Interactive Extraction of 3D Trees from Medical Images Supporting Gaming and Crowdsourcing

by

Mian Huang

B.Eng., Dalian University of Technology, 2009

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2017

Approval

Name:	**Mian Huang**
Degree:	**Master of Science (Computing Science)**
Title:	***Interactive Extraction of 3D Trees from Medical Images Supporting Gaming and Crowdsourcing***
Examining Committee:	**Chair:** Mark Drew Professor

Ghassan Hamarneh
Senior Supervisor
Professor _____

Ping Tan
Supervisor
Associate Professor _____

Hao (Richard) Zhang
Internal Examiner
Professor
School of Computing Science _____

Date Defended: 20th April 2017

Abstract

Analysis of vascular and airway trees of circulatory and respiratory systems is important for a wide range of clinical applications. Automatic segmentation of these tree-like structures from 3D image data remains challenging due to complex branching patterns, geometrical diversity, and pathology. Existing automated techniques are sensitive to parameters setting, may leak into nearby structures, or miss true bifurcating branches; while interactive methods for segmenting vascular trees are hard to design and use, making them impractical to extend to 3D and to vascular trees with many branches (e.g., tens or hundreds). We propose SwifTree, an interactive software to facilitate this tree extraction task while exploring crowdsourcing and gamification. Our experiments demonstrate that: (i) aggregating the results of multiple SwifTree crowdsourced sessions can achieve more accurate segmentation; (ii) using the proposed game-mode can reduce time needed to achieve a pre-set tree segmentation accuracy; and (iii) SwifTree outperforms automatic segmentation methods especially with respect to noise robustness.

Keywords: vessel extraction; skeletonization; crowdsourcing

Dedication

This thesis work is dedicated to my family and friends.

Acknowledgements

First and foremost, I would like to thank my senior supervisor Ghassan Hamarneh for his mentorship, support, and infinite patience over the past two years. I would also like to thank my other collaborators for their help, feedback, and support on my work.

I would also like to thank my examiners Ping Tan, Hao (Richard) Zhang, and Mark Drew for their feedback on this thesis.

I would further like to thank my fellow students in the Medical Image Analysis Lab for their camaraderie and support throughout my studies. It was a pleasure to share a lab with all of you.

Finally, I would like to thank my family for always supporting me. Any success I achieve is as much due to them as it is due to myself.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Analysis of anatomical branching trees in the human body (i.e., vascular and airway trees of circulatory and respiratory systems) is important for a wide range of applications including understanding normal and pathological anatomy and physiology, clinical diagnoses, and disease research such as planning therapy, tracking diseases, fluid simulation studies, and discovering biomarkers, etc. [31, 4, 40, 57, 49, 63, 58, 38, 20]. These analyses in turn require extracting or delineating (also known as segmenting) the trees from medical images such as magnetic resonance imaging (MRI) or computed tomography (CT). There is also a growing need for large numbers of segmented 3D imaging datasets for training machine learning systems and for validating newly proposed methods but there is a scarcity of segmented 3D trees.

There are numerous methods for segmenting tree-like structures from 2D and 3D medical images. These methods may be classified into different categories depending on the underlying segmentation approach, for example, region growing and active contours [44, 22, 25, 67, 59]; trackers [23, 66, 9, 51, 12, 36, 37]; minimal path [69, 61, 8, 41, 32]; machine learning [43, 42, 71, 72, 70]; graph-based methods [7, 6, 30, 53, 44] and others [55, 73, 3, 21, 66, 23, 26]. For more in-depth reviews of related works, the reader is referred to the following surveys [64, 34, 39, 15, 56, 35, 52]. We note that current fully automatic tree segmentation methods are not yet completely accurate and reliable. It is therefore necessary to have some level of human interaction or intervention to guarantee acceptable segmentation results. To address the need for user-interaction when segmenting anatomical tube-like and tree structures from images, several semi-automatic or interactive tree segmentation methods have been proposed. However, these methods are not without limitations as we describe below.

Vickerman et al. proposed VESGEN 2D [65], which quantifies major vessel parameters within 2D vascular trees, networks, and tree-network composites from 2D binary images without involving too much user input. However, VESGEN requires a pre-segmented (binary) image and does not support 3D.

Live-vessel [54, 17] is an interactive tool that incorporates multi-scale vesselness filtering [24] into the seminal 2D live-wire [5, 27] framework to simultaneously compute optimal medial axes (center-lines) and boundaries of vasculature. It also incorporates color image gradients and quaternion curvature in the segmentation process [60]. However, Live-Vessel is designed for 2D only and it is nontrivial how the user-interaction and visualization may be implemented in a 3D extension.

Heng et al. [28] discussed a 3D intelligent scissors system, in which a user uses a 3D (6 degrees-of-freedom) stylus to specify the endpoints of a branch then a Dijkstra's algorithm is used to find the optimal path between the endpoints. However, it is not always guaranteed that a branch is correctly delineated if the user only provides end points. Also, a 3D stylus is not commonly available for users and its use may not be trivial for controlling the view in 3D space.

Deschamps et al. [16] presented a path tracking method, given only one or two end seed points and the 3D image as inputs, to build a vessel path. The technique makes use of the user-provided seeds points to fit a minimal path. However, it is not always ensured that a branch is correctly described if the user only provides start and finish points for the minimal path, particularly for a noisy and complicated data structure. The user-desired seed points are not easy to find and select using three orthogonal views provided in the tool.

Abeysinghe et al. [1] proposed Gorgon, an interactive tool for identifying skeletons in 3D images. By supplying user inputs, such as simple mouse clicking and scribbling on an isosurface rendered from image volume, the tool guides the creation of skeletons. The method formulates the task of drawing 3D centerlines given 2D user inputs as a constrained optimization problem. This problem is then solved on a discrete graph using a shortest-path algorithm. However, the resulting skeletons are not always connected and smooth. Isosurface rendering could be problematic for data sets with a low signal-to-noise ratio. When the branches overlap, it is difficult for the user to select the desired points by 2D mouse clicking and scribbling.

Diepenbrock et al. [18] proposed a workflow that allows the user to interactively generate vessel segmentations from 3D images. They present the user with multiple linked axis-aligned, orthogonal views and different quality metrics (such as centerline uncertainty). These aids allow the user to assess different aspects of the segmentation and modify it accordingly, e.g., by editing branches, vessel walls, and centerlines. However, the view angle can not be changed directly from this orthogonal slice view, which may limit the user's ability to interrogate the data. Further, the use of mouse clicks, on a 2D screen, makes navigating the 3D scene unintuitive.

Yu et al. [68] proposed a Tree Analyzer system that uses several viewing and interactive tools to examine and correct problems in trees extracted via an automatic tree extraction "Wizard". The navigation and editing requires the user to apply different manipulators (e.g., 3D site locators, pickers, 3D cursor, 3D two-view site locator, and intersection center

locator). However, we found these unintuitive to control and require extensive training. Even though the Tree Analyzer provides a controllable camera view, the user still needs to sketch in 3D space, a challenge for mouse-based interaction. Marks et al. [50] developed Volume Calculator (VolCal) plugin for the ImageJ (`http://imagej.net/Volume_Calculator`). Given a binary (i.e., segmented) image VolCal produces a 3D schematic representation of the tree that can be directly manipulated by the user via mouse clicking. In addition to requiring a segmented image, a challenging task by itself, VolCal also relies on user mouse clicks as input, which are difficult for 3D data.

As described above, airway and vessel segmentation from 3D data remain a challenging task due to complex branching topology, structural (geometrical) diversity and possible pathology. Existing automated techniques are sensitive to parameters setting, may leak into nearby structures (e.g., other vasculature), or miss true bifurcating branches. Interactive methods for segmenting vascular and airway trees in 3D, on the other hand, generally suffer from unintuitive/complex 3D viewing, navigation, user-input or editing, or may conceal errors unless the user visits every point along the branch. We argue that without the user confirming segmentation everywhere along all branches of the tree, suspicion of possible erroneous segmentation regions remain. Therefore, we set out to develop a tool that allows the user to intuitively and quickly traverse the anatomical tree in its entirety in a 3D volume.

The objective of *Gamification* is to transform a mundane task into an immersive and engaging experience. Gamification has been leveraged in many ways, e.g., improving work productivity, patient rehabilitation, education and enhancing cognitive skills, etc [14]. A few works have leveraged gamification for medical image analysis tasks [46, 13, 2, 29]. However, to the best of our knowledge, gamification has never been applied to tree extraction from medical images. Even though it may seem that gaming and serious medical applications are two incompatible domains, we demonstrate otherwise in this work. *Crowdsourcing*, on the other hand, provides a possible source of labelled (so called ground truth) data by leveraging humans' cognitive abilities and intelligence. Crowdsourcing is increasing in popularity and target applications, e.g., missing person search, disaster management, astronomy, rehabilitation, etc. However, only a few works leveraged crowdsourcing to address medical image analysis tasks [48, 47, 11] albeit none of these methods targeted tree extraction.

In this thesis, we present SwifTree, a new interactive method for anatomical tree extraction from 3D medical images that also supports gamification and crowdsourcing. There are three unique aspects of SwifTree that distinguishes it from related works: (i) The operator steers their way down the bifurcating tree branches using intuitive controls; (ii) via crowdsourcing, SwifTree allows multiple users to collaborate and generate several results that are then aggregated to produce the final extracted tree; and (iii) to address the mundane and time-consuming nature delineating many branches, SwifTree resorts to gamification. In game-mode, the tree extraction is realized via game-play, in which the user (or player) navi-

gates the 3D volume using game-like controls (e.g., speed-up, turn-left) by "flying" through branches and identifying bifurcations locations.

To the best of our knowledge, this is the first work to explore gamification and crowdsourcing for vascular or airway tree extraction. This work is a first step towards enabling the collection of large numbers of segmented anatomical trees. Table 1.1 summarizes the most relevant works.

Table 1.1: Comparison of closest works. The meanings of the column headings are as follows. Crowd: method leverages crowdsourcing; Game: offers a "game" mode; 3D: handles 3D data; View: provides a view within the 3D volume; Control: controls the viewing position and angle; Tree: supports extracting branching tree-like structures; Skeleton: extracts centerline; Hierarchy: generates abstract representation of tree hierarchy. *The tree hierarchy generated by SwifTree is user-defined.

Work	Crowd	Game	3D	View	Control	Tree	Skeleton	Hierarchy
Maier-Hein et al. [48]	✓							
Chavez-Aragon et al. [10]	✓							
Donath et al. [19]	✓							
Maier et al. [47]	✓							
Sommer et al. [62]						✓		
Dickie et al. [54]						✓	✓	
Vickerman et al. [65]						✓	✓	✓
Abeysinghe et al. [1]			✓			✓	✓	
Yu et al. [68]			✓			✓	✓	✓
Marks et al. [50]			✓			✓	✓	✓
Edmond et al. [20]			✓	✓		✓	✓	
Heng et al. [28]			✓	✓		✓	✓	✓
Diepenbrock et al. [18]			✓	✓		✓	✓	✓
Proposed SwifTree	✓	✓	✓	✓	✓	✓	✓	✓*

Chapter 2

Methods

A high-level overview of our proposed SwifTree can be seen in Figure 2.1. First, a 3D image selected by the user is loaded into SwifTree. The image is then processed to extract image features. The features are used to control the properties of glyphs placed in a 3D scene to provide helpful cues to the user. Multiple views of the 3D scene are presented to the user by placing virtual cameras at suitable vantage points for visualization (Section 2.1). The user is provided with controls (e.g., keyboard shortcuts) to facilitate navigating through the tree branches within the 3D scene (Section 2.2), to travel, virtually, through the tree branches and construct trees in both a 3D spatial layout and in an abstract graph tree representation. Several features are added to the basic interactive mode for gamification (Section 2.3). In the crowdsourcing setup, multiple users' tree extraction results are aggregated to yield a single tree and graph (Section 2.4). We implemented SwifTree in different platforms (Section 2.5). The details follow.

2.1 Image pre-processing and glyph visualization

We leverage several types of glyphs to highlight different image features. Figure 2.2 shows a schematic of the components that comprise a SwifTree 3D scene. Interpolation is performed to show the basic intensity feature of the image (Section 2.1.1). In addition, we visualize the tree structure including tree boundary (Section 2.1.2) and tree core (Section 2.1.3). We also provide a few auxiliary glyphs (Section 2.1.4).

2.1.1 Image intensity display by interpolating an oblique slice

To display the image intensity, a gray-scale oblique slice (also called multi-planar reconstruction), cutting through the 3D volume, is rendered facing the user's viewing direction. The slice representation is common in different 3D image visualization software. This slice is essentially a planar rectangular polygonal mesh, whose length is proportional to the length of the diagonal of the image. Tri-linear interpolation based on the intensity values of the 3D

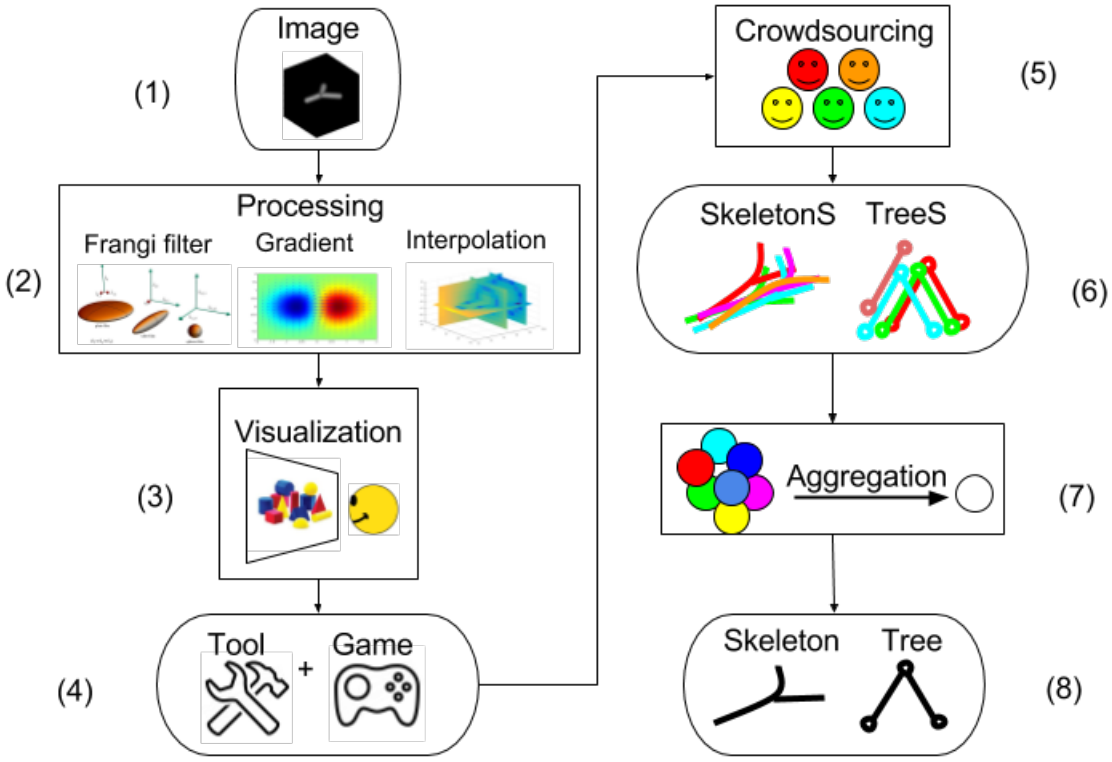

Figure 2.1: SwifTree work-flow: (1) A 3D image is loaded. (2) The image is processed to extract interpolated intensity, gradient vectors, and Frangi filter based features. (3) Different glyphs are used to visualize cues for the user based on the extracted features. (4) Based on the visualization and controller, interactive tool and game mode are available to select. (5) One or more users or players are recruited via crowdsourcing and take part in the tree extraction session. (6) The results generated from all the sessions are collected. (7) The results are aggregated into one solution. (8) Final single 3D tree and graph representation is acquired.

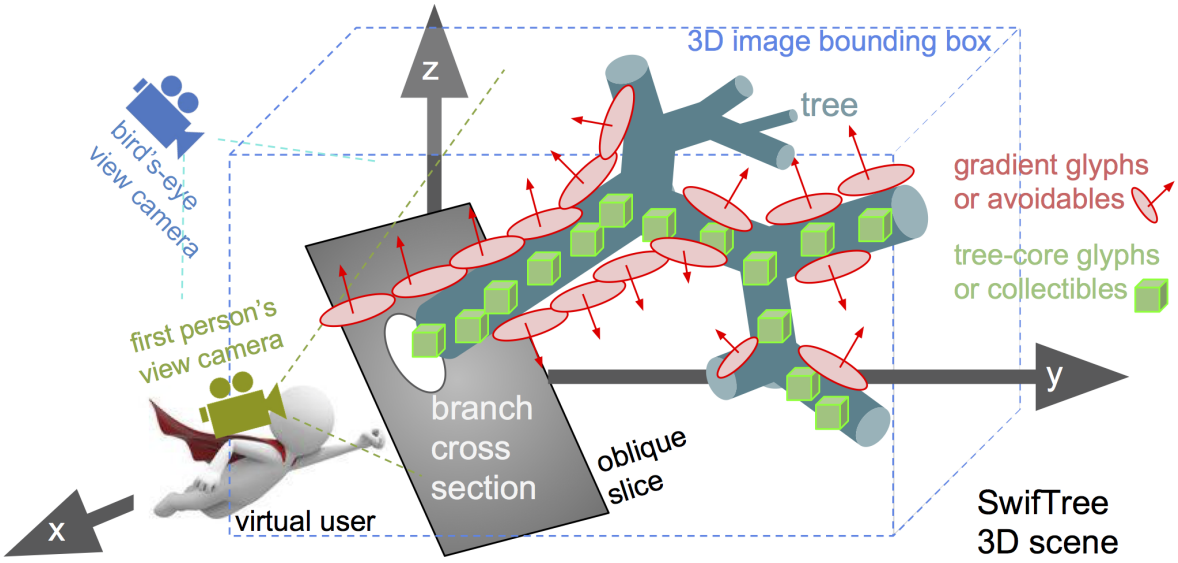

Figure 2.2: Elements of SwifTree 3D scene (see text of Section 2.1)

image volume is performed at the center position $[x, y, z]$ of each of the faces of the mesh in order to assign an intensity value to that face. Next, we describe how the interpolation is realized.

