths for h61 of using Kiraly *et al.*'s method versus the Tree Analyzer. Y axis represents both Tree Analyzer and Kiraly *et al.*'s measurements in mm. X axis represents the measured branch lengths (in mm) made by a skilled human operator using a microscope by Zamir *et al.*, which represents the ground truth.[11] Diamonds indicate the Tree Analyzer's measurements and their corresponding Zamir *et al.*'s measurements. (The corresponding measurements of "matched" branches are found by using Michael Graham's matching algorithm.) Squares indicate Kiraly *et al.*'s measurements and the corresponding Zamir *et al.*'s measurements. Solid line is the linear regression line of Tree Analyzer's measurements versus Zamir *et al.*'s measurements. Dotted line is the regression line of Kiraly *et al.*'s results.

tiny axes are produced as well. Fig. 18d shows the Tree Analyzer's axes, which are smooth and exhibit no spurious branch axes or trifurcations. For Analyze^{TM}, 18 trifurcations were found in the generated axes. Note that a trifurcation can be defined if two bifurcations are very close and their parent branch belongs to their main trunk.

4. 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 vascular 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 multi-detector CT images of the lungs 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.

This paper attempted to give a brief overview of this large system. It supplements and updates presentations given earlier.[11,12] The most up-to-date description of the system is given in the thesis by Yu.[16]

ACKNOWLEDGMENTS

This work was partially supported by grants #EB000305, #CA74325 and #CA091534 from the NIH. We also thank Michael Graham for the matching results of h61 (Fig. 17).

