Vessel tracking by connecting the dots

Release 0.00

Andrzej Szymczak

August 8, 2008

Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401, USA,
aszymcza@mines.edu

Abstract

We propose an algorithm for tracking blood vessel segments in Computed Tomographic (CT) images. Our procedure first finds *core points* that tend to concentrate along the centerlines of vessels. Intuitively, the core points are centers of intensity plateaus in two-dimensional slices through the input image. The starting and the end point of the desired vessel (S and E) are also considered core points. The weighted *core graph* is built by connecting nearby core points with edges. Edge weights are designed so that edges of large weights are unlikely to follow a vessel segment. We compute the shortest path connecting S and E in the core graph. The output is the result of applying shortcutting operations to this path.

Latest version available at the Insight Journal [http://hdl.handle.net/1926/1338]

Contents

1 Introduction

According to the US Center for Disease Control and Prevention data, heart disease has recently been among the leading causes of death in the USA [6]. Automatic tracking of vessels in heart CT (Computed Tomography) scans is an important step toward early detection of plaques, aneurysms, stenoses and abnormal configurations of coronary arteries which could potentially lead to heart failure. Even though visual inspection of 2D slices is currently the most common way of making diagnosis based on cardiac CT scans, reliable methods for tracking vessels in 3D imagery could eventually make the process less labor-intensive by helping radiologists to correlate the information from different slices. In this paper, we propose an algorithm for tracking vessels in CT scans based on the ideas previously explored in [7, 8] for the purpose of reconstruction of coronary trees and airway trees from 3D images.

Blood vessel extraction algorithms have been receiving significant amount of attention in recent years (overviews of the subject can be found in [1, 3, 4]). A detailed discussion of prior work would increase the length of this paper well beyond the page limit. The procedure described in this paper is an example of a *ridge-based algorithm*: it generates points that tend to be densely distributed along intensity ridges and uses this set of points as a basis for vessel reconstruction. Intensity ridges naturally define an approximation of the vessel skeleton or centerline.

2 Algorithm

The input to our algorithm is a 3D greyscale image (cardiac CT scan) and two points, the starting point S and the end point E of the vessel to be tracked. The output is a path connecting S and E which, in most cases, closely tracks the blood vessel between the two points. As most computer vision algorithms, our procedure comes with no theoretical quality guarantee. Measurements of the quality of the output for several test datasets provided by the workshop organizers are given in Section 3. There are numerous parameters and constants used throughout the algorithm. All of them were tuned experimentally based on the properties of the training data, our intuition and past experience. Detailed analysis of the dependence of the results on these parameters is beyond the scope of this paper. However, our experiments indicate that the output is quite stable with respect to the values of these parameters.

The algorithm proceeds in several major steps described below. First, we apply smoothing and remove most vessels outside the heart from the input image (Section 2.1). We generate a set of *core points* that tend to concentrate near the centerlines of vessels (Section 2.2). A simple filtering procedure is applied to remove some of the outliers from the core point set (Section 2.3). We build a weighted *core graph* with edge weights that tend to penalize edges that do not follow a vessel (Section 2.4). Finally, the shortest path in the core graph, connecting the starting and end point of the vessel is found and smoothed by applying simple shortcutting operations (Section 2.5). The resulting path is output by the algorithm.

2.1 Preprocessing

Preprocessing has three major goals: normalization of the voxel values, cleaning up (i.e. removing the vessels and other structures from) the lung area and mild smoothing of the input image. Normalization allows us to reduce common artifacts of CT scans such as variation of contrast between neighboring slices that we encountered in our previous work. Cleanup of the lung area reduces the size of the core point set (Section 2.2). Smoothing removes noise from the image.