First we find the eight corners of a cube that surround the face center $[x, y, z]$. We denote the image intensity values at these corners by I_{000}, I_{100}, I_{010}, I_{110}, I_{001}, I_{101}, I_{011}, and I_{111}. Let x_d, y_d, and z_d be defined as follows:

$$x_d = (x - x_0)/(x_1 - x_0), y_d = (y - y_0)/(y_1 - y_0), z_d = (z - z_0)/(z_1 - z_0),$$

where x_0 indicates the closet x coordinate of the 3D image volume lattice point (i.e., voxel centre) to x from below, i.e., $x_0 \leq x$, and similarly x_1 is the closest x coordinate to x from above, i.e., $x_0 \geq x$. y_0, y_1, z_0 and z_1 are defined in a similar way.

Next, we perform linear interpolation between I_{000} and I_{100}, I_{001} and I_{101}, I_{011} and I_{111}, I_{010} and I_{110}. We interpolate along x (imagine we are pushing the front face of the cube to the back), so we have: $I_{00} = I_{000}(1 - x_d) + I_{100}x_d$, $I_{01} = I_{001}(1 - x_d) + I[x_1, y_0, z_1]x_d$, $I_{10} = I_{010}(1 - x_d) + I[x_1, y_1, z_0]x_d$, $I_{11} = I_{011}(1 - x_d) + I[x_1, y_1, z_1]x_d$, where I_{000} means the intensity value of (x_0, y_0, z_0).

Then we do interpolation between I_{00} and I_{10}, I_{01} and I_{11}. We interpolate these values (along y, as we were pushing the top edge to the bottom), so we have: $I_0 = I_{00}(1-y_d)+I_{10}y_d$, $I_1 = I_{01}(1 - y_d) + I_{11}y_d$.

Finally, we calculate the value I_c via linear interpolation of I_0 and I_1. We interpolate these values along z (walking through a line): $I_c = I_0(1-z_d)+I_1z_d$. This gives us a linearly predicted value for the point. The above operations can be visualized in Figure 2.3

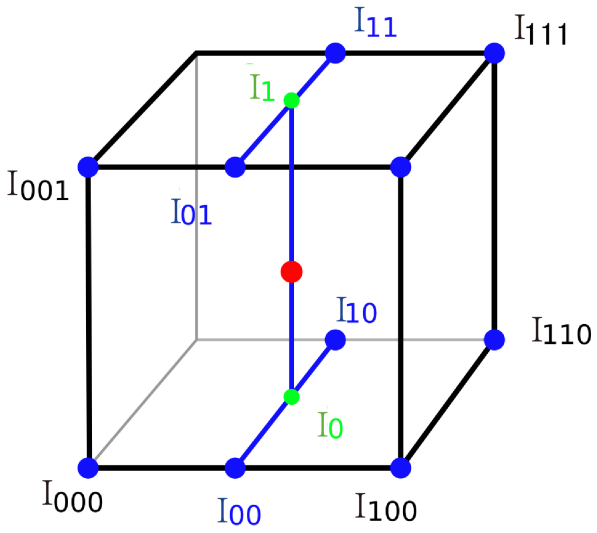

Figure 2.3: Interpolation

To display the interpolated intensity, we consider a mapping in general:

$$P = aF^b \times P_C, \tag{2.1}$$

where P is a mapped value of a certain property, a and b are scalars controlling the linear and nonlinear mappings respectively, F represents a certain feature, P_C is a constant base value of the particular property. We use Equation 2.1 to map the interpolated image intensity to a gray color: $c_s = aI_c^b \times [255, 255, 255]$, where c_s is the mapped color, I_c is the interpolated intensity and $[255, 255, 255]$ denotes the base value of the property as RGB (to make c_s a gray scale color). The mapped gray scale color c_s is applied as the texture onto the corresponding mesh face. The slice would depict the cross-section of a branch as a single bright disk. As the user moves towards a bifurcation, the disk gradually splits into two, one for each child branch. A simple chart showing this cross section splitting pattern can be seen in Figure 2.4. In addition to the intensity based slice, we provide other two types of glyphs to emphasize two key image features to augment the visualization.

2.1.2 Tree boundary glyphs by gradient calculation

We also render 2D polygonal gradient glyphs based on the 3D image intensity gradient to highlight an estimate of the surface boundary surrounding the tree branches. The size and opacity of the polygons are proportional to the gradient magnitude while the normal vector for the polygon points in the gradient direction. We first calculate the gradient vector field:

$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial z}\right] = [I_x, I_y, I_z], \tag{2.2}$$

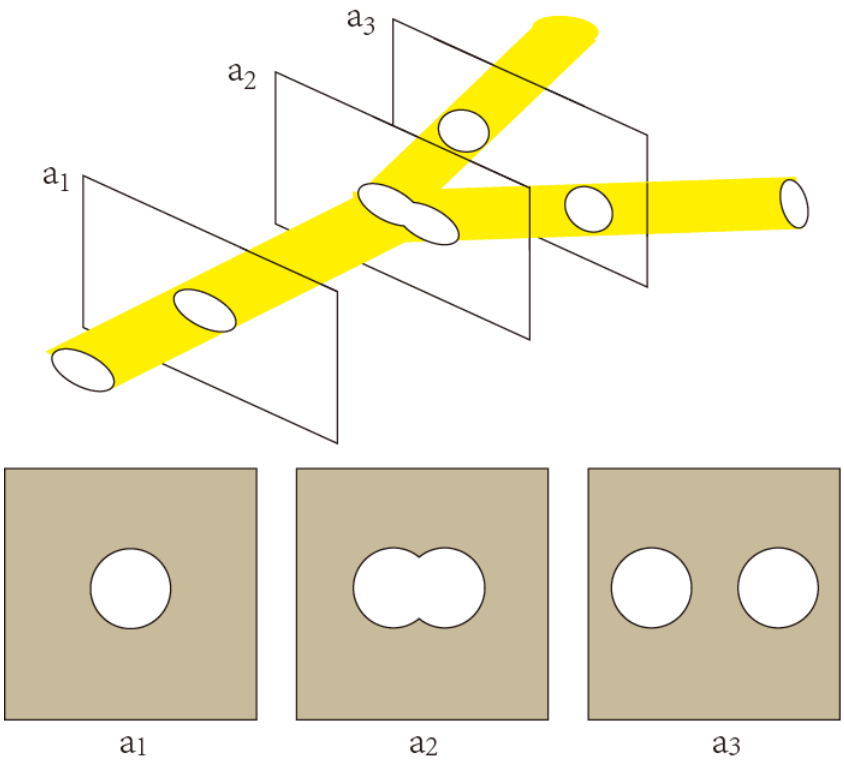

Figure 2.4: Oblique slices through the 3D volume depicting the cross sectional view of a bifurcation (i.e., as the parent branch splits into two child branches)

$$\text{with } I_x = \frac{I(x+\Delta x, y, z) - I(x-\Delta x, y, z)}{2},$$

where ∇I is the gradient of the intensity, $\frac{\partial I}{\partial x} = I_x$ is the gradient of intensity along x, $I(x, y, z)$ is the intensity at the location (x, y, z), Δx is the difference between x and the closest point along the $x-axis$. We examine both the magnitude and direction of the image gradient ∇I at each voxel and use them to control the appearance of the tree boundary glyph. We first derive two vectors:

$$\begin{cases} n_1 = [I_y, -I_x, 0]; \\ n_2 = \nabla I \times n_1 = \left[-I_z \cdot I_x, -I_y \cdot I_z, I_x^2 + I_y^2\right], \end{cases} \tag{2.3}$$

where n_1 and n_2 are two vectors perpendicular to each other and to the gradient direction ∇I. At each voxel, we visualize the tree boundary glyph as a circular disc. The radius of the disc is proportional to the gradient magnitude at this voxel. The face normal of the disc is set to be the gradient direction $\nabla I/||\nabla I||$. The circular disc is approximated by a polygon with vertices set in the 2D space spanned by $n_1 = n_1/||n_1||$ and $n_2 = n_2/||n_2||$:

$$v^i = n_1 \cos(2\pi/i) + n_2 \sin(2\pi/i), \tag{2.4}$$

where v^i is the location of the ith vertex of the polygon (Figure 2.5).

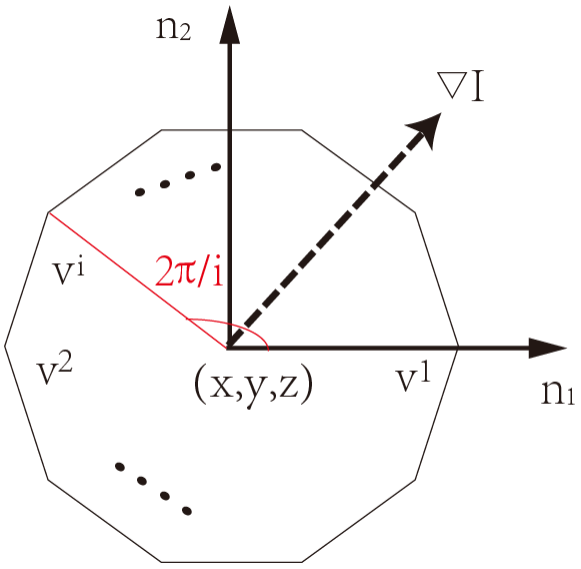

Figure 2.5: The polygon generated according to the gradient

To make the size of the polygon proportional to the gradient magnitude, we specify Equation 2.1 such that $v^i_{local} = a||\nabla I(x, y, z)||^b \times v^i$. Then we make the polygon at location

$[x, y, z]$:

$$v^i_{global}(x, y, z) = v^i_{local} + [x, y, z], \tag{2.5}$$

where $v^i_{global}(x, y, z)$ is the coordinate of ith vertex of a polygon whose center is located at (x, y, z). The color and the opacity of the polygon are also related to the magnitude of the gradient $||\nabla I||$ at each voxel. We re-use Equation 2.1 such that the color $c_a = a||\nabla I(x, y, z)||^b \times [255, 0, 0]$, and the opacity of the polygon can be calculated similarly using Equation 2.1 with $\alpha_a = a||\nabla I(x, y, z)||^b$. The mapped color and opacity value are used as texture applied to the polygons.

2.1.3 Tree core glyphs by vesselness filtering

To highlight the voxels in the interior of tree branches, we use 3D tree-core glyphs, whose size and opacity are proportional to the tubularness response as calculated using the Frangi filter [24]:

$$I_V(s) = \begin{cases} 0; \\ (1 - \exp(-\frac{R_A^2}{2\alpha^2}))\exp(-\frac{R_B^2}{2\beta^2})(1 - \exp(-\frac{S^2}{2c^2})), \end{cases} \tag{2.6}$$

where $I_V(s)$ is the vesselness scalar field obtained by the Frangi filter; exp is the exponential function; R_A differentiates between plate- and line-like structures in the image, R_B measures deviation from blob-like structures, and S emphasizes areas of high contrast. α, β and c are scalars that control the sensitivity of the filter to the measures R_A, R_B and S.

A 3D image is acquired with each voxel storing the vesselness value $I_V(s)$. We visualize the tree core glyphs as a diamond shaped object: At a particular voxel, the vesselness value $I_V(s)$ is mapped to a yellowish color as $c_c = a||I_V(s)||^b \times [255, 255, 0]$, opacity as $\alpha_c = a||I_V(s)||^b$ and size as $s_c = a||I_V(s)||^b$. The mapped color c_c and opacity α_c are applied as the texture onto the diamond object while the mapped size s_c is used as the length of the diamond diagonal.

2.1.4 Auxiliary glyphs

The user interrogates different locations within the volume that are indicated by a 3D polyhedral cursor. The user's task is to 'label' the voxels that make up the tree by controlling the cursor position to touch those voxels. 'On' and 'off' attributes are assigned to voxel to indicate when an a volxel is within-tree or not. The voxels which are not labeled specifically are treated as not within-tree automatically. To convey to the user which branches are already explored, we visualize the path generated from the cursor as a dark green trail along the traversed voxels. These path are collected to build the tree. Additionally, two virtual cameras are added to the scene: one camera provides a first-person local view whereas the other displays a more global bird's-eye view. The former is attached to the cursor with a fixed distance of a few voxels away. The view direction of this camera is set to be the vector

from the position of the cursor to the camera. So the camera will be always 'looking' at the position of the cursor to provide a first-person local perspective view. The latter camera is located at a fixed position on top of the image volume. The view orientation of this camera is from the center of the the camera to the image, which is chosen to provide the bird's eye view, i.e., a global view of the whole scene.

2.2 Navigating the cursor in 3D space for tree extraction

The aforementioned 3D cursor can be moved and rotated interactively by the user (e.g., move-forward, rotate-left, etc.). In this way, the user controls the cursor to label the voxels in the 3D volume as foreground (tree branches) or background (not tree branches).

2.2.1 Movement

First, the cursor can be moved forward, backward, up, down, left and right. To keep a fixed distance between the oblique slice (Section 2.1.1), the first-person camera and cursor (Section 2.1.4), their movement can be applied by an affine transformation as follows:

$$P_H(t+1) = \alpha_H(t) T_H(t) P_H(t), \tag{2.7}$$

$$\text{with } T_H(t) = \begin{bmatrix} I & \vec{v}_H(t) \\ \vec{0} & 1 \end{bmatrix},$$

where $P_H(t+1)$ denotes the position of the object ($H = a$ for the cursor, $H = c$ for the camera and $H = s$ for the slice) at the time instant $t+1$, $\alpha_H(t)$ is a scalar indicating the moving stride size of the object at the time t. $T_H(t)$ is the translation matrix of the movement. $P_H(t)$ are the coordinates of vertices of the object. In the translation matrix $T_H(t)$, $\vec{v}_H(t)$ is a unit vector representing the direction of the movement (forward, backward or panning according to the cursor's up vector). I is a 3×3 identity matrix and $\vec{0}$ is a 1×3 vector of zeros. The movement is implemented by the user's clicking on the keyboard. Effectively, this equation changes the coordinate of the objects from $P_H(t)$ at time t to the new coordinates $P_H(t+1)$ at time $t+1$. The tri-linear interpolation (Section 2.1.1) is performed every time the slice position changes.

2.2.2 Rotation

Additionally, we provide a functionality to rotate the cursor so that it can turn into the user-desired direction. To keep the relative orientation between the camera, slice and cursor, their rotation is calculated by:

$$P_H(t+1) = T_p(t) R(t) T_p^{-1}(t) P_H(t), \tag{2.8}$$

$$\text{with } T_p(t) = \begin{bmatrix} I & \vec{p}_C(t) \\ \vec{0} & 1 \end{bmatrix},$$

$$R(t) = \begin{bmatrix} \cos\theta + u_x(t)^2\,(1-\cos\theta) & u_x(t)u_y(t)\,(1-\cos\theta) - u_z(t)\sin\theta & u_x(t)u_z(t)\,(1-\cos\theta) + u_y(t)\sin\theta \\ u_y(t)u_x(t)\,(1-\cos\theta) + u_z(t)\sin\theta & \cos\theta + u_y(t)^2\,(1-\cos\theta) & u_y(t)u_z(t)\,(1-\cos\theta) - u_x(t)\sin\theta \\ u_z(t)u_x(t)\,(1-\cos\theta) - u_y(t)\sin\theta & u_z(t)u_y(t)\,(1-\cos\theta) + u_x(t)\sin\theta & \cos\theta + u_z(t)^2\,(1-\cos\theta) \end{bmatrix},$$

where $P_H(t+1)$ is as defined above. $T_p(t)$ is the translation matrix from the origin i.e., $[0,0,0]$ to the cursor center position $\vec{p}_C(t)$ at the moment t. $R(t)$ is the rotation matrix at time t. In the translation matrix $T_p(t)$, $\vec{p}_C(t)$ is the center position of the cursor at the time t. I and $\vec{0}$ are as defined above. In the rotation matrix $R(t)$, θ is the angle of the rotation. $u(t) = [u_x(t), u_y(t), u_z(t)]$ is the axis to rotate around: when $u(t)$ is set to be $u(t) = \pm u_U(t)$ to turn left or right, where $u_U(t)$ is the cursor's up vector; when $u(t)$ is set to be $u(t) = \pm(v_C(t)) \times u_U(t)$ to turn up or down, where $u(t)$ is the vector perpendicular to the cursor's up vector $u_U(t)$ and first-person camera viewing direction $v_C(t)$. The tri-linear interpolation (Section 2.1.1) is performed every time the slice orientation changes. A simple diagram shows the terminology of the rotation in Figure 2.6.

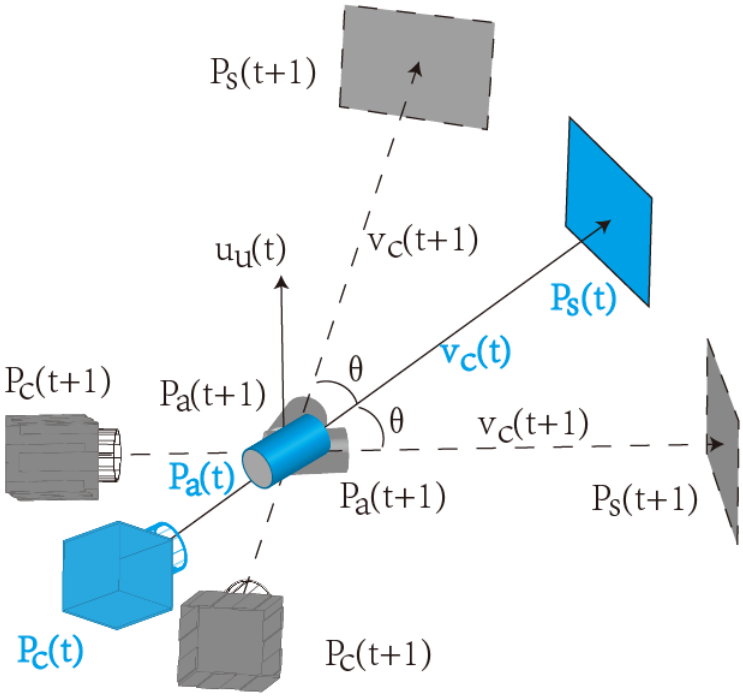

Figure 2.6: Rotation of the cursor, camera and slice

2.2.3 Handling bifurcations and graph tree generation

Once the user encounters a bifurcation while travelling along a branch, i.e., by observing the branch cross-section splitting (a simple diagram showing the splitting pattern can be seen in Figure 2.4), they press a key to *push* the current state parameters (i.e., location and camera viewpoints) into a *bifurcation stack*. After the user traverses one of the child branches (and optionally the grandchild branches), they *pop* the state parameters, to move the cursor and cameras back up the tree hierarchy to a previously-identified bifurcation location, so that the other child branches can be explored. In particular, after one child branch of the bifurcation is visited, the cursor is moved back to the bifurcation location immediately and automatically to visit the other branch. At the same time, a tree structure (graph tree) is generated as described in Algorithm 1.