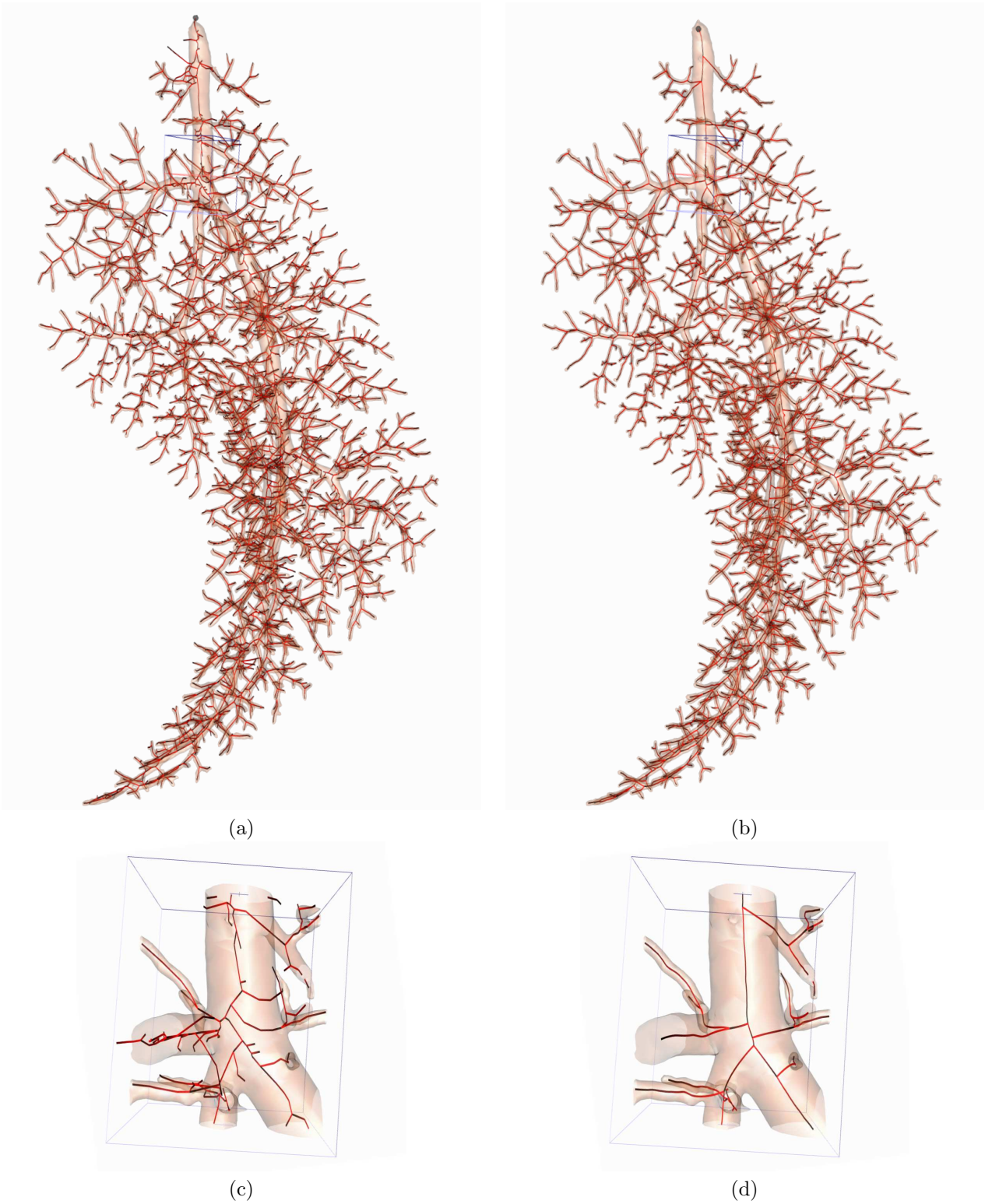


Figure 18. Tree analysis results for r216_psf020826 ([rat liver micro-CT image] dimensions $620 \times 500 \times 1000$ [real volume dimensions: 12.57mm $\times$ 10.13mm $\times$ 20.27mm], $\Delta x = \Delta y = \Delta z = 20.27\mu$m): (a) Analyze^{TM},[23,24] (b) Tree Analyzer. (c) The axes generated using Analyze^{TM} are not smooth due to the integer format. Many spurious tiny axes are produced as well. (d) The Tree Analyzer's axes, which are smooth and exhibit no spurious branch axes or trifurcations.

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. Agustin Garcia-Sanz, Alicia Rodriguez-Barbero, Michael D. Bentley, Erik L. Ritman, and J. Carlos Romero, "Three-dimensional microcomputed tomography of renal vasculature in rats," *Hypertension*, vol. 31, no. Part 2, pp. 440–444, 1998.
3. A. Feldkamp, S. A. Goldstein, A. M. Parfitt, G. Jesion, and M. Kleerekoper, "The direct examination of three-dimensional bone architecture in vitro by computed tomography," *Bone Min. Res.*, vol. 4, pp. 3–11, 1989.
4. D. Selle, B. 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.
5. 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.
6. 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.
7. P. J. Yim, "Gray-scale skeletonization of small vessels in magnetic resonance angiography," *IEEE Transactions on Medical Imaging*, vol. 19, no. 6, pp. 568–576, June 2000.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. J Han and M Kamber, *Data Mining: Concepts and Techniques*, Morgan Kauffman, San Francisco, 2001.
14. D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1 –8, Jan.-Mar. 2002.
15. Will Schroeder, Ken Martin, and Bill Lorensen, *The Visualization Toolkit, 2nd. Ed.*, Prentice Hall, Upper Saddle River, New Jersey, 1998.
16. K.C. Yu, *Multi-Generational Analysis of Anatomical Trees in High-Resolution 3D Images*, Ph.D. thesis, The Pennsylvania State University, 2005.
17. 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.
18. J Z Turlington and W E Higgins, "New techniques for efficient sliding thin-slab volume visualization," *IEEE Transactions on Medical Imaging*, vol. 20, no. 8, pp. 823–835, Aug. 2001.
19. 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.
20. 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.
21. K. C. Tam, G. Lauritsch, and K. Sourbelle, "Filtering point spread function in backprojection cone-beam ct and its applications in long object imaging," *Physics in Medicine and Biology*, vol. 47, no. 15, pp. 2685–703, Aug. 2002.
22. K. Sourbelle, G. Lauritsch, K. C. Tam, F. Noo, and W. A. Kalender, "Performance evaluation of local ROI algorithms for exact ROI reconstruction in spiral cone-beam computed tomography," *IEEE Transactions on Nuclear Science*, vol. 48, no. 3, pp. 697–702, June 2001.
23. R. A. Robb, *AVW Reference Manual*, Biomedial Imaging Resource, Mayo Foundation, Rochester, MN, 1995.
24. 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.