First, we compute the mean intensity μ of voxels of intensity greater than 500 for each axial slice. 500 is meant to be a conservative lower bound on the intensity of a voxel inside the heart. Then, we go over all voxels of the slice and, if the intensity of the voxel is I, we change it to $\max(0,(I-500)/(\mu-500))$. Note that, in particular, this maps all voxels whose original intensity is less than 500 to zero intensity. Let us call

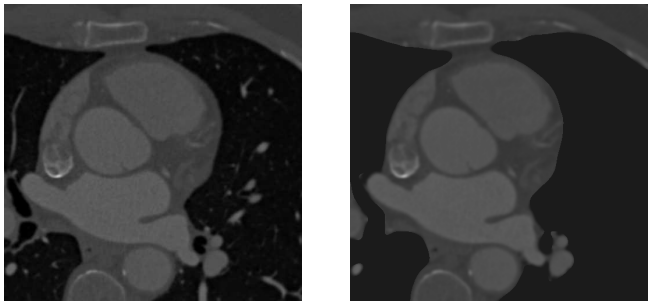

Figure 1: Left: a slice through one of the datasets. Right: the same slice after the preprocessing stage; notice that most of the area inside the lungs is black (zero intensity). The intensity inside the heart appears smoother as a result of using the Gaussian filter.

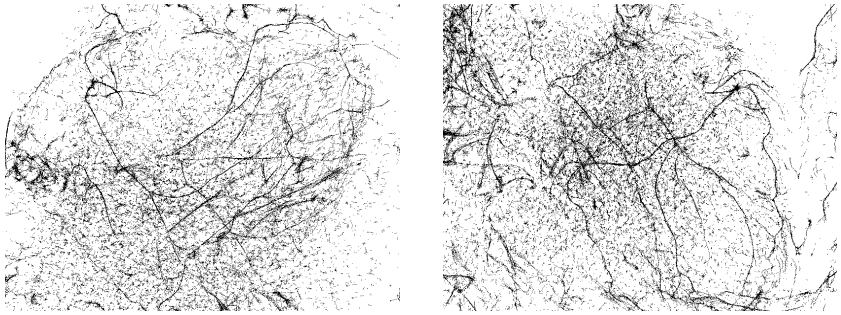

Figure 2: Core point sets used by our algorithm for a few of the input datasets. Notice the streaks of points running along the centerlines of the vessels.

the resulting image $\mathcal{V}_1$.

We apply the Gaussian filter with width $\sqrt{2}$ to $\mathcal{V}_1$ to obtain the image $\mathcal{V}_2$ and the Gaussian filter with width $6\sqrt{2}$ to obtain the image $\mathcal{V}_3$. The image used in the subsequent stages of the algorithm is obtained by setting the intensity of every voxel of $\mathcal{V}_2$ such that the intensity of the corresponding voxel in $\mathcal{V}_3$ is less than 0.5 to zero. In particular, this maps most of the voxels corresponding to the lung area to zero intensity (Figure 1).

2.2 Core points

The key step of our algorithm is computation of the set of *core points*. Core points form dense streaks near the centerlines of vessels. Needless to say, our algorithm is not perfect and the core point set contains numerous outliers (Figure 2). Note that this paper uses the approach of [8] rather than that of [7] to generate the core points. Visual inspection revealed that this leads to higher quality core point sets.

The core point set consists of 3D points generated based on analysis of two-dimensional slices of the input three-dimensional CT scan. A slice is a 2D grey-scale image. For each slice under consideration, we examine the evolution of connected components of the set of pixels obtained by thresholding as the threshold decreases from the maximum to the minimum pixel intensity. Thresholding the image yields union of pixels whose intensities are greater or equal to the threshold. Throughout this paper, we consider pixels to be closed rectangles.

A core point is the center of mass of a topologically simple component resulting from thresholding at the moment of slowest expansion due to threshold decrease (in other words, when the component hits a steep

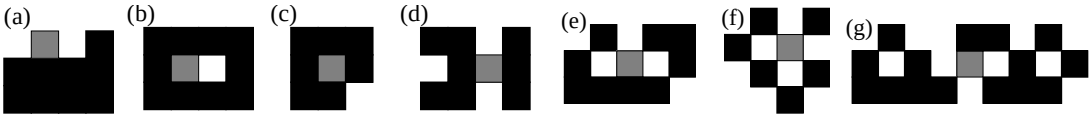

Figure 3: Adding a new pixel (shown in grey) to S (connected components of S intersecting the new pixel are shown black). In cases (a) and (b), there is no topology change (thus, no topological event takes place). In all other cases we have a topological event; (c) - a hole in a component disappears; (d) - two components with no holes merge into one with no holes; (e) and (f) - components with no holes become component with holes; (g) - components with holes merge.

wall). In addition to its coordinates, for every core point we record an uncertainty measure (an estimate of the minimum expansion speed for its component). The details are given below.