An example of a graph tree growing procedure can be seen in Figure 2.7.

Algorithm 1 Tree Generating Algorithm

Input: $K(t)$ {Key pressed at the moment t}
Output: Tree graph T

Initialization
$i \leftarrow 1$, {the current node's index sets to be 1}
$T(i).father \leftarrow 0$, {Node i's father sets to be Node 0 (root)}
$T(i).visit \leftarrow 1$, {Node i's visited time sets to be 1}
while $t \leq t_{total}$ **do**
  if $K(t) == S$ **then**
    $T(i+1).father \leftarrow i$, {Node $(i+1)$'s father sets to be Node i}
      $T(i+1).visit \leftarrow 1$, {Node $(i+1)$'s visited time sets to be 1}
      $i \leftarrow i+1$,
  else if $K(t) == R$ **then**
    $i \leftarrow T(i).father$, {the current Node i sets to be Node i's father}
    while $T(i).visit > 1$ **do**
      $i \leftarrow T(i).father$, {the current Node i sets to be Node i's father}
    end while
    $T(i).visit \leftarrow T(i).visit + 1$, {current Node i's visited time increases by one}
  end if
end while

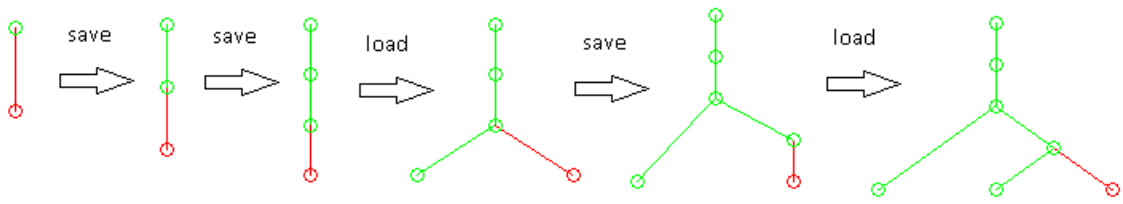

Figure 2.7: An example of the graph tree growing progress

2.2.4 Auxiliary controller

There are several auxiliary controllers to help the user navigate and perform tree extraction. The first-person's view may show fewer or more branches. The user can zoom in to focus their attention on the particular branch they are tracking, or zoom out to take a look at the neighbouring branches. In addition, the user can choose what they want to see from the scene including the first-person's view and bird's-eye view: all the glyphs (Section 2.1) can be switched on and off based on the user's choice. The tree graph shown in the tree graph window is also growing interactively with the user's key strokes. The extraction can be turned on and off by the user to label the voxel as tree and non-tree by travelling through them. The stride size of movement can also be set according to the user's preference.

2.3 Interactive and game mode

We call the mode described above a 'non-game interactive tool' (or 'interactive' for short) mode. We also provided a 'game' mode to complete the tree extraction.

In SwifTree's game mode, a few things are different compared to interactive tool mode: The cursor is an avatar that possesses a velocity controlled by the player. The player navigates the 3D volume by 'flying' through branches and identifying bifurcation locations using game-like controls (e.g., speed up, slow down, turn left). Also in game mode, the tree-core glyphs are set to be *collectibles* (Figure 2.2), i.e., as the user's cursor passes over these glyphs, they are collected and hidden with an accompanying sound effect and a score increment. The gradient glyphs, on the other hand, are *avoidables* (Figure 2.2), since they represent branch boundaries. In SwifTree's non-game interactive mode, the user's cursor can be seen as an inertia-less paintbrush manipulated by the user.

2.4 Crowdsourcing and aggregation

We recruit multiple users or players to carry out a tree extraction session. The collected tree branches for the same image across all sessions are first unioned together and then a 3D spherical kernel is used to perform morphological closing. Then a medial axis transform is applied to extract the tree skeleton and network analysis is performed to create the abstract graph tree representation [33]. To ensure a robust and reliable tree given possible errors made by the users/players, we examine consensus among the results. The tree branches are first union among users/players:

$$S_U = \bigcup_{i=1}^{n_U} S_i, \tag{2.9}$$

where S_U is the union from all the sessions. n_U is the number of trees generated from all the sessions and S_i is the the tree from the ith session. To smooth the union S_U, a unity-sum

3D kernel K is convolved with the union result S_U:

$$S_S = S_U * K. \tag{2.10}$$

Then S_S is thresholded to create a binary image. After that, a medial axis skeletonization and network analysis [33] are performed on the binary image to achieve the aggregated skeleton tree and graph tree.

2.5 Implementation details

We used MATLAB (R2015b) to test several visualization and interaction mechanisms. Then we ported SwifTree to: (i) the cross-platform game engine Unity3D (`unity3d.com`). A snapshot of the implementation can be seen in Figure 2.8; and (ii) an online cross-browser version using JavaScript (v6.0) and the WebGL-based 3D graphics library Three.js (r83) (`threejs.org`), with PHP and MySQL to automatically collect the tree segmentation data generated by the users.

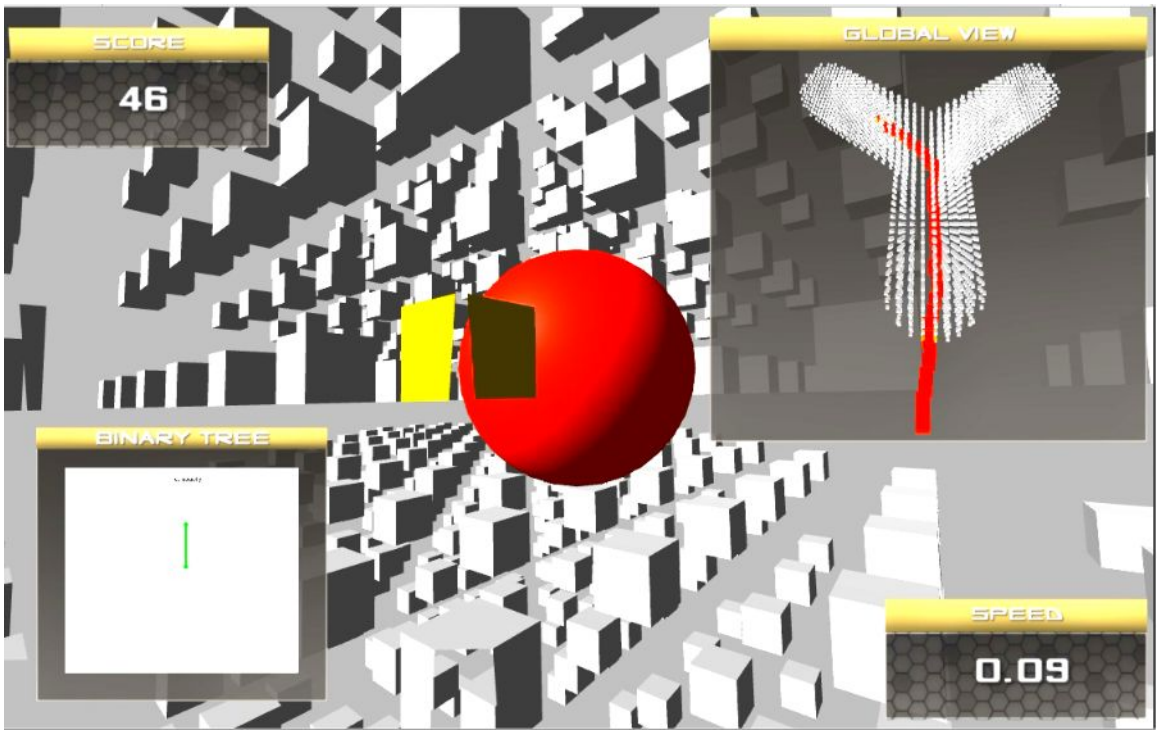

Figure 2.8: Implementation of SwifTree by Unity3D

The user is provided with a user manual that explains how the tool should be used (Appendix A). The manual explains how the user should load an image and, optionally, load a pre-saved session, and begin the extraction process. The manual also introduces the elements of the GUI, the functionality provided by different controllers, and how the tree

extraction should be performed using the GUI and controllers. The manual also explains how the user should deal with bifurcations to visit all child branches. The user is also prompted to save the extraction session. Finally, the user is shown a summary of the time and keystrokes they have used during the extraction session. More details are available in the user manual (Appendix A).

After launching the software, the user is asked to load an image.

Chapter 3

Data

Sample slices of the 3D volumetric datasets used in our experiments can be seen in Figure 3.1, which include both synthetic and real in-vivo image datasets:

- In-silico Phantom (computer simulated trees) including synthetic vascular images (Y-Junc and Helix) and Vascusynth (`http://vascusynth.cs.sfu.ca/Data.html`),
- Physical Phantom Luboz,
- Real data including Renal Vasculature MRA (Magnetic Resonance Angiogram) and Brain Vasculature CTA (Computed Tomography Angiography) from OSIRIX dataset (`http://http://www.osirix-viewer.com/resources/dicom-image-library/`), as well as Airway CT (Computed Tomography) from (`http://image.diku.dk/exact/`).

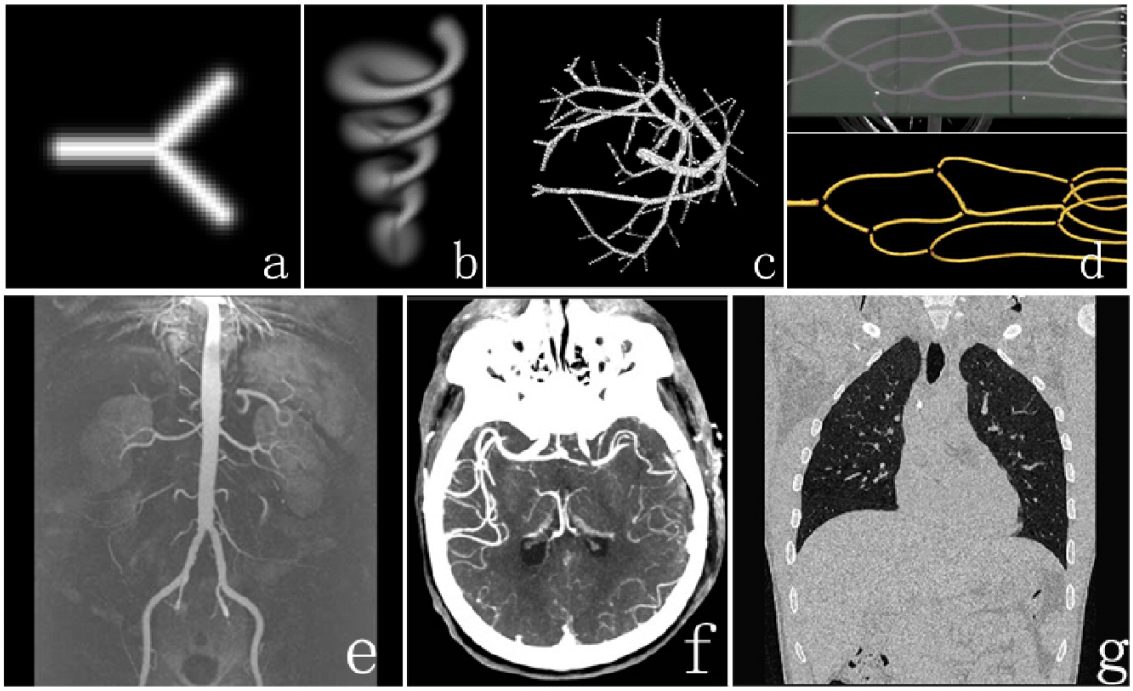

Figure 3.1: Datasets: (a-c) In-silico phantoms: Y-Junc (60x60x60), Helix (50x50x100), and Vascusynth (101x101x101); (d) Physical Phantom (168x168x159); (e) Kidney MRA (576x448x72); (f) Brain MRA (352x448x176); (g) Airway CT (512x512x587).

Chapter 4

Results

4.1 Qualitative Results

A user manual on how to use SwifTree is attached in the Appendix A. The graphical user interface (GUI) of SwifTree consisting of three windows is shown in Figure 4.1. The window on the left displays the first-person view in the 3D space. The upper right window illustrates the bird's-eye view showing a global broad view of the whole scene. The slice, discs and diamonds objects indicate the intensity, gradient and vesselness features, respectively. The lower right window presents the tree growing process. The progress of the 3D tree extraction and the abstract tree graph construction for the different datasets using SwifTree are shown in Figure 4.2.

We collected results completed by 10 tree extraction sessions for each image using SwifTree (Figure 4.3). The 10 sessions are aggregated to obtain a single tree for each image. Figure 4.4 shows the aggregated tree overlapping with the original image. The reader is referred to a simplified and anonymized web-based version of SwifTree at `http://79.170.44.151/swiftree.org/` and to the supplementary video and images.

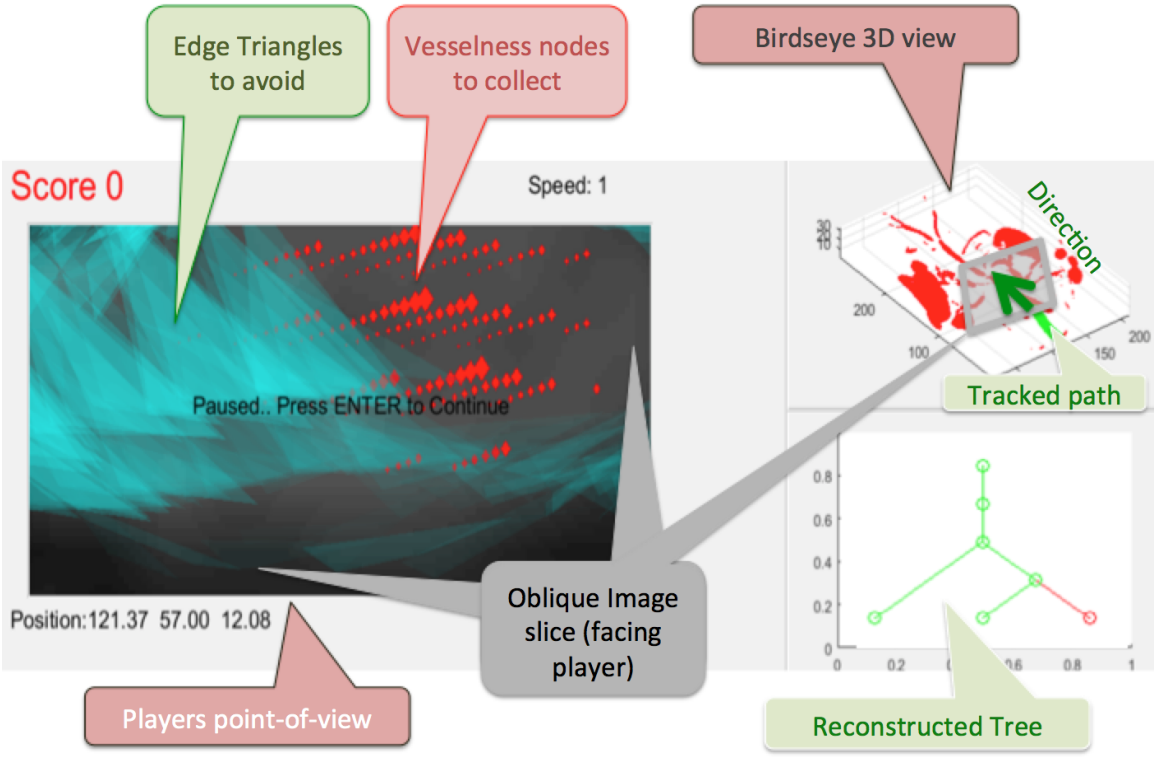

Figure 4.1: The GUI of SwifTree

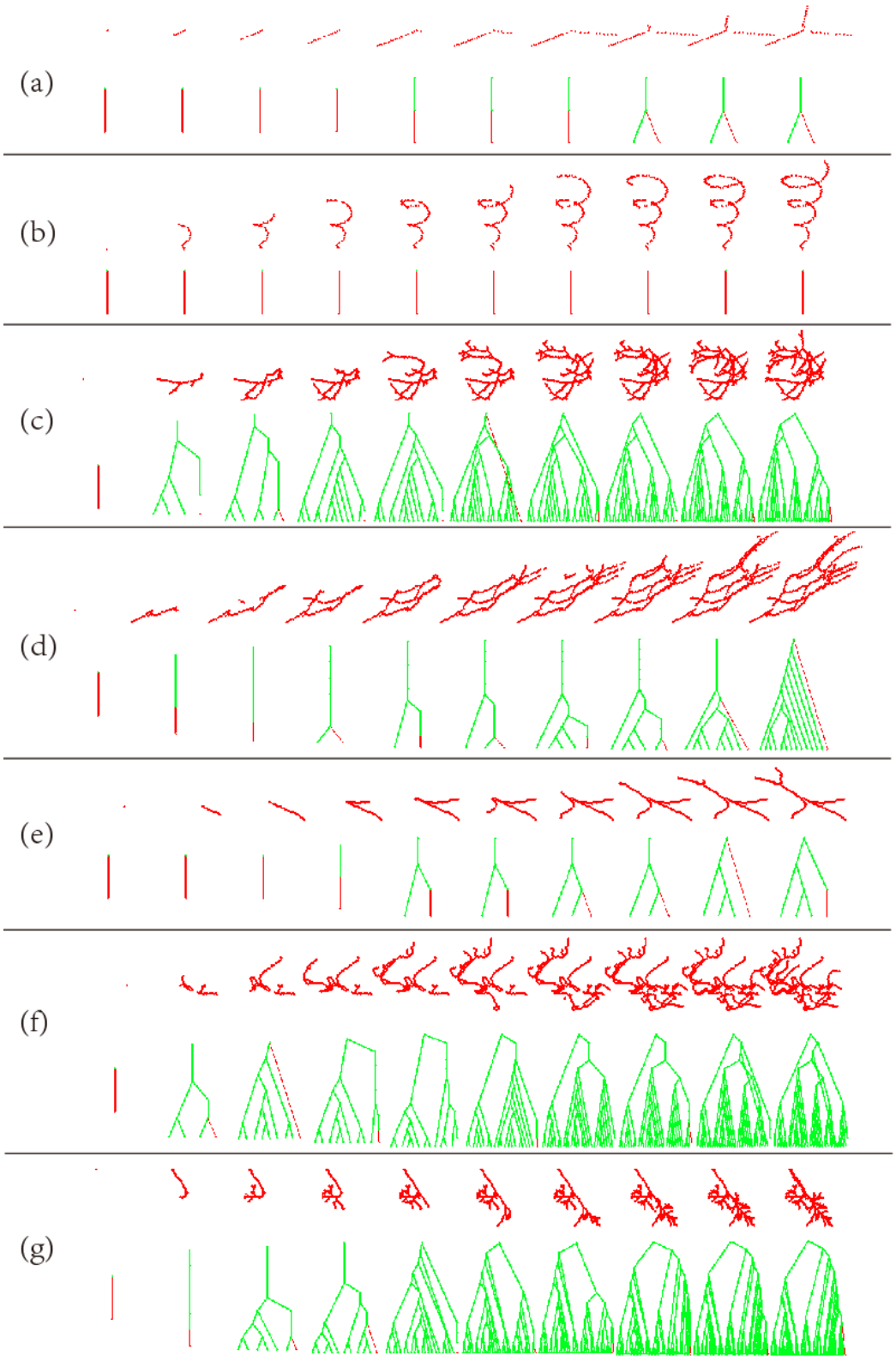

Figure 4.2: Illustration of the sequence of steps (from left to right) used to extract a 3D tree using SwifTree for: (a)Y-Junc; (b)Helix; (c)Vascusynth; (d)Phantom; (e)Kidney; (f)Brain; and (g)Airway. Top: 3D spatial domain; bottom: corresponding abstract tree graph.

