Construction of Virtual Environment for Endoscopy

S. Loncaric and T. Markovinovic and T. Petrovic and D. Ramljak and E. Sorantin*

Department of Electronic Systems and Information Processing
Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
E-mail: sven.loncaric@fer.hr

*Section of Pediatric Radiology
Department of Radiology, University Hospital Graz, Austria
E-mail: erich.sorantin@kfunigraz.ac.at

Abstract

In this paper, a technique is presented for CT image analysis and visualization of the bronchial airways. The technique provides a non-invasive way to examine the interior of the bronchial tubes and to detect various properties of the tubes such as abnormal morphology caused by foreign objects stuck in the airways or some other disease. The input to the procedure are chest images obtained by spiral computed tomography (CT). 3-D neural network-based segmentation of CT images is performed to extract the airways. The resulting 3-D binary volume representing the location of the airways is thinned to extract the medial axis of the bronchial tubes. The marching cube algorithm is used to perform a triangulation of the airway surface. The extracted axis and the triangulated surface is converted to Virtual Reality Modeling Language (VRML) format. The virtual environment in VRML format is then used for inspection and visualization of the bronchial tube interior.

1 Introduction

Virtual reality techniques have been used in many areas of human activity [1]. In particular, medicine is one large field where virtual reality techniques can applied [2, 3, 4]. The main reason for this is that human body is a complex biological system with many subsystems that are mutually interconnected. Modern diagnostic techniques are used to represent and visualize the inside of human body but in many cases it is difficult or impossible to represent complex three-dimensional (3-D) structures and their spatial relationship. Here, virtual reality techniques are used not only for visualization but also for interactive exploration of 3-D biological environment.

2 Methods and procedures

In this paper, a technique is presented for CT image analysis and visualization of the bronchial airways. This technique provides a non-invasive way to examine the interior of the bronchial tubes and to detect various properties of the tubes such as abnormal morphology caused by foreign objects stuck in the airways or some other disease. Virtual endoscopy of the bronchial tubes, or virtual bronchoscopy, eliminates some restrictions imposed in classical endoscopy. For example, in classical endoscopy, it is impossible to position endoscope outside the airways because this would cause an injury. In virtual endoscopy, there are no restrictions on the position of the viewpoint so the operator can also have view of structures outside the airways the would be impossible to see using classical endoscopy.

However, virtual endoscopy cannot eliminate the conventional endoscopy. The drawback of the presented technique, and of virtual endoscopy in general, is that it is impossible to see the actual tissue. In many cases medical experts can draw many conclusions about the nature of disease from the texture and color of the tissue. The image that the operator can see through the eye piece of the endoscope is not present in virtual endoscopy.

The technique presented in this paper consists of several steps. The outline of the procedure is shown in Figure 1. The input to the procedure are chest images

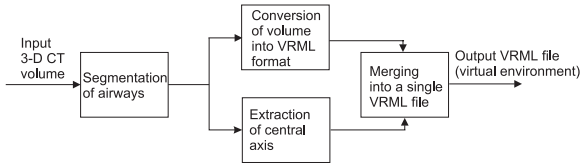

Figure 1: The outline of the procedure for virtual endoscopy.

obtained by spiral computed tomography (CT). The 3-D image analysis of the data is performed to extract the airways from the images. This part of the procedure includes preprocessing of images, followed by image segmentation. This step is challenging by itself because the bronchial tubes are not always well recognized in the CT image. The segmentation method is based on a neural network trained using some of the input images. The result of the procedure is a 3-D binary image representing the location of the airways.

In the consequent step, the 3-D binary image is converted from raster format to Virtual Reality Modeling Language (VRML) format [5]. The procedure uses the marching cube algorithm [6, 7] to perform triangulation of bronchial surfaces. The triangles produced by the algorithm are represented in the VRML format. VRML format has become a popular format for representation of 3-D virtual environments and for its transfer through the World Wide Web (WWW) over the Internet [5, 8]. VRML language provides the interaction with the virtual environment. The additional flexibility of the VRML paradigm is in inclusion of Java nodes which provide user with possibility of adding their own Java code to further customize properties of their virtual environment. Java has become a popular language of the new generation because of its platform independence and network support [9].