Components resulting from thresholding

To analyze the connected components of the sets resulting from thresholding, we start from empty set of pixels and insert pixels one by one in order of decreasing intensity. By S we shall denote the union of pixels inserted so far. 8-connectivity is used to determine connected components of S and their properties and 4-connectivity is used when dealing with the complement of S. As a result of adding a new pixel to S, the topological structure of connected components of S may change: new components may appear, some components may merge and some may change the topology (i.e. holes in the components can appear or disappear; by holes we mean bounded connected components of the complement). We shall call these structural changes *topological events*. Examples of topological events as well as voxels which do not induce a topological event when inserted into S are shown in Figure 3.

We keep track of the connected components of S using the disjoint-set datastructure [2, Chapter 21]. For each pixel p that is inserted, we look up the connected components of S that intersect p and merge them and p into a single component. Throughout the process, for each component F of S we keep track of:
- Topology (i.e. the number of holes) of F
- Size (number of pixels in F)
- Center of mass of F
- A binary *boundary flag* indicating whether F contains a pixel on the boundary of the slice.

All of the above quantities are updated each time a new pixel p is inserted into S. Logical OR is performed on the boundary flags of components intersecting p to obtain the boundary flag of the component containing p after its insertion into S. The size of the new component is obtained by summing the sizes of the components adjacent to p and adding 1 (to account for p). Its center of mass can be obtained by properly weighting (proportionally to the size) the centers of mass of the adjacent components and the p's coordinates. Finally, the number of holes of the new component can be determined as follows. If all 8 neighbors of p are outside S, a new component with no holes (containing only p) is introduced. If all 4 edge neighbors of p are already in S, a hole disappears from the component that will contain p after it is added to S. Otherwise, the number of holes of the new component is equal to $h+b-a$, where a stands for the number of components adjacent to p before it is added to S, h is the total number of holes these components have and b denotes the number of components in the intersection of the boundary of p and the set S just before inserting p.

Admissible components

An *admissible component* of S is a connected component of no more than 900 pixels, with no holes and containing no boundary pixels. 900 is intended to be the upper bound on the size of the section through the vessel of interest. With each admissible component F, we record its characteristic at the moment of its

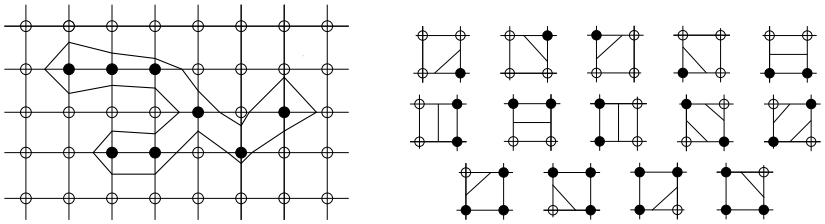

Figure 4: Computing 2D contours. Filled disks are centers of pixels in a connected component F of the set S, circles are centers of other pixels and the thick line is the contour defined by F. By a *grid interval* we mean an interval connecting the center of a pixel with the center of one of its four neighbors (on the right, on the left, below and above). *Grid squares* are squares formed by four grid intervals. The contouring algorithm generates one or two contour intervals for each grid square Z with at least one vertex in F and at least one vertex outside F. The intervals connect pairs of points on the edges of Z in a way depending on which vertices of Z are in F and which are not (all 14 possible cases are shown on the right). The endpoints of the intervals are computed as points with intensity equal to the threshold, assuming that the intensity varies linearly along the grid intervals.

slowest expansion since the last topological event involving that component. Expansion speed is intended to measure the average speed with which the boundary of F expands as the threshold decreases. Since F is a discrete set (union of pixels), we estimate the expansion speed of an admissible component F based on the growth of the *contour* defined by that component rather than the component itself. To compute contours, we employ the classical Marching Squares algorithm (described in Figure 4 for completeness).

The expansion speed of a component F at threshold t_0 is defined as $-\frac{dA(t)/dt_{|t=t_0}}{P(t)}$ where $A(t)$ is the area enclosed by the contour defined by F and isovalue t and $P(t)$ is the perimeter of that contour. We approximate the expansion speed between t_1 and t_2 using the following formula based on the finite difference approximation of the derivative $dA(t)/dt$:

$$\bar{R}(F;t_1,t_2) := \frac{A(t_2)-A(t_1)}{(t_1-t_2)P(\frac{t_1+t_2}{2})}. \tag{1}$$

If $t_1 = t_2$, we set $\bar{R}(F;t_1,t_2)$ to ∞.