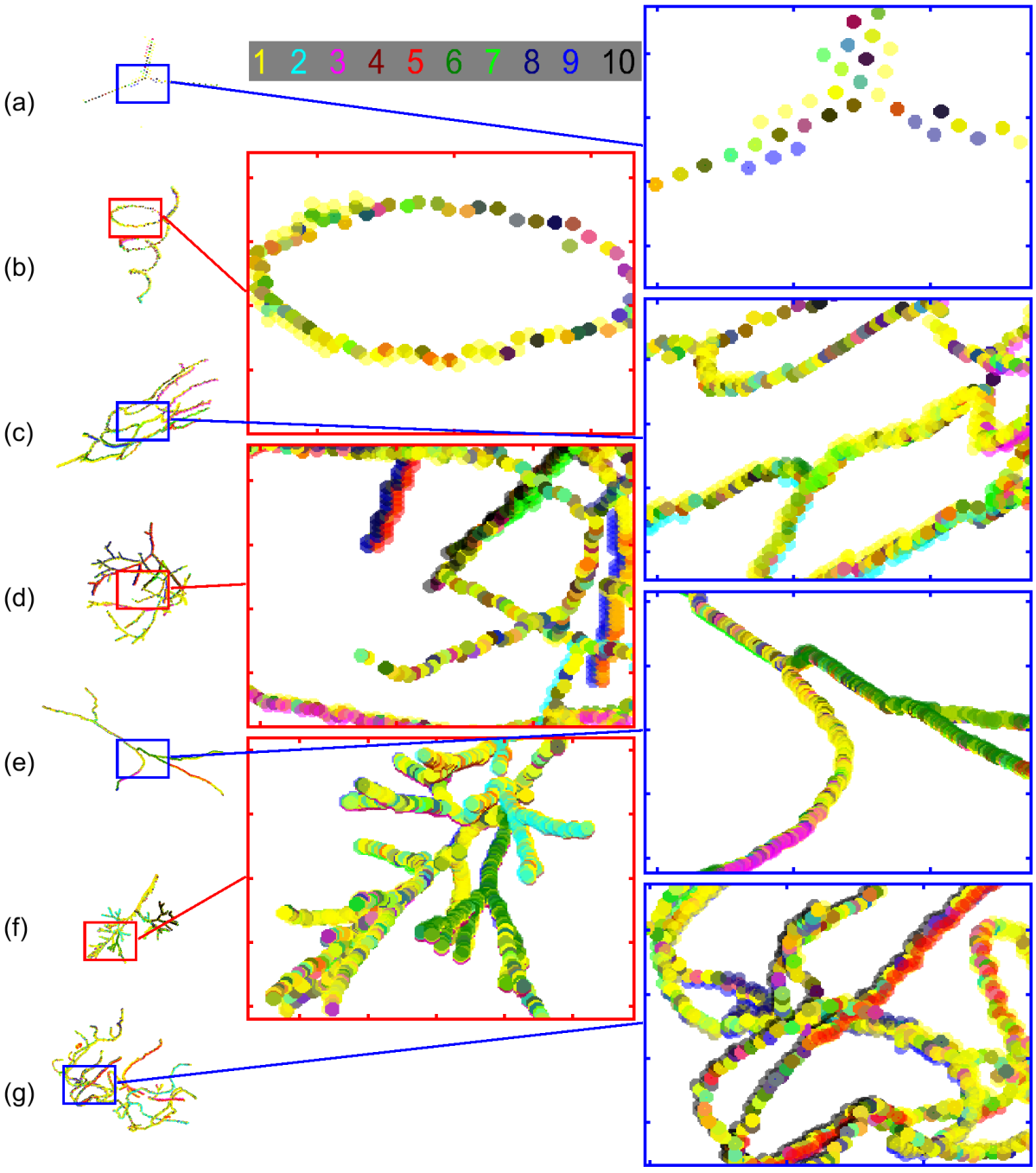

Figure 4.3: Visualization of crowdsourcing 10 sessions of tree extraction by SwifTree for the image (a)Y-Junc, (b)Helix, (c)Phantom, (d)Vascusynth, (e)Kidney, (f)Airway, and (g)Brain. Different sessions are shown in different colors.

.

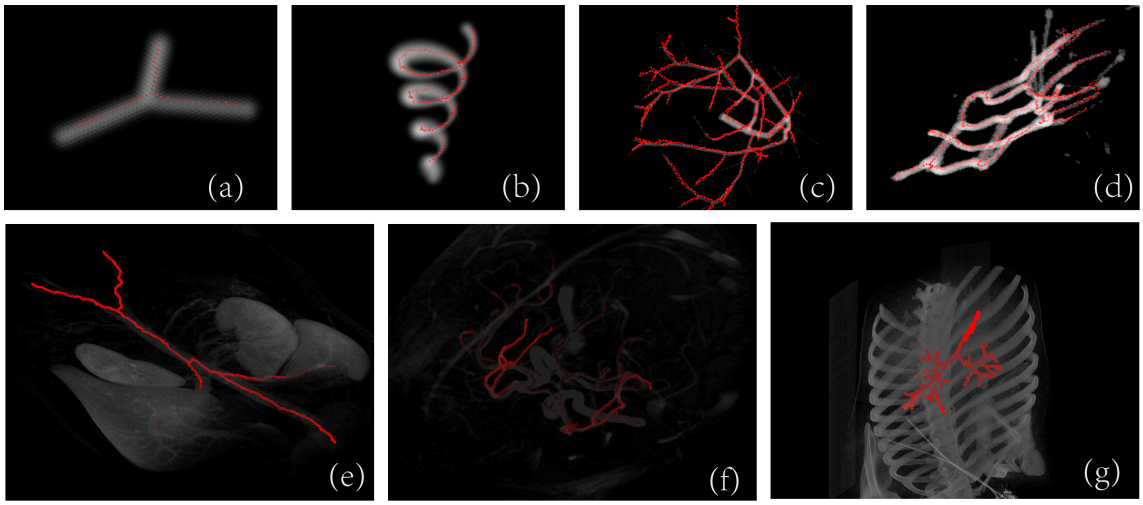

Figure 4.4: The aggregated tree from 10 SwifTree sessions' tree extraction overlaid on the original images: (a) Y-Junc, (b) Helix, (c) Vascusynth, (d) Phantom, (e) Kidney, (f) Brain, and (g) Airway.

4.2 Quantitative Results

To evaluate the performance of the tree extraction tool, we consider not only the accuracy of the tree extracted from them but also the time and effort (number of mouse and keyboard clicks). To quantify the accuracy, we adopt the criteria as described in [45]. Namely, we measure the following: a) Branch count (BC): The number of branches that are detected correctly. A branch is considered detected as long as the length of the centerlines is more than 1 mm. b) Branches detected (BD): The percentage of branches that are detected correctly with respect to the total number of branches present in the reference. c) Tree length (TL): The sum of the length of the centerlines of all correctly detected branches. d) Tree length detected (TLD): The fraction of tree length that is detected correctly relative to the total tree length in the reference e) Leakage count (LC): The number of unconnected groups of 'correct' regions that are neighboring with a 'wrong' region, indicating how easy/difficult it is to manually separate leakages from the correctly detected branches. f) False positive rate (FPR): The fraction of the voxels passed by extracted tree that is not marked as 'correct' in the reference.

4.2.1 Tree extraction accuracy

Table 4.1 compares SwifTree to ITK-Snap (`itksnap.org`) and Gorgon (`gorgon.wustl.edu`). We see that SwifTree achieves the highest BD accuracy for all the datasets, the highest TLD for all the datasets except Phantom, and the lowest FPR for all the datasets except Airway.

We also summarize the time and effort by ITK-Snap, Gorgon and SwifTree. Table 4.2 and Figure 4.5 show the time consumption and number of user clicks for tree extraction by ITK-Snap, Gorgon and SwifTree, respectively.

To consider both the accuracy and effort together, Figure 4.6 gives an overview of the performance of the different tools using a scatter plot of tree length detected versus time expenditure.

4.2.2 Benefit of crowdsourcing

To assess the value of crowdsroucing, in this experiment, we test crowdsourcing extension of SwifTree by comparing non-crowdsourcing and crowdsourcing extraction. We collected the results from 10 tree extraction sessions performed by one single user for each dataset using SwifTree (i.e., 70 sessions). We waited for all the sessions to start aggregating at the same time.

Figure 4.8 top shows the tree extraction for Brain, with the branches color coded according to the index of sessions that detected them. Figure 4.8 bottom shows the line plots of the percentage of tree length detected vs time consumption for the sessions that extracted

Table 4.1: The accuracy of tree extraction by ITK-Snap, Gorgon and SwifTree. Highest accuracy in bold. '*': unit is cm. '-': tree extraction was not feasible.

Image	Tool	BC	BD	TL	TLD	LC	FPR
Y-Junc	Gorgon	1	33.33%	3.00	5.56%	2	85.71%
	ITK-Snap	**3**	**100.00%**	46.49	86.18%	**1**	1.89%
	SwifTree	**3**	**100.00%**	**48.70**	**90.28%**	**1**	**1.11%**
Helix	Gorgon	1	33.33%	1.41	0.48%	**1**	97.88%
	ITK-Snap	**3**	**100.00%**	152.86	51.84%	35	37.99%
	SwifTree	**3**	**100.00%**	**222.58**	**75.49%**	5	**1.37%**
Vascusynth	Gorgon	27	24.55%	400.96	28.72%	**85**	72.92%
	ITK-Snap	58	52.73%	747.60	53.56%	273	59.09%
	SwifTree	**87**	**79.09%**	**988.09**	**70.79%**	152	**14.71%**
Phantom	Gorgon	28	43.08%	564.18	48.60%	98	43.06%
	ITK-Snap	47	72.31%	**896.64**	**77.24%**	45	9.64%
	SwifTree	**52**	**80.00%**	841.35	72.48%	**27**	**4.42%**
Kidney	Gorgon	5	21.74%	19.69 (cm)	27.20%	9	79.18%
	ITK-Snap	13	56.52%	40.39 (cm)	55.80%	57	42.61%
	SwifTree	**21**	**91.30%**	**47.92** (cm)	**66.21%**	**7**	**4.90%**
Brain	Gorgon	-	-	-	-	-	-
	ITK-Snap	30	24.00%	34.54 (cm)	30.48%	**82**	19.63%
	SwifTree	**82**	**65.60%**	**64.54** (cm)	**56.96%**	144	**12.42%**
Airway	Gorgon	-	-	-	-	-	-
	ITK-Snap	57	19.32%	28.45(cm)	17.33%	**81**	**11.21%**
	SwifTree	**151**	**51.19%**	**91.00**(cm)	**55.42%**	284	19.73%

Table 4.2: Time consumption and number of clicks by SwifTree, ITK-Snap and Gorgon. The fewest clicks and lest time consumption among tools are colored in bold for each data. '-' indicates tree extraction is not feasible by Gorgon for the image Brain and Airway

	Time			Effort		
	Gorgon	ITK-Snap	SwifTree	Gorgon	ITK-Snap	SwifTree
Y-Junc	4:13	2:10	**00:51**	144	299	**43**
Helix	16:06	8:51	**04:09**	279	721	**215**
Phantom	**9:27**	20:13	16:31	**222**	904	868
Vascusynth	**6:02**	22:26	17:40	**135**	871	1049
Kidney	12:21	**3:55**	17:21	**125**	233	989
Airway	-	**15:21**	37:23	-	**621**	2017
Brain	-	**13:36**	26:43	-	**537**	1352

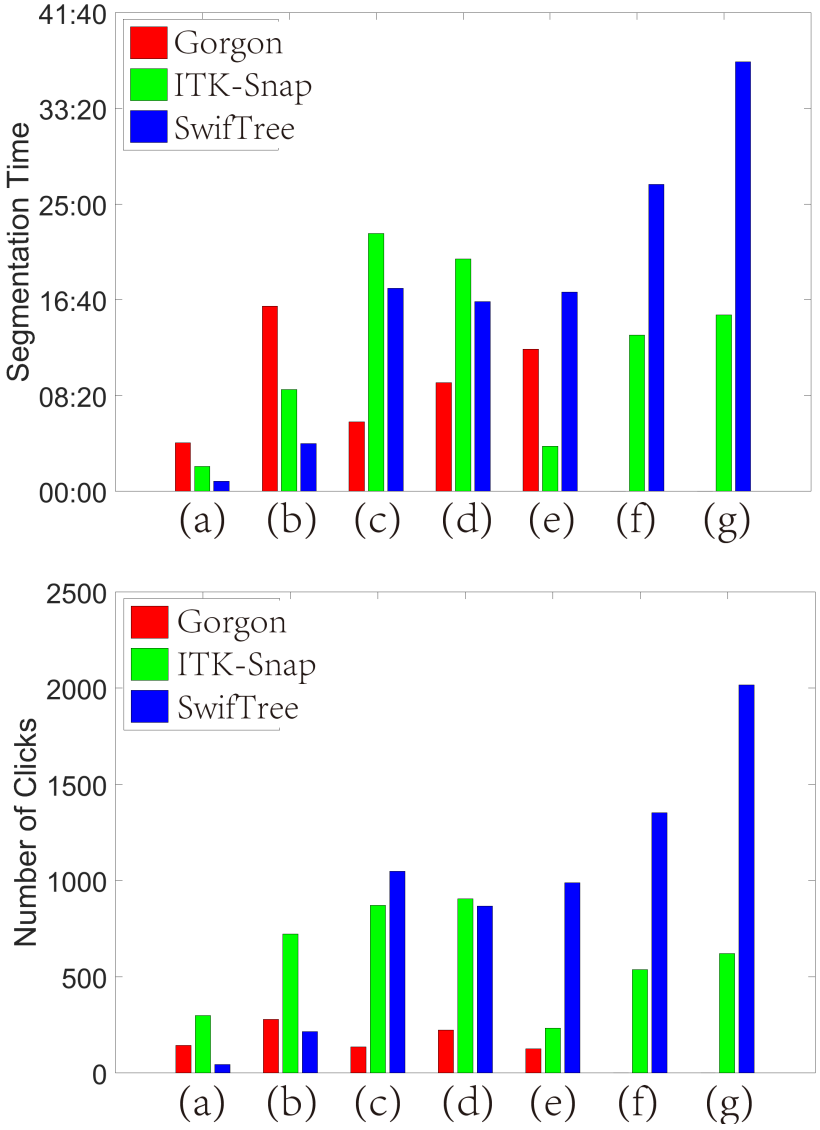

Figure 4.5: Tree extraction time consumption (above) and number of clicks (below) for different tools and different data (a) Y-Junc, (b) Helix, (c) Vascusynth, (d) Phantom, (e) Kidney, (f) Brain, and (g) Airway.

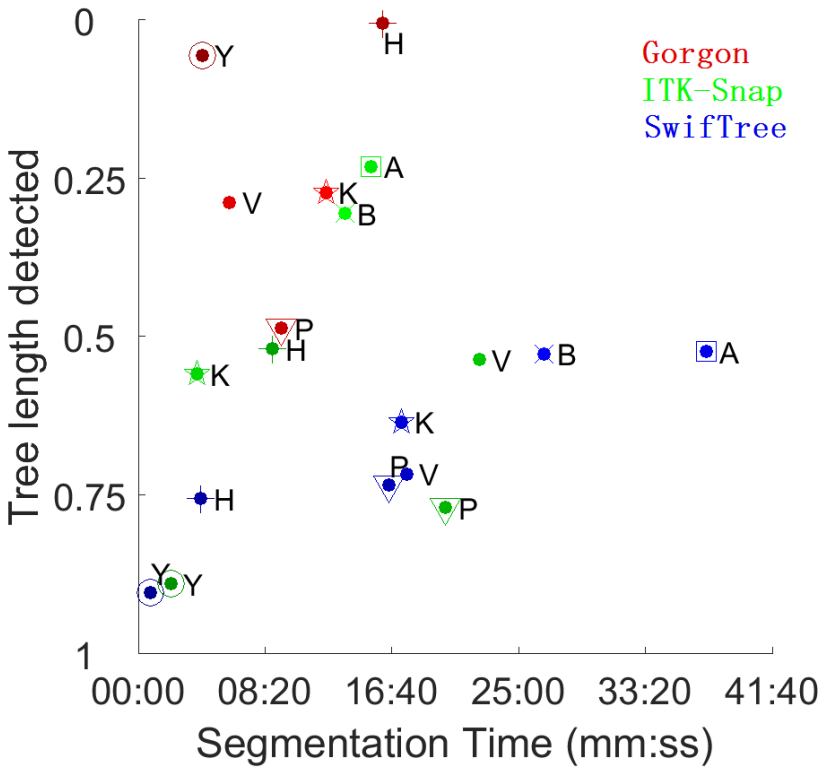

Figure 4.6: Tree length detected versus time consumption for tree extraction by ITK-Snap, Gorgon and SwifTree. Y◯: Y-Junc, H+: Helix, V•: Vascusynth, P▽: Phantom, K⋆: Kidney, B×: Brain and A□: Airway. Tree length detected increases downward, so BOTTOM LEFT is the desired location for the best method

them. The black dashed and green dotted line shows the tree length detected of the tree aggregated from all the 10 sessions for each image.

As can be seen in Figure 4.8, the tree aggregated from all participating sessions gives a more complete tree than any of the trees from the individual sessions. Also, the aggregated tree has the highest tree length detected with the highest initial slope (i.e., fastest increase). A small dip can be seen in the TLD of the aggregated tree due to false positive branches from some sessions.

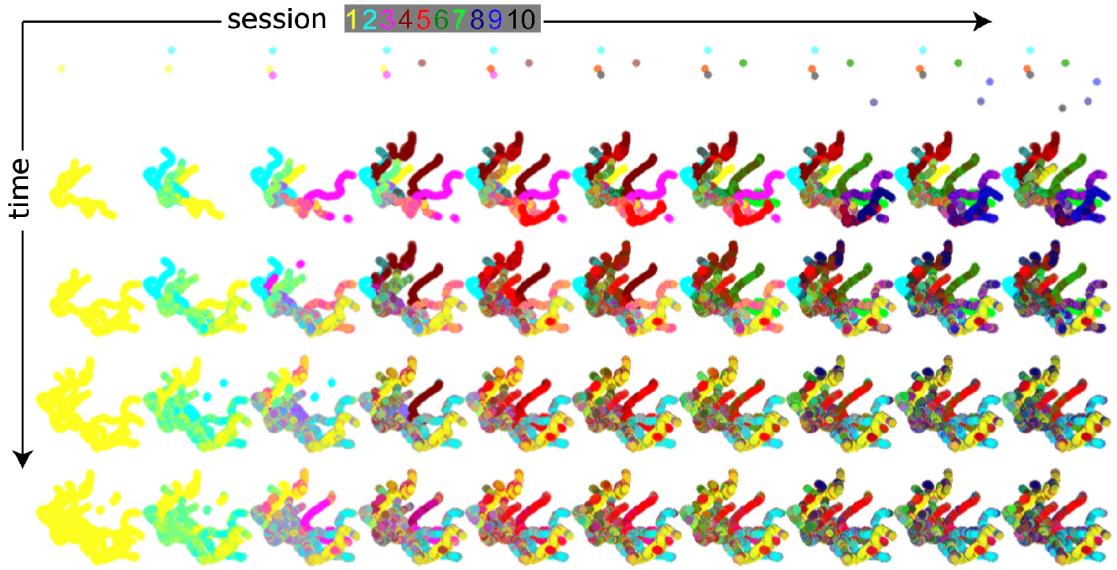

Figure 4.7: Benefits of crowdsourcing: The temporal progress of each of 10 sessions running SwifTree on the Brain dataset. As time advances and more sessions are included, the aggregated tree becomes more accurate and complete.

4.2.3 Benefit of gamification

In this experiment, we evaluated the performance for both the interactive non-game and game mode. In Figure 4.10, the line plot of tree length detected vs time consumption from the non-game and game mode for all the images is shown at the beginning of each row. On the right, the tree extracted from the non-game and game mode with time increasing from left to right. To investigate the influence of the gamification only, we additionally scaled up the time axis by the number of sessions to exclude crowdsourcing effect. In the figure, we select the total time spent by the interactive non-game mode as the last tick of time axis. So the the tree length detected curves for both non-game and game mode behave towards the end (rightmost side)

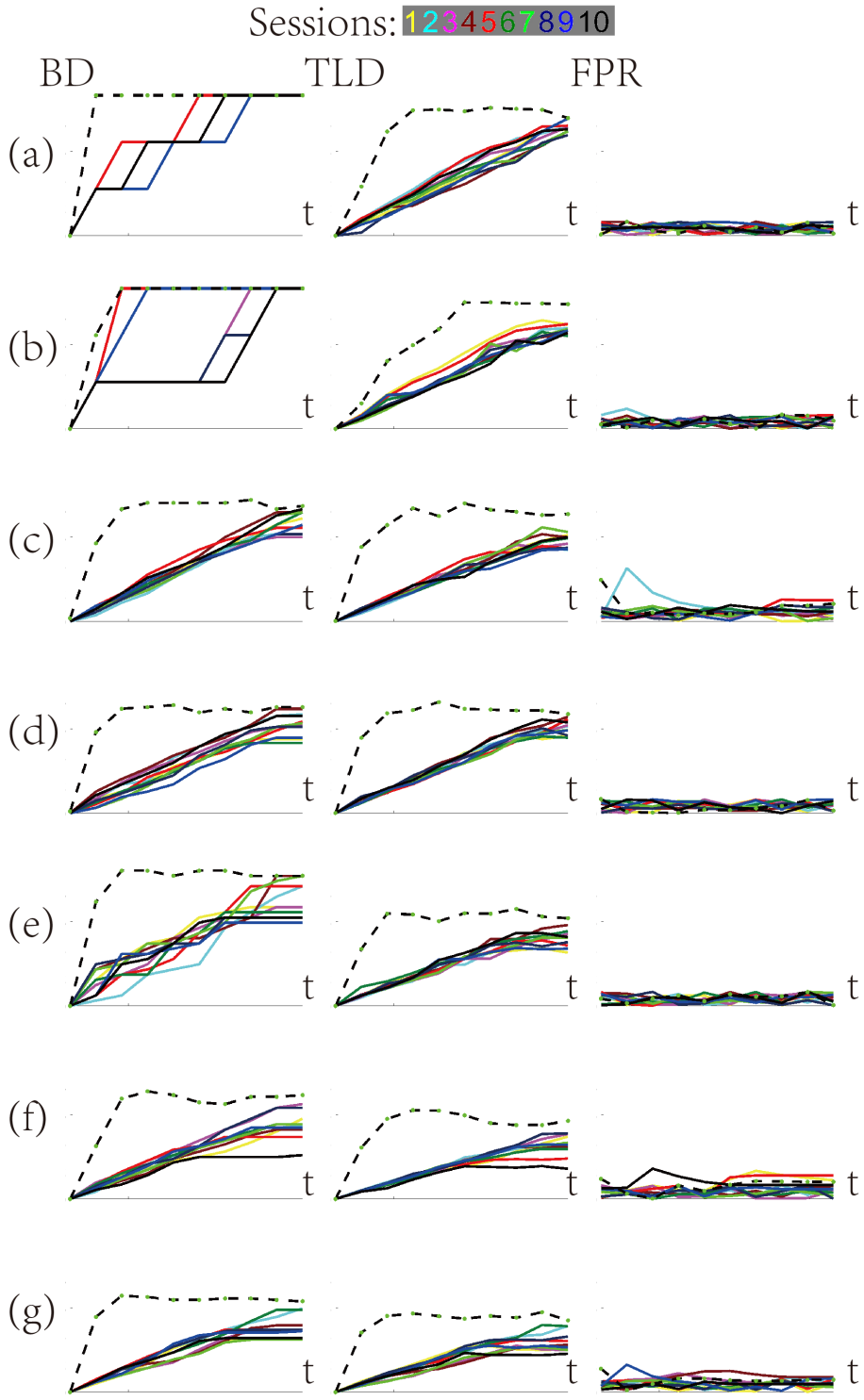

Figure 4.8: Benefits of crowdsourcing: Plots of BD, TLD and FPR vs time, for all data sets (a) Y-Junc (b) Helix (c) Vascusynth (d) Phantom (e) Kidney (f) Brain (g) Airway. Each solid colored curve corresponds to one tree extraction session. The black dashed curve, with better branch and tree detection (i.e., higher than other curves) and relatively lower false positive rate, corresponds to the aggregated tree from all 10 sessions.

It can be seen that enabling SwifTree game-mode features (i.e., velocity, sound effects, score, collectibles, and avoidables) reduces the time needed to reconstruct a pre-set tree compared to the non-game mode.

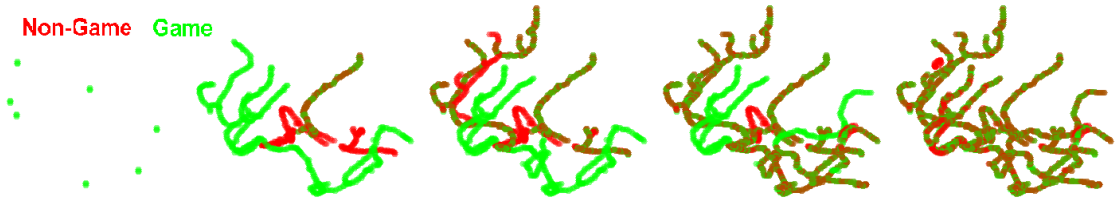

Figure 4.9: Benefit of gamification. Results on Brain. Progress of tree extraction shown at 5 instants. Game-mode sessions extract more branches quicker than non-game mode.

4.2.4 Robustness to noise

In this experiment, we compare automatic and interactive methods in terms of their robustness applied to noise. We polluted the images with additive white Gaussian noise with different standard deviations. We compared our SwifTree with two automatic methods: Frangi filter and Skeletonize3D plug-in of Imagej with respect to different noise levels.

As seen in Figure 4.11, the tree length detected, false positive rate and branches detected versus noise variance are presented on the left. The three orthographic views of image of different noise levels are shown on the right. In Figure 4.11, we see that Frangi filter and Skeletonize3D report high detection rates of branches and trees (left and middle columns). However, almost all these detections are false positives (right colume). SwifTree's false positive rate is much lower.

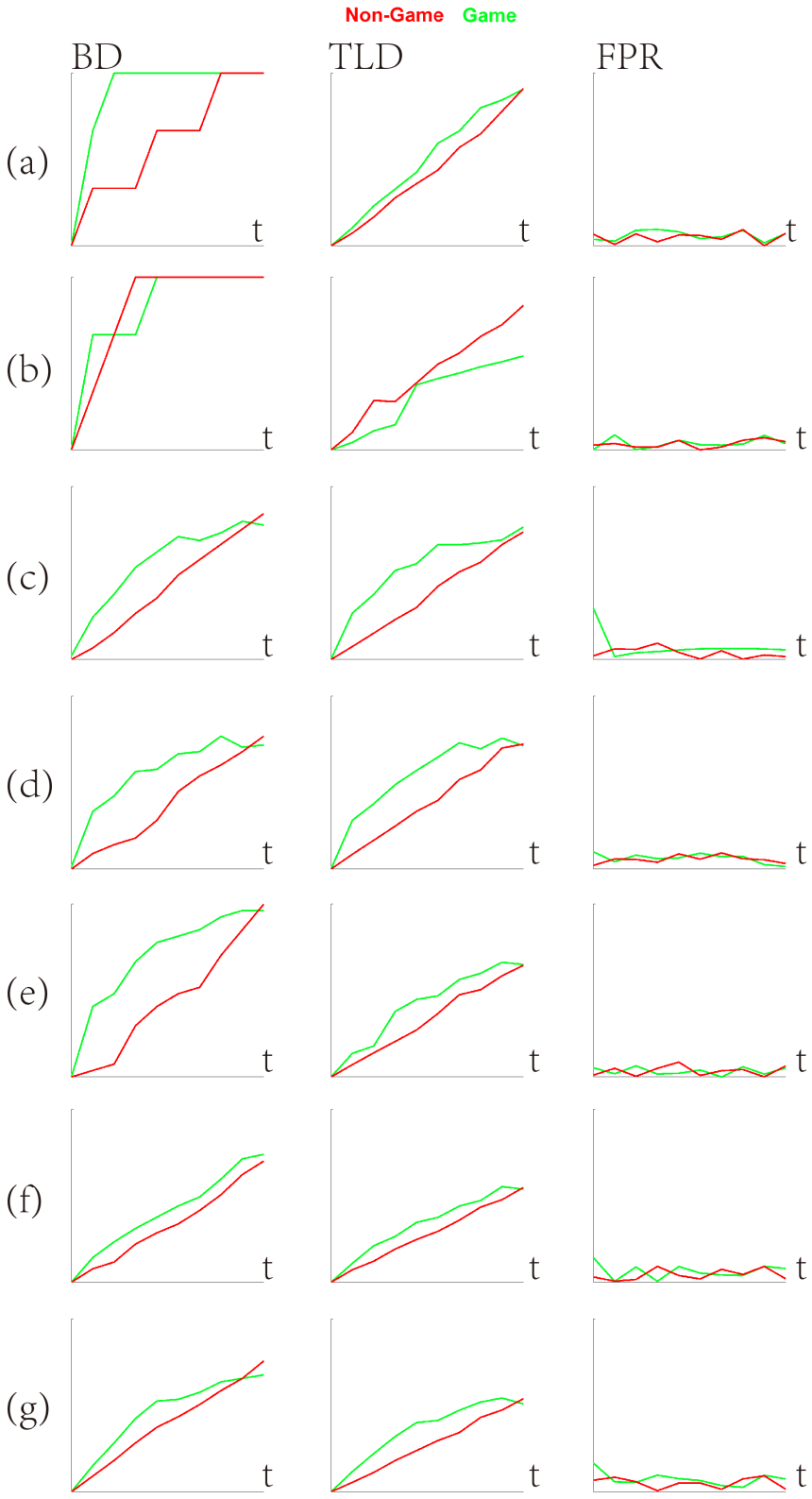

Figure 4.10: Benefit of gamification. Results on (a) Y-Junc (b) Helix (c) Vascusynth (d) Phantom (e) Kidney (f) Brain (g) Airway. TLD, BD and FPR vs time for game-mode (green) and interactive (non-game) mode (red). Game-mode sessions extract more branches quicker than non-game mode.

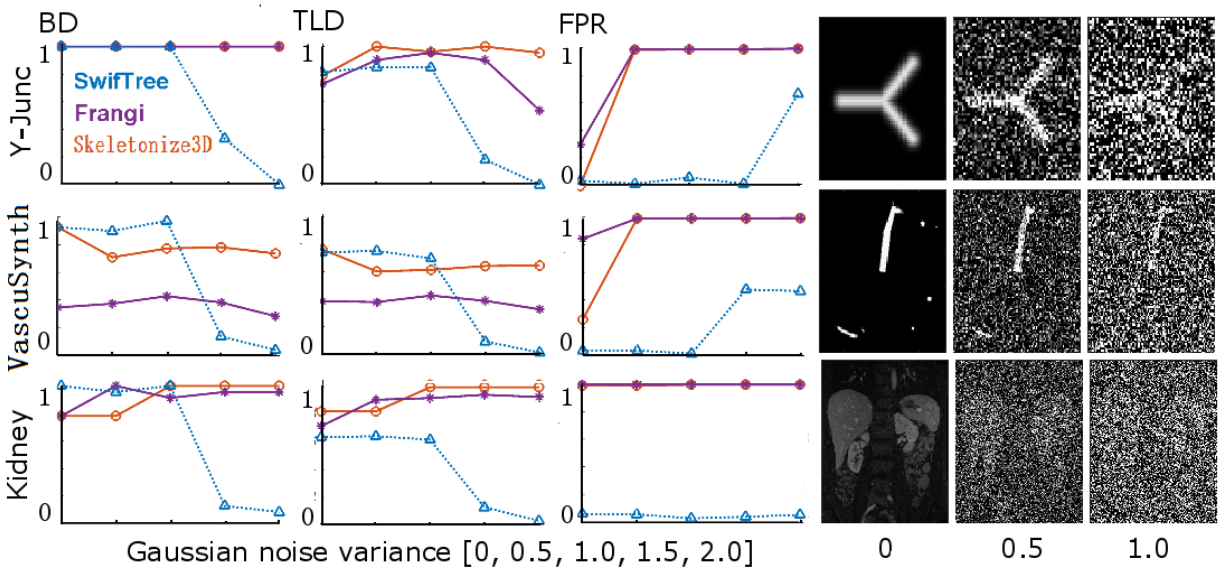

Figure 4.11: Robustness to noise. Left: Comparison of Frangi filter, Imagej Skeletonize3D and SwifTree in terms of robustness to noise. BD, TLD, and FPR are reported for the 3 methods across 3 datasets: Y-Junc (top), Vascusynth (middle) and Kidney (bottom). Right: Sample slices from each dataset at selected noise levels for illustration.

Chapter 5

Conclusion and Future work

We proposed SwifTree, a novel tool for extracting tree-like structures from 3D images. We showed that by exploring gamification and crowdsourcing, SwifTree can achieve more accurate results faster and is more robust to noise than traditional segmentation tools. We demonstrated that aggregating the results of multiple SwifTree crowdsourced sessions has the potential of achieving more accurate segmentation. Using the proposed game-mode can reduce time needed to achieve a pre-set tree segmentation accuracy. SwifTree outperforms automatic segmentation methods especially with respect to noise robustness.