The expansion speed estimate (1) is computed whenever a new pixel p is added to an admissible component F, with t_2 equal to the intensity of p and t_1 equal to the intensity of the latest pixel that was added to F before p (Figure 5). For any admissible component F, we keep track of:
- The minimum value of the expansion speed estimate (denoted by $R(F)$) since the last topological event involving F,
- The center of mass $M(F)$ of F for the threshold that yields the minimum expansion speed.

Whenever an admissible component F undergoes a topological event or becomes inadmissible, $M(F)$ is inserted into the set of core points. The *uncertainty measure* given by $R(F)$ is recorded with that point.

Core point set

We experimented with a few ways of selecting slices for the analysis using the above described method. For example, one can use all axis oriented slices to obtain results close to those reported here. Using more slices could improve the quality of the output at the expense of running time. The results reported in this paper were obtained using slices perpendicular to vectors with coordinates in $\{0,1,-1\}$. For the purpose

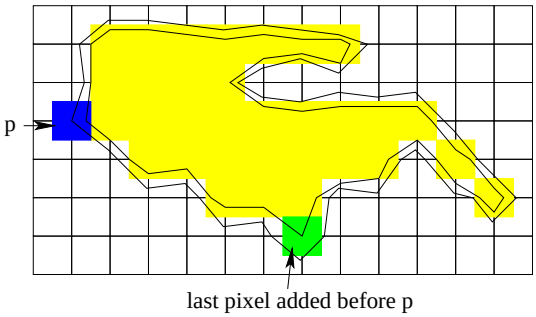

Figure 5: Computing the expansion speed of a component of F when p is added to the component. The inner polygonal line is the contour corresponding to isovalue t_1 (intensity at the last pixel added before p) and the outer line is the contour corresponding to isovalue t_2 (intensity of p). To compute the expansion speed of F, we divide the area between the two contours by $t_1 - t_2$ times the length of the contour corresponding to isovalue $\frac{t_1+t_2}{2}$ (half way between the two shown in the figure).

of core point generation, we assumed that the voxel centers have integer coordinates (thus, we ignored the anisotropy). Pairs of consecutive slices were 1 apart for each of the slicing directions and the spacing between the samples was the same as in the original dataset (1 along each dimension).

For each of the slicing directions, we generate the core point sets from the corresponding family of parallel slices and select $15,000$ points of lowest uncertainty (i.e. lowest $R(F)$ described in the previous section). This results in $195,000$ points since there are 13 slicing directions (perpendicular to vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, $(1,1,0)$, $(0,1,1)$, $(1,0,1)$, $(1,-1,0)$, $(0,1,-1)$, $(1,0,-1)$, $(1,1,1)$, $(1,-1,1)$, $(1,1,-1)$, $(1,-1,-1)$).

2.3 Filtered core point set

The core point set is cleaned up to remove some of the outliers. The filtering procedure works as follows. We build the Euclidean minimum spanning forest $\mathcal{F}$ of the graph with vertices at the core points and with edges connecting any pair of core points no more than 0.9mm away. Now, we treat leaf vertices of $\mathcal{F}$ as branch endpoints. The branch of a leaf vertex v is defined as the simple path (i.e. path that uses no vertex more than once) in the forest starting at v and ending at a vertex of degree other than 2. The length of the branch is defined as the number of edges of the branch. We iteratively select the shortest branch and remove its edges from the $\mathcal{F}$ until it has no branches of length 4 or less. Finally, we remove all isolated (degree-0) vertices in the resulting graph from the core point set obtaining the *filtered core point set*.