- **Stronger validation:** Our initial experiments show that gamificaiton and crowdsourcing may have some value and worth exploring more. However, more experiments must be conducted and more comprehensive validation results should be obtained.
- **Alternative visualization:** In this work, we explored different types of glyphs and basic GUI that we believe were intuitive. Nevertheless, the GUI may be improved by exploring the use of other types of glyphs and visualization mechanisms. For example, one may consider a visualization that gives the user the ability to see 'ahead' rather than just the local slice.
- **Handling trifurcations:** In this work, we only support bifurcations, however, trifurcations may also appear in real data, albeit rarely. Trifurcation may be handled as two close-by bifurcations or the user may indicate a trifurcation, so the they can traverse all 3 child branches.
- **Handling loops:** Our tool currently supports have a loop but only in the 3D tree segmentation representation. The loop, however, is currently not reflected in the graph/topology. Future work may include the ability to work with loops.
- **Rigorous Aggregation:** It would be interesting to consider more rigorous statistical approach to combine trees. This will be particularly important when dealing with

results provided by users who try to earn more point by submitting quick but erroneous results.

- **Professional game design** Gamification can encourage the user to be more patient, i.e., willing to spend more time when performing mundane tasks. Therefore, future work would benefit from professional game design elements (e.g. virtual reality, scoring, and levels). For instance, the game can keep user/players' records to accumulate the reward and rank, multi-player competitive mode (more than one players start at different location in one image, the one with the highest score wins) to stimulate the enthusiasm.

- **Crowdsourcing platform** The next phase of our work involves releasing SwifTree publicly as a 'Human Intelligence Task' (HIT) on the established crowdsourcing platform Amazon Mechanical Turk, then analyzing the results collected from a large scale study involving hundreds of workers or 'Turkers'. Further, we will set up our own platform that allows the user who needs to segment tree from an image to submit the task to our system and recruit other users to complete it.

- **User studies:** Future work, should involve performing user studies to assess the advantage of different game and tool design choices. One key aspect that must be evaluated related to the trade-off between speed and accuracy. For example, crowdsourcing from multiple users can make the tree extraction progress faster and more efficient. However this increase in speed may result in decreased accuracy (e.g., more false positives as users may perform a task quickly to earn points). In the current work, we choose the speed of the cursor at most 2px per second to avoid accuracy reduction. We will later take advantage of the user study to find the best speed. The study will also examine which views and glyph types are most useful.

  .

Bibliography

[1] Sasakthi S Abeysinghe and Tao Ju. Interactive skeletonization of intensity volumes. *The Visual Computer*, 25(5-7):627–635, 2009.

[2] Shadi Albarqouni, Stefan Matl, Maximilian Baust, Nassir Navab, and Stefanie Demirci. Playsourcing: A novel concept for knowledge creation in biomedical research. In *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 269–277. Springer, 2016.

[3] Nina Amenta, Manasi Datar, Asger Dirksen, Marleen de Bruijne, Aasa Feragen, Xiaoyin Ge, Jesper Holst Pedersen, Marylesa Howard, Megan Owen, Jens Petersen, et al. Quantification and visualization of variation in anatomical trees. In *Research in Shape Modeling*, pages 57–79. Springer, 2015.

[4] AnaiS Badoual, Maxime Gerard, Benjamin De Leener, Nadine Abi-Jaoudeh, and Samuel Kadoury. 3d vascular path planning of chemo-embolizations using segmented hepatic arteries from mr angiography. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 1346–1349. IEEE, 2016.

[5] William A Barrett and Eric N Mortensen. Interactive live-wire boundary extraction. *Medical image analysis*, 1(4):331–341, 1997.

[6] Christian Bauer, Thomas Pock, Horst Bischof, and Reinhard Beichel. Airway tree reconstruction based on tube detection. In *Proceedings of the 2nd International Workshop on Pulmonary Image Analysis*, pages 203–214. CreateSpace, 2009.

[7] Christian Bauer and Homer Simpson. Segmentation of 3d tubular tree structures in medical images. 2010.

[8] Dirk Breitenreicher, Michal Sofka, Stefan Britzen, and Shaohua K Zhou. Hierarchical discriminative framework for detecting tubular structures in 3d images. In *International Conference on Information Processing in Medical Imaging*, pages 328–339. Springer, 2013.

[9] Suheyla Cetin, Ali Demir, Anthony Yezzi, Muzaffer Degertekin, and Gozde Unal. Vessel tractography using an intensity based tensor model with branch detection. *IEEE transactions on medical imaging*, 32(2):348–363, 2013.

[10] Alberto Chávez-Aragón, Won-Sook Lee, and Aseem Vyas. A crowdsourcing web platform-hip joint segmentation by non-expert contributors. In *Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium on*, pages 350–354. IEEE, 2013.

[11] Veronika Cheplygina, Adria Perez-Rovira, Wieying Kuo, Harm AWM Tiddens, and Marleen de Bruijne. Early experiences with crowdsourcing airway annotations in chest ct. In *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 209–218. Springer, 2016.

[12] O Chutatape, Liu Zheng, and SM Krishnan. Retinal blood vessel detection and tracking by matched gaussian and kalman filters. In *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE*, volume 6, pages 3144–3149. IEEE, 1998.

[13] Cassandra Coburn. Play to cure: Genes in space. *The Lancet Oncology*, 15(7):688, 2014.

[14] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[15] Maryam Taghizadeh Dehkordi, Saeed Sadri, and Alimohamad Doosthoseini. A review of coronary vessel segmentation algorithms. *Journal of medical signals and sensors*, 1(1):49, 2011.

[16] Thomas Deschamps and Laurent D Cohen. Fast extraction of minimal paths in 3d images and applications to virtual endoscopy. *Medical image analysis*, 5(4):281–299, 2001.

[17] Ryan Dickie, Ghassan Hamarneh, and Rafeef Abugharbieh. Live-vessel: Interactive vascular image segmentation with simultaneous extraction of optimal medial and boundary paths. Technical report, Citeseer, 2009.

[18] Stefan Diepenbrock and Timo Ropinski. From imprecise user input to precise vessel segmentations. In *Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM 2012), Norrköping, Sweden, September 27-28, 2012*, pages 65–72. The Eurographics Association, 2012.

[19] Axel Donath and Daniel Kondermann. Is crowdsourcing for optical flow ground truth generation feasible? In *International Conference on Computer Vision Systems*, pages 193–202. Springer, 2013.

[20] Evan Cyril Edmond, Samantha Xue-Li Sim, Hui-Hua Li, Eng-King Tan, and Ling-Ling Chan. Vascular tortuosity in relationship with hypertension and posterior fossa volume in hemifacial spasm. *BMC Neurology*, 16, 2016.

[21] Aasa Feragen, Pechin Lo, Marleen de Bruijne, Mads Nielsen, and François Lauze. Toward a theory of statistical tree-shape analysis. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):2008–2021, 2013.

[22] Marco Feuerstein, Takayuki Kitasaka, and Kensaku Mori. Adaptive branch tracing and image sharpening for airway tree extraction in 3-d chest ct. In *Proc. Second International Workshop on Pulmonary Image Analysis*, pages 273–284, 2009.

[23] Charles Florin, Nikos Paragios, and Jim Williams. Particle filters, a quasi-monte carlo solution for segmentation of coronaries. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 246–253. Springer, 2005.

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

[25] Michael W Graham, Jason D Gibbs, and William E Higgins. Robust system for human airway-tree segmentation. In *Medical Imaging*, pages 69141J–69141J. International Society for Optics and Photonics, 2008.

[26] Roman Grothausmann, Manuela Kellner, Marko Heidrich, Raoul-Amadeus Lorbeer, Tammo Ripken, Heiko Meyer, Mark P Kuehnel, Matthias Ochs, and Bodo Rosenhahn. Method for 3d airway topology extraction. *Computational and mathematical methods in medicine*, 2015, 2015.

[27] Ghassan Hamarneh, Johnson Yang, Chris McIntosh, and Morgan Langille. 3d live-wire-based semi-automatic segmentation of medical images. In *Medical imaging*, pages 1597–1603. International Society for Optics and Photonics, 2005.

[28] Pheng-Ann Heng, Hanqiu Sun, Kwong-Wai Chen, and Tien-Tsin Wong. Interactive navigation of virtual vessel tracking with 3d intelligent scissors. *International Journal of Image and Graphics*, 1(02):273–285, 2001.

[29] Christoph Hennersperger and Maximilian Baust. Play for me: Image segmentation via seamless playsourcing. *The Computer Games Journal*, pages 1–16.

[30] Xin Hu and Yuanzhi Cheng. Centerline-based vessel segmentation using graph cuts. In *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, pages 94432E–94432E. International Society for Optics and Photonics, 2015.

[31] Sung Hoon Jung, Soo Kweon Koo, Jang Won Choi, Ji Seung Moon, and Sang Hoon Lee. Upper airway structural changes induced by cpap in osas patients: a study using drug-induced sleep endoscopy. *European Archives of Oto-Rhino-Laryngology*, pages 1–6, 2016.

[32] Jeremy Kawahara, Chris McIntosh, Roger Tam, and Ghassan Hamarneh. Globally optimal spinal cord segmentation using a minimal path in high dimensions. In *2013 IEEE 10th International Symposium on Biomedical Imaging*, pages 848–851. IEEE, 2013.

[33] Michael Kerschnitzki, Philip Kollmannsberger, Manfred Burghammer, Georg N Duda, Richard Weinkamer, Wolfgang Wagermaier, and Peter Fratzl. Architecture of the osteocyte network correlates with bone material quality. *Journal of bone and mineral research*, 28(8):1837–1845, 2013.

[34] Cemil Kirbas and Francis Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys (CSUR)*, 36(2):81–121, 2004.

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

[36] Rahul P Kumar, Fritz Albregtsen, Martin Reimers, Thomas Langø, Bjørn Edwin, and Ole Jakob Elle. 3d multiscale vessel enhancement based centerline extraction of blood vessels. In *SPIE Medical Imaging*, pages 86691X–86691X. International Society for Optics and Photonics, 2013.

[37] Rahul Prasanna Kumar, Fritz Albregtsen, Martin Reimers, Bjørn Edwin, Thomas Langø, and Ole Jakob Elle. Three-dimensional blood vessel segmentation and centerline extraction based on two-dimensional cross-section analysis. *Annals of biomedical engineering*, 43(5):1223–1234, 2015.

[38] Delphine Lariviere, Khadija Benali, Baptiste Coustet, Nicoletta Pasi, Fabien Hyafil, Isabelle Klein, Maria Chauchard, Jean-François Alexandra, Tiphaine Goulenok, Antoine Dossier, et al. Positron emission tomography and computed tomography angiography for the diagnosis of giant cell arteritis: A real-life prospective study. *Medicine*, 95(30):e4146, 2016.

[39] David Lesage, Elsa D Angelini, Isabelle Bloch, and Gareth Funka-Lea. A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical image analysis*, 13(6):819–845, 2009.

[40] David Lesage, Guillaume Pizaine, Shiro Miyayama, Hicham Kobeiter, Bradford J Wood, Peter Mielekamp, William van der Sterren, and Alessandro Radaelli. Virtual parenchymal perfusion for selective intra-arterial therapy of liver cancer. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 1147–1150. IEEE, 2016.

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

[42] Pechin Lo, Jon Sporring, Haseem Ashraf, Jesper JH Pedersen, and Marleen de Bruijne. Vessel-guided airway tree segmentation: A voxel classification approach. *Medical image analysis*, 14(4):527–538, 2010.

[43] Pechin Lo, Jon Sporring, and Marleen de Bruijne. Multiscale vessel-guided airway tree segmentation. In *Proc. of Second International Workshop on Pulmonary Image Analysis*, pages 323–332, 2009.

[44] Pechin Lo, Jon Sporring, Jesper Johannes Holst Pedersen, and Marleen de Bruijne. Airway tree extraction with locally optimal paths. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 51–58. Springer Berlin Heidelberg, 2009.

[45] Pechin Lo, Bram Van Ginneken, Joseph M Reinhardt, Tarunashree Yavarna, Pim A De Jong, Benjamin Irving, Catalin Fetita, Margarete Ortner, Rômulo Pinho, Jan Sijbers, et al. Extraction of airways from ct (exact'09). *IEEE Transactions on Medical Imaging*, 31(11):2093–2107, 2012.

[46] Miguel Angel Luengo-Oroz, Asier Arranz, and John Frean. Crowdsourcing malaria parasite quantification: an online game for analyzing images of infected thick blood smears. *Journal of medical Internet research*, 14(6):e167, 2012.

[47] Lena Maier-Hein, Sven Mersmann, Daniel Kondermann, Sebastian Bodenstedt, Alexandro Sanchez, Christian Stock, Hannes Gotz Kenngott, Mathias Eisenmann, and Stefanie Speidel. Can masses of non-experts train highly accurate image classifiers? In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 438–445. Springer, 2014.

[48] Lena Maier-Hein, Sven Mersmann, Daniel Kondermann, Christian Stock, Hannes Gotz Kenngott, Alexandro Sanchez, Martin Wagner, Anas Preukschas, Anna-Laura Wekerle, Stefanie Helfert, et al. Crowdsourcing for reference correspondence generation in endoscopic images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 349–356. Springer, 2014.

[49] Marija Marčan, Bor Kos, and Damijan Miklavčič. Effect of blood vessel segmentation on the outcome of electroporation-based treatments of liver tumors. *PloS one*, 10(5):e0125591, 2015.

[50] Peter C Marks, Marilena Preda, Terry Henderson, Lucy Liaw, Volkhard Lindner, Robert E Friesel, and Ilka M Pinz. Interactive 3d analysis of blood vessel trees and collateral vessel volumes in magnetic resonance angiograms in the mouse ischemic hindlimb model. *The open medical imaging journal*, 7:19, 2013.

[51] Chris McIntosh and Ghassan Hamarneh. Vessel crawlers: 3d physically-based deformable organisms for vasculature segmentation and analysis. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 1084–1091. IEEE, 2006.

[52] Sílvia Delgado Olabarriaga and Arnold WM Smeulders. Interaction in the segmentation of medical images: A survey. *Medical image analysis*, 5(2):127–142, 2001.

[53] Jens Petersen, Mads Nielsen, Pechin Lo, Zaigham Saghir, Asger Dirksen, and Marleen De Bruijne. Optimal graph based segmentation using flow lines with application to airway wall segmentation. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 49–60. Springer, 2011.

[54] Kelvin Poon, Ghassan Hamarneh, and Rafeef Abugharbieh. Live-vessel: Extending livewire for simultaneous extraction of optimal medial and boundary paths in vascular images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 444–451. Springer, 2007.

[55] Jiantao Pu, Carl Fuhrman, Walter F Good, Frank C Sciurba, and David Gur. A differential geometric approach to automated segmentation of human airway tree. *IEEE transactions on medical imaging*, 30(2):266–278, 2011.

[56] Jiantao Pu, Suicheng Gu, Shusen Liu, Shaocheng Zhu, David Wilson, Jill M Siegfried, and David Gur. Ct based computerized identification and analysis of human airways: a review. *Medical physics*, 39(5):2603–2616, 2012.

[57] M Russ, R O'Hara, SV Setlur Nagesh, M Mokin, C Jimenez, A Siddiqui, D Bednarek, S Rudin, and C Ionita. Treatment planning for image-guided neuro-vascular interventions using patient-specific 3d printed phantoms. In *SPIE Medical Imaging*, pages 941726–941726. International Society for Optics and Photonics, 2015.

[58] Sethuraman Sankaran, Leo Grady, and Charles A Taylor. Fast computation of hemodynamic sensitivity to lumen segmentation uncertainty. *IEEE transactions on medical imaging*, 34(12):2562–2571, 2015.

[59] Yanfeng Shang, Rudi Deklerck, Edgard Nyssen, Aneta Markova, Johan de Mey, Xin Yang, and Kun Sun. Vascular active contour for vessel tree segmentation. *IEEE Transactions on Biomedical Engineering*, 58(4):1023–1032, 2011.

[60] Lilong Shi, Brian Funt, and Ghassan Hamarneh. Quaternion color curvature. In *Color and Imaging Conference*, volume 2008, pages 338–341. Society for Imaging Science and Technology, 2008.

[61] Sahar Soleimanifard, Michael Schär, Allison G Hays, Robert G Weiss, Matthias Stuber, and Jerry L Prince. Vessel centerline tracking and boundary segmentation in coronary mra with minimal manual interaction. In *2012 9th IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 1417–1420. IEEE, 2012.

[62] Christoph Sommer, Christoph Straehle, Ullrich Köthe, and Fred A Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *2011 IEEE international symposium on biomedical imaging: From nano to macro*, pages 230–233. IEEE, 2011.

[63] Julio Sotelo, Jesus Urbina, Israel Valverde, Cristian Tejos, Pablo Irarrázaval, Marcelo E Andia, Sergio Uribe, and Daniel E Hurtado. 3d quantification of wall shear stress and oscillatory shear index using a finite-element method in 3d cine pc-mri data of the thoracic aorta. *IEEE transactions on medical imaging*, 35(6):1475–1487, 2016.

[64] Jasjit S Suri, Kecheng Liu, Laura Reden, and Swamy Laxminarayan. A review on mr vascular image processing: skeleton versus nonskeleton approaches: part ii. *IEEE transactions on information technology in biomedicine: a publication of the IEEE Engineering in Medicine and Biology Society*, 6(4):338–350, 2002.

[65] Mary B Vickerman, Patricia A Keith, Terri L McKay, Dan J Gedeon, Michiko Watanabe, Monica Montano, Ganga Karunamuni, Peter K Kaiser, Jonathan E Sears, Quteba Ebrahem, et al. Vesgen 2d: Automated, user-interactive software for quantification and mapping of angiogenic and lymphangiogenic trees and networks. *The anatomical record*, 292(3):320–332, 2009.

[66] X Wang, T Heimann, P Lo, M Sumkauskaite, M Puderbach, Marleen de Bruijne, HP Meinzer, and I Wegner. Statistical tracking of tree-like tubular structures with efficient branching detection in 3d medical image data. *Physics in medicine and biology*, 57(16):5325, 2012.

[67] Yu Wang, Arunachalam Narayanaswamy, and Badrinath Roysam. Novel 4-d open-curve active contour and curve completion approach for automated tree structure extraction. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1105–1112. IEEE, 2011.

[68] Kun-Chang Yu, Erik L Ritman, and William E Higgins. Graphical tools for improved definition of 3d arterial trees. In *Medical Imaging 2004*, pages 485–495. International Society for Optics and Photonics, 2004.

[69] Yong Zhang, Kun Chen, and S Wong. 3d interactive centerline extraction. *The Insight Journal*, 2008.

[70] Mengliu Zhao and Ghassan Hamarneh. Bifurcation detection in 3d vascular images using novel features and random forest. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 421–424. IEEE, 2014.

[71] Yefeng Zheng, Maciej Loziczonek, Bogdan Georgescu, S Kevin Zhou, Fernando Vega-Higuera, and Dorin Comaniciu. Machine learning based vesselness measurement for coronary artery segmentation in cardiac ct volumes. In *Proc. SPIE Med. Imaging*, volume 7962, pages 79621K–1, 2011.

[72] Jinghao Zhou, Sukmoon Chang, Dimitris Metaxas, and Leon Axel. Vascular structure segmentation and bifurcation detection. In *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 872–875. IEEE, 2007.

[73] Wen-Bo Zhu, Bin Li, Lian-Fang Tian, Xiang-Xia Li, and Qing-Lin Chen. Topology adaptive vessel network skeleton extraction with novel medialness measuring function. *Computers in biology and medicine*, 64:40–61, 2015.

Appendix A

User Manual

SwifTree is an interactive software designed to track and identify, quickly and accurately, all branches in an anatomical tree (e.g., vascular or airway trees) from a 3D volumetric image. SwifTree requires MATLAB and the following Toolboxes: Neural Network Toolbox, Image Processing Toolbox, and Statistics and Machine Learning Toolbox. SwifTree has been tested on: MATLAB 2015b on Windows 7 64bit; MATLAB 2016b on Mac OS X. The following figures give an overview of SwifTree.

(1) Software Setup

   (a) Download and unzip SwifTree.zip from */cs/ghassan/students_less/mianh/*

   (b) Launch MATLAB

   (c) Browse to folder */SwifTree/code* using the 'cd' or the MATLAB Folder GUI.

   (d) Run SwifTree: type SwifTree in the command prompt and press return.

(2) Load 3D Image Data
You will be asked to choose an image format (e.g., MAT, DICOM, MetaImage). There are several example MAT files provided in the Data folder. You can find these data by pressing the button below in Fig [A.2]
For this tutorial, choose the 3D Phantom under *Data/phatom.mat*, and press 'Open' in Fig [A.3]

(3) Pre-processing
After some delay for loading and processing the image (around 1 min, depending on your machine specifications and image size), SwifTree is ready to be used to analyze the image in Fig [A.4]

(4) SwifTree Windows
You should see an image like the one below. Your task is to segment all the branches of the vascular tree in the loaded Phantom image by steering your avatar (green ball in the middle) into the bright region (i.e., the cross-section of the branch). When you are ready, press return to start the branch tracking process. The following figure highlights the main components of the SwifTree graphical interface in Fig [A.5]

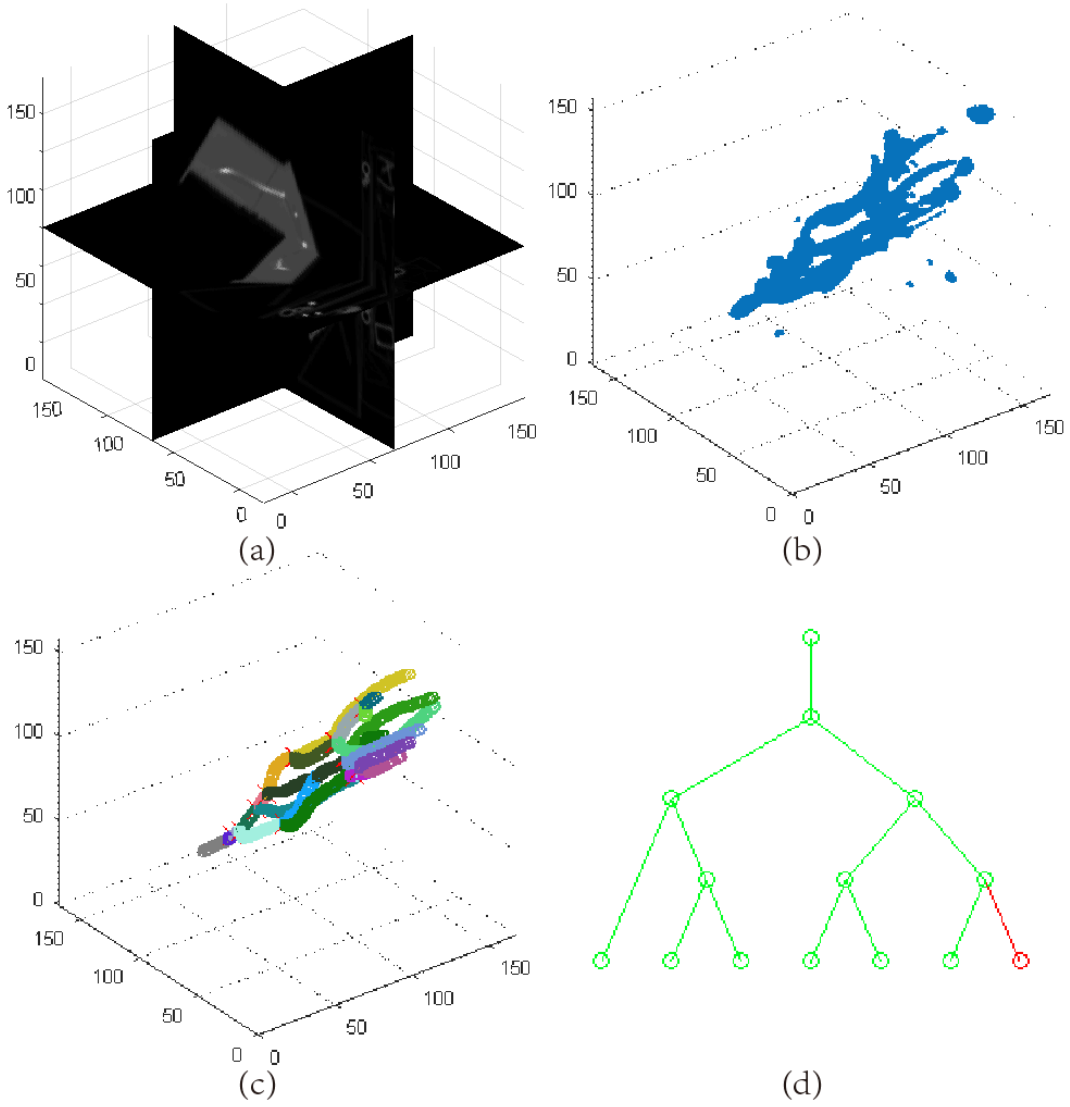

Figure A.1: Overview of SwifTree: (a) Input image is shown by three orthogonal slices. (b) Simple thresholding to show approximately where the anatomical tree is. (c) The tracking and identification of different branches using SwifTree. (d) The tree hierarchy produced using SwifTree.

Figure A.2: Choose an image of a variety of format

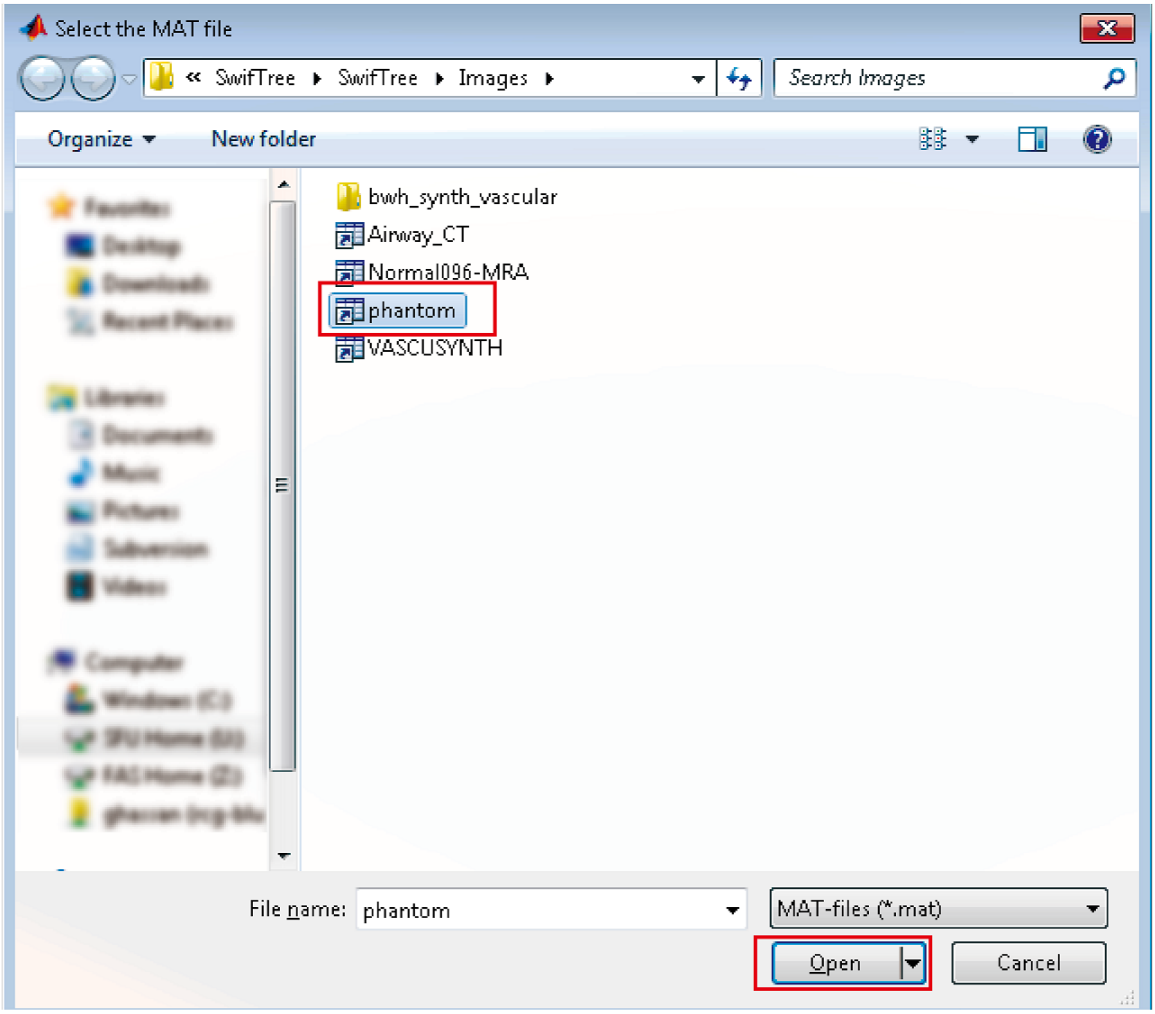

Figure A.3: Choose the 3D phantom as an example

```
>> SwifTree
Initializing...Please wait
Loading dataset phantom.mat
Initialization complete
```

Figure A.4: Pre-processing the image

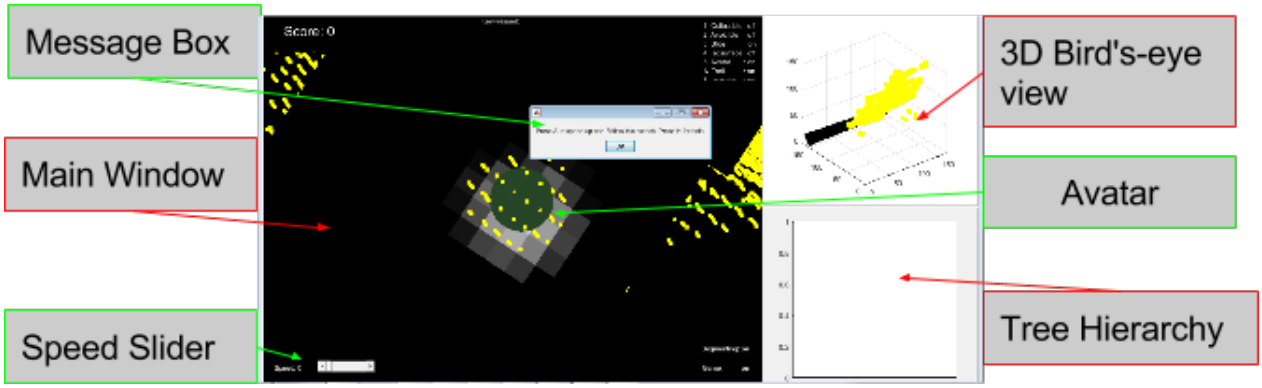

Figure A.5: The viewing windows of SwifTree

(5) Controls

You have two important controls: speed and direction. Speed is controlled by the keys 'A' to accelerate and 'B' to brake. You can also use the speed Slider. Direction is controlled by pressing the arrow keys on the keyboard: . You can also pan by pressing keyboard keys: i, k, j, and l, for up, down, left, and right, respectively. Other helpful controls are shown in the figure below and will be detailed later in Fig [A.6]

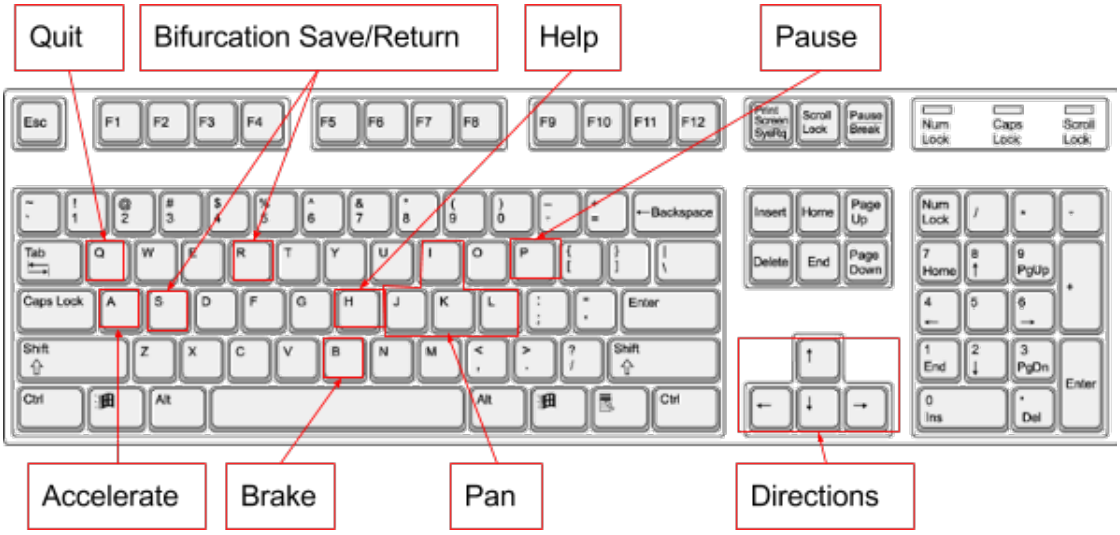

Figure A.6: Keyboard controllers

(6) Main Window Information

As you track, most of your attention should be focused on the Main Window, which will show the branch cross-section as well as other helpful information, as shown in the figure below. The Display channels show which visual objects are displayed, which will be explained later in Fig [A.7]

(7) Tracking

Now accelerate and steer through the first branch. To have a good segmentation result, you need to stay in the middle of the branch, keeping the avatar in the bright region,

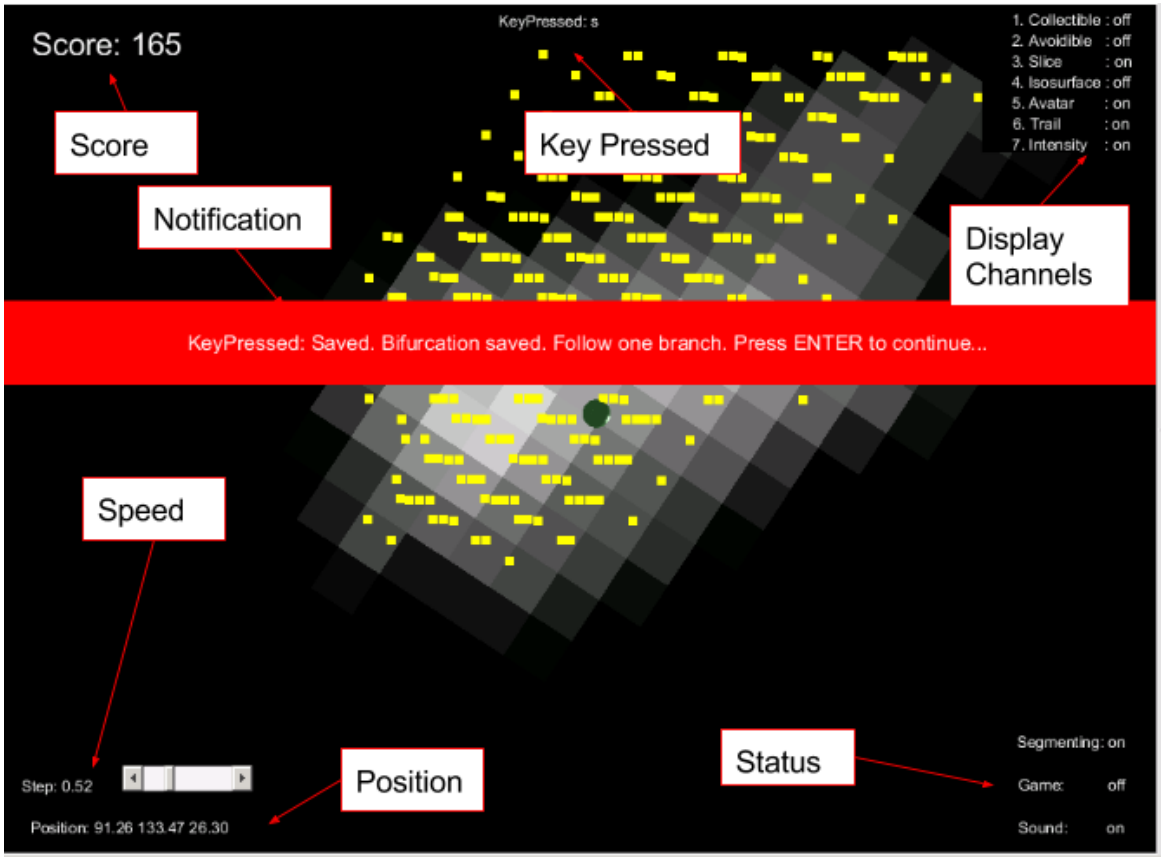

Figure A.7: Main window

and collecting the yellow dots. As long as you are doing so, you will hear beeps and your score will increase in Fig [A.8]