2.4 Core graph

The vertex set of the core graph is obtained by adding the starting and end point (S and E) of the vessel to be traced to the set of filtered core points. The edges of the core graph $\mathcal{C}_D$ (where D, a positive real number, is a parameter that allows one to control the number of edges of the graph) connect pairs of vertices that are no more than D mm away. Note that we could use the full graph; the bound D on the edge length is introduced only to reduce the number of edges and therefore the running time. The weights are assigned to the edges according to the following rules. Edges of length $d <= 0.9$mm are assigned the weight of $d/100$. For an edge connecting a point p with point q, the weight is given by $\min\left\{\sqrt{|\vec{pq}|^2-(\vec{pq}\cdot T_p)^2}, \sqrt{|\vec{pq}|^2-(\vec{pq}\cdot T_q)^2}\right\} + 0.8 * |\bar{pq}|$, where T_p and T_q are estimated tangent vectors at p and q. Note that $\sqrt{|\vec{pq}|^2-(\vec{pq}\cdot T_s)^2}$ is the length of the component of $\vec{pq}$ perpendicular to T_s for $s \in \{p,q\}$. Intuitively, the edge weights are designed to penalize long edges and to additionally penalize

Table 1: Average overlap per dataset

Dataset nr.	OV %	OV score	OV rank	OF %	OF score	OF rank	OT %	OT score	OT rank	Avg. rank
8	90.6	62.4	–	48.2	34.0	–	92.7	46.5	–	–
9	93.2	72.6	–	76.1	64.8	–	94.7	72.4	–	–
10	91.4	58.3	–	33.5	16.9	–	91.4	58.2	–	–
11	94.7	57.5	–	28.6	24.8	–	94.7	48.5	–	–
12	89.2	46.1	–	24.8	13.9	–	93.0	46.8	–	–
13	98.3	70.8	–	92.5	57.8	–	98.7	74.4	–	–
14	98.0	72.0	–	45.2	38.2	–	97.9	61.5	–	–
15	99.7	85.2	–	92.5	72.3	–	99.7	74.8	–	–
16	96.9	62.1	–	54.6	40.3	–	97.2	61.1	–	–
17	90.2	64.1	–	15.4	20.4	–	90.2	47.3	–	–
18	98.3	86.6	–	77.6	64.1	–	98.3	74.2	–	–
19	98.5	84.0	–	73.1	63.3	–	98.5	74.2	–	–
20	93.3	58.8	–	46.0	26.3	–	93.3	46.8	–	–
21	98.1	80.4	–	91.1	82.0	–	98.2	74.5	–	–
22	99.6	87.4	–	98.0	86.5	–	99.6	87.3	–	–
23	98.4	73.2	–	66.2	46.0	–	98.4	61.7	–	–
Avg.	**95.5**	**70.1**	**–**	**60.2**	**47.0**	**–**	**96.0**	**63.1**	**–**	**–**

edges $\bar{pq}$ whose direction is far away from the estimated tangent direction at both p and q. Note that there is a sharp discontinuity in the weight formula: weights of edges of length slightly above 0.9mm are much higher than weights of edges of length slightly below 0.9mm. Thus, to some extent, the algorithm follows the approach of [7]: it finds long dense streaks of core points and then connect through gaps in the streaks.

The tangent vector T_p at a point p (required for the edge weight computation) is estimated using the least squares line fit to filtered core points less than 2mm away from p. T_p is a unit vector parallel to the optimal line. We set T_p to zero for $p \in \{S, E\}$. This means that the terms involving T_S and T_E are not used when computing the edge weights.

2.5 Shortest path and shortcutting

In order to track the vessel between points S and E, we find the shortest path in the core graph $\mathcal{C}_{16}$ connecting S and E using the Dijkstra's algorithm [2]. If such a path does not exist (which does not happen for any of the datasets provided by the workshop organizers) we attempt to find the shortest path in the core graph with doubled edge length bound, i.e. we consider $\mathcal{C}_{32}$, $\mathcal{C}_{64}$ etc until the path is found.

We then apply a simple shortcutting technique in order to improve the smoothness of the output path. A shortcutting operation is equivalent to removing a vertex (other than the first or the last) from the path. We select the vertex to be removed based on a simple angle criterion. If $v_0, v_1, \ldots v_N$ are the consecutive points along the current path, we search for $i \in \{1, 2, \ldots, N-1\}$ such that the angle $\angle v_{i-1} v_i v_{i+1}$ is the smallest. If this angle is less or equal than 90 degrees, we remove v_i from the path and apply the same procedure to the resulting path. If the angle is more than 90 degrees, the shortcutting process is terminated.

3 Experimental results

The results obtained for the test datasets provided by the Workshop organizers are shown in Tables 1,2 and 3. The process of collecting the data and determining the ground truth is described in [5].