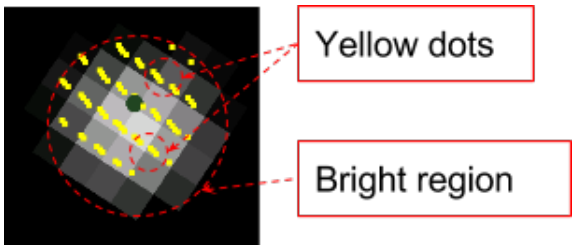

Figure A.8: Tracking the tree branch

(8) Bifurcation Identification
If a branching or bifurcation exists, you will encounter it as you travel down a branch and the bright cross-section will begin to split into two, like this in Fig [A.9] Once you

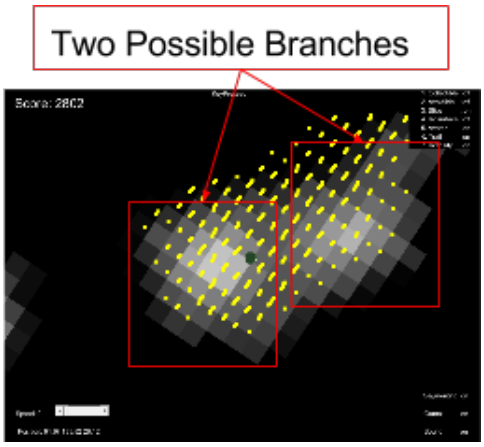

Figure A.9: Bifurcation identification

have identified the appearance of a bifurcation, press button 'S' so you can Save this branch location and visit it again later, which is important so you can travel down all branches. Note how the tree hierarchy changes once you identify a bifurcation in Fig [A.10]

(9) Tracking Branches
After you identified the bifurcation, start travelling down one of the branches (e.g., the one to the left). When you reach the end of one branch, press 'R' to Return to the last saved bifurcation location. Green rings filled with black show the trail recorded when exploring the left branch. So now you need to travel down the other branch on the right. Note how the tree hierarchy shows a split now in Fig [A.11]

(10) Need help?
You can always press 'H' for Help at any time. This will show you the keyboard controls and other commands in Fig [A.12]

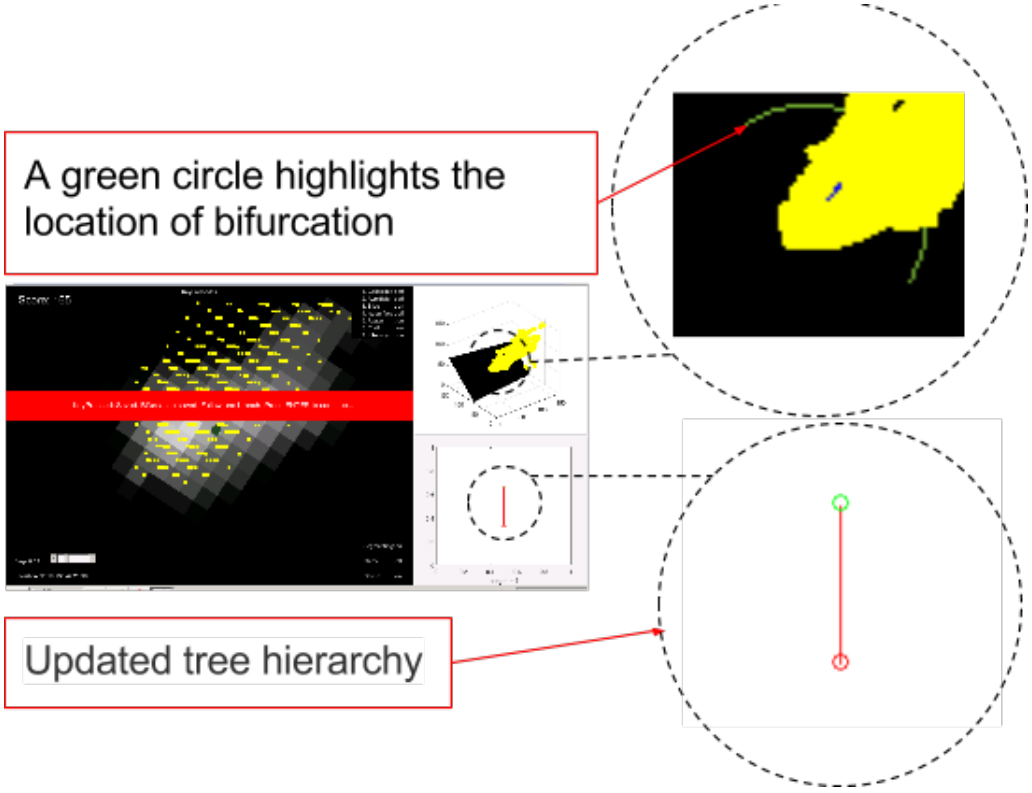

Figure A.10: Save the bifurcation

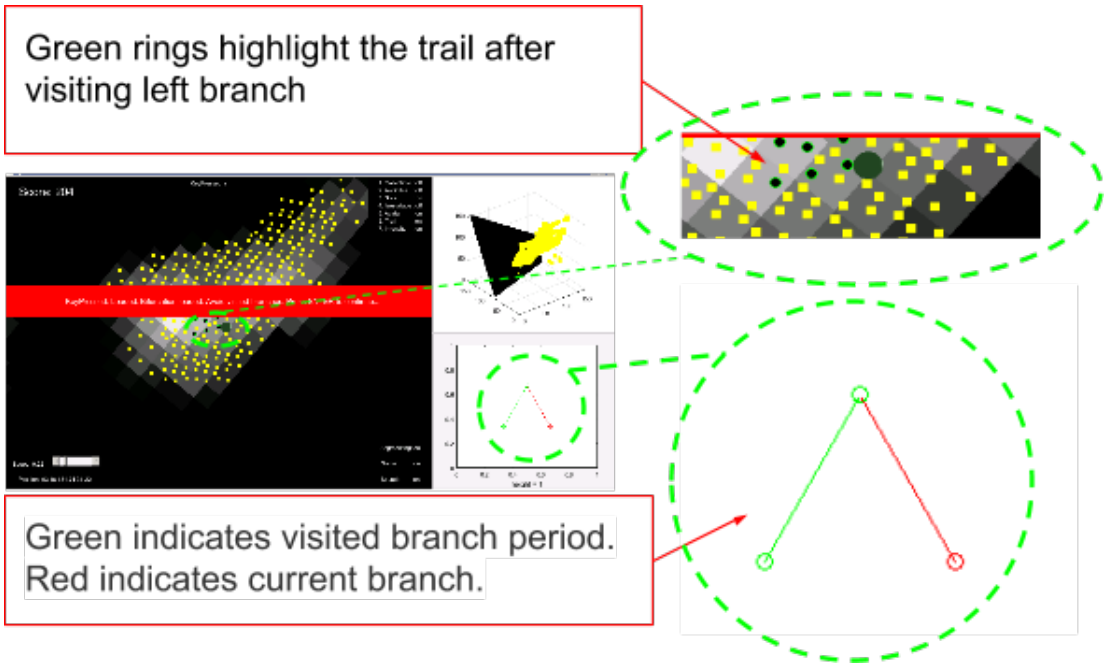

Figure A.11: Load the bifurcation

MOVEMENT

Pan
- I: up
- K: down
- J: left
- L: right

Turn
- \uparrow: up
- \downarrow: down
- \leftarrow: left
- \rightarrow: right

A: accelarate
B: brake
F: forward
V: reverse

DISPLAY:

1: Collectibles
2: Avoidables
3: Slice
4: Isosurface
5: Avatar
6: Trail

OTHER

S: save position
R: load position
T: segment
M: (m)ute on/off
G: (g)ame on/off
Z: zoom in
X: zoom out
H: help
P: pause
Q: quit

Figure A.12: Help

(11) Save and Quit
When you are done, press 'Q' to save the sessions and results then exit. You can always load your session at this point and continue later in Fig [A.13]

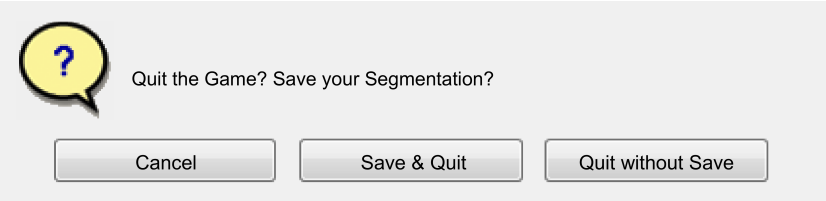

Figure A.13: Save and Quit

(12) Show results
The tracking result and other summary will be shown automatically after 'Save and Quit':

(a) Tracked branches in 3D (and the path to where the results are saved)

(b) Tree topology

(c) Time spent

(d) Keystrokes pressed.

You can also see all these results by executing showSegResult in the MATLAB command prompt in Fig [A.14][A.15]

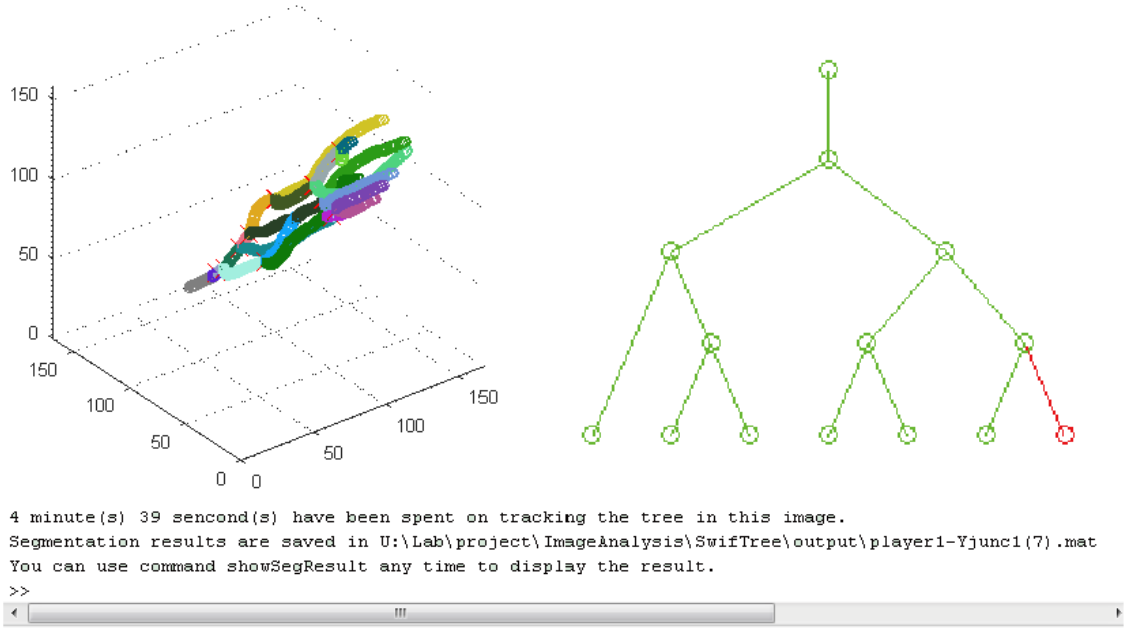

Figure A.14: Show segmentation result

(13) Continue segmentation from a previous session
If the image data selected is detected to have been segmented previously, you will be

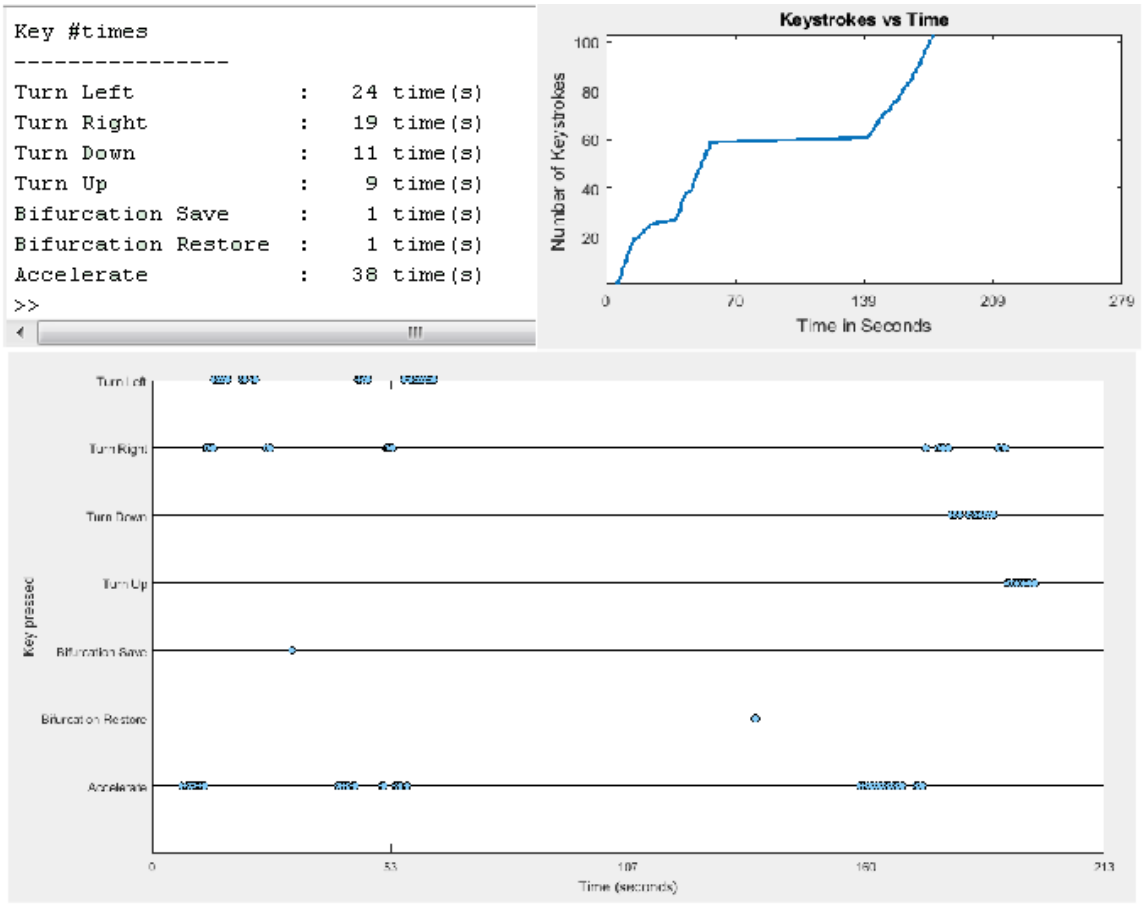

Figure A.15: Summary of the time and effort spent by the user

given the option to either continue the segmentation from the last saved point of a previous session or start from scratch.

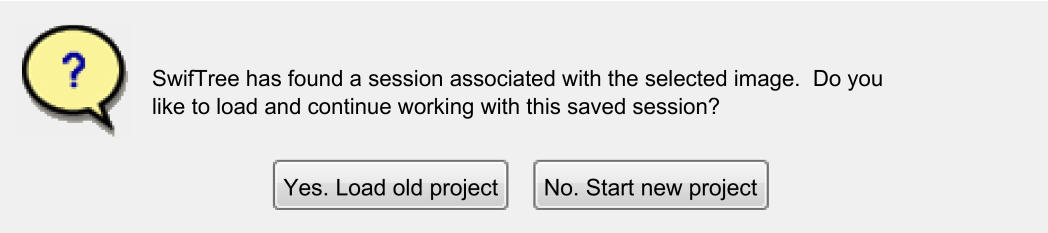

Figure A.16: Continue from a previous session

Different mode selection A.17 and data type selection A.18

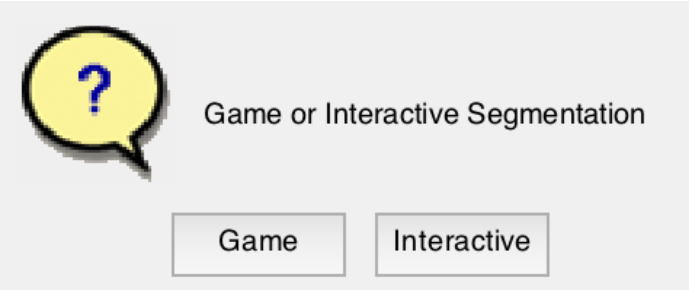

Figure A.17: Interactive and game mode

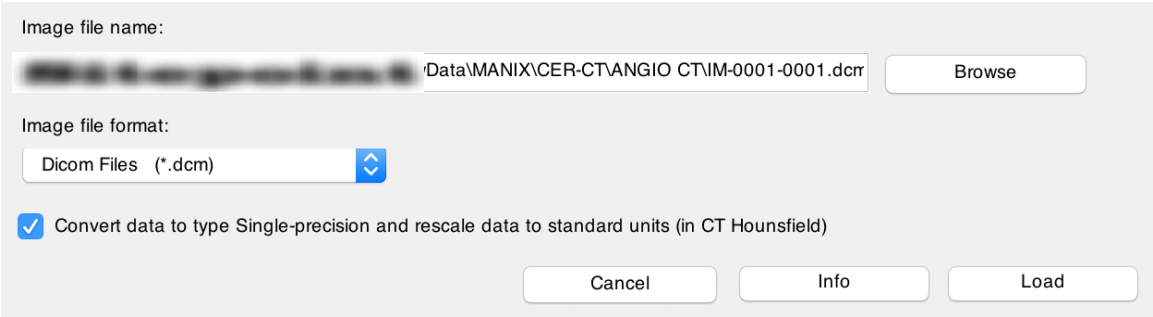

Figure A.18: Multiple data format selection