References

[1] Katja Bühler, Petr Felkel, and Alexandra La Cruz. Geometric methods for vessel visualization and quantification - a survey. Technical Report TR_VRVis_2002_035, VRVis Research Center, Vienna,

Table 2: Average accuracy per dataset

Dataset nr.	AD mm	AD score	AD rank	AI mm	AI score	AI rank	AT mm	AT score	AT rank	Avg. rank
8	0.50	38.3	–	0.38	40.1	–	0.47	38.9	–	–
9	0.55	26.8	–	0.32	28.7	–	0.50	27.3	–	–
10	0.53	25.2	–	0.39	27.1	–	0.53	25.5	–	–
11	0.49	31.2	–	0.42	31.9	–	0.49	31.2	–	–
12	0.48	27.3	–	0.36	29.6	–	0.43	28.4	–	–
13	0.37	28.5	–	0.36	28.7	–	0.36	28.8	–	–
14	0.43	35.0	–	0.40	35.5	–	0.43	34.7	–	–
15	0.34	29.3	–	0.34	29.4	–	0.34	29.6	–	–
16	0.41	24.6	–	0.38	24.9	–	0.42	24.0	–	–
17	0.55	47.3	–	0.38	48.3	–	0.55	47.3	–	–
18	0.34	28.9	–	0.33	29.2	–	0.34	28.9	–	–
19	0.37	37.0	–	0.36	37.3	–	0.37	36.9	–	–
20	0.58	30.5	–	0.43	32.5	–	0.58	30.5	–	–
21	0.32	27.8	–	0.31	28.0	–	0.31	28.3	–	–
22	0.36	28.5	–	0.35	28.5	–	0.36	28.5	–	–
23	0.36	31.4	–	0.34	31.7	–	0.36	31.4	–	–
Avg.	**0.44**	**31.1**	**–**	**0.36**	**32.0**	**–**	**0.43**	**31.3**	**–**	**–**

Table 3: Summary

Measure	% / mm min.	% / mm max.	% / mm avg.	score min.	score max.	score avg.	rank min.	rank max.	rank avg.
OV	74.2%	100.0%	95.5%	40.1	100.0	70.1	–	–	–
OF	0.0%	100.0%	60.2%	0.0	100.0	47.0	–	–	–
OT	74.1%	100.0%	96.0%	41.1	100.0	63.1	–	–	–
AD	0.26 mm	1.19 mm	0.44 mm	18.8	78.1	31.1	–	–	–
AI	0.26 mm	0.52 mm	0.36 mm	19.7	79.6	32.0	–	–	–
AT	0.26 mm	0.96 mm	0.43 mm	18.3	78.1	31.3	–	–	–
Total							**–**	**–**	**–**

Austria, 2002. 1

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990. 2.2, 2.5

[3] Petr Felkel, Rainer Wegenkittl, and Armin Kanitsar. Vessel Tracking in Peripheral CTA Datasets – An Overview. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, page 232, Washington, DC, USA, 2001. IEEE Computer Society. 1

[4] Cemil Kirbas and Francis Quek. A review of vessel extraction techniques and algorithms. *ACM Comput. Surv.*, 36(2):81–121, 2004. 1

[5] Coert Metz, Michiel Schaap, Theo van Walsum, Alina van der Giessen, Annick Weustink, Nico Mollet, Gabriel Krestin, and Wiro Niessen. 3D Segmentation in the Clinic: A Grand Challenge II - Coronary Artery Tracking. *The Insight Journal, Sepcial issue '3D Segmentation In The Clinic: A Grand Challenge II at MICCAI 2008 - Coronary Artery Tracking', to appear*, 2008. 3

[6] National Center for Health Statistics. Health, United States, 2004, With Chartbook on Trends in the Health of Americans. Hyatsville, Maryland, `http://www.cdc.gov/nchs/hus.htm`. 1

[7] A. Szymczak, A. Stillman, A. Tannenbaum, and K. Mischaikow. Coronary vessel trees from 3D imagery: A topological approach. *Medical Image Analysis*, 10(4):548–559, 2006. 1, 2.2, 2.4

[8] Andrzej Szymczak, Arthur Stillman, and Allen Tannenbaum. 3D Models of Airway Trees from CT Scans: A Point-Based Approach. `www.mines.edu/~aszymcza/papers/bronchi-short.pdf`. 1, 2.2