In the next step, a 3-D central axis of the bronchial tubes is extracted from the 3-D segmented airways. The medial axis is used as a path for a fly-through animation of the airways. The medial axis is determined by computing the 3-D skeleton of the bronchial tubes volume. The obtained axis is smoothed and represented as a sequence of points which are used as camera and target points for the animation path. The camera and target points are integrated into the VRML file containing the triangulated airways. The resulting VRML model can be viewed using any VRML-enabled WWW browser, locally or over the network. Upon opening the VRML file a fly-through animation through the virtual bronchial tubes is shown. The user can also interactively travel through the virtual bronchial tubes to examine the structure in more detail.

2.1 Segmentation

Segmentation of the CT images is the first step in the visualization of the bronchial airways. The procedure is based on a neural network previously trained using some of the sample input images. Neural network is chosen because its ability to model complex non-linear functions and the fact that this function does not need to be known in advance since it is "discovered" during training phase [10]. This approach results in a relatively fast and accurate automatic segmentation and has been successfully used in other medical image analysis problems [11]. The output of this procedure is a 3-D binary image representing the location of the airways.

A fully-connected feed-forward network (multilayer perceptron) with three layers shown in Figure 2 is used. The input layer and the hidden layer have m neurons and the output layer has one neuron. Normalized gray levels of pixels from the $n \times n$ square neighborhood of the pixel to be segmented are inputs in the network. The network output is binary coded indicating whether the segmented pixel belongs to the airways or not. The images used in the network train-

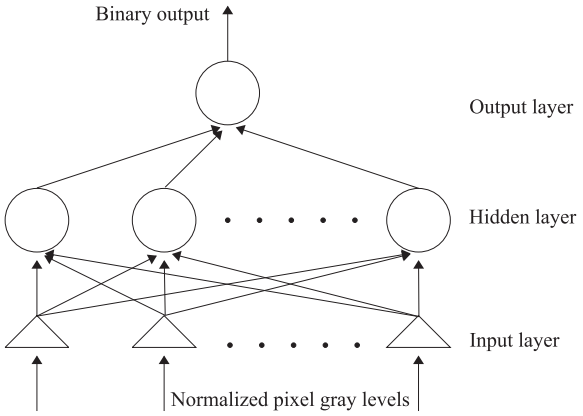

Figure 2: Network Topology.

ing are divided into three groups. Patterns derived from one group are used to train the network and patterns derived from another are used to test the trained network. The third group is used in testing of the segmentation program.

A number of networks is trained using varying number of neurons in input and hidden layer (m), area used (n) and training algorithm. The best network with respect to segmentation accuracy and speed is then chosen and implemented in the segmentation program.

The training is done using Stuttgart Neural Network Simulator 4.1 (SNNS 4.1) on a Sun Ultra workstation. Standard backpropagation training algorithm produced best results and is subsequently used. The best network performance is achieved with 61 neuron in input and hidden layer. An input receptive field of the size 31×31 as network inputs.

The segmentation program takes a sequence of images (like the one shown on the left side of Figure 3) as input and outputs a sequence of segmented (black and white) images (like the one shown on the right in Figure 3) which in fact represents a 3-D image representing location of the airways.

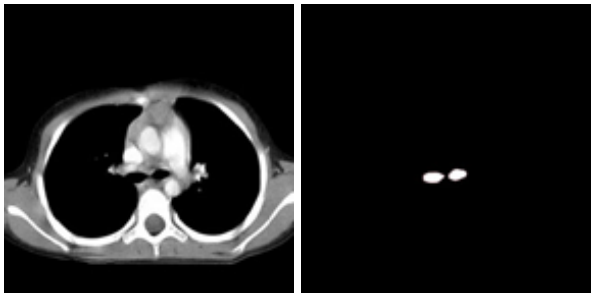

Figure 3: Sample input and output image.

The program is written in standard C++, thus making it portable to wide range of platforms, and finally tested using a third group of input images. The neural network is implemented using snns2c utility from the SNNS 4.1 package.

2.2 Thinning

In the thinning step, a 3-D central axis of the bronchial tubes is extracted from the 3-D segmented airways. The medial axis is used as a path for a flythrough animation of the airways. The medial axis is determined by computing the 3-D skeleton of the bronchial tubes volume.

Many thinning algorithms use distance transformation as the first step [12]. A value that represents the distance from the nearest edge of the 3-D shape is assigned to every point of the distance map. Background points are assigned zero value. Since the space is discrete, approximations are necessary. Most distance transformations use a 3x3x3 scanning mask which produces satisfactory results for many applications. Although the results are more accurate with 5x5x5 or even bigger neighborhood, the algorithms are complex and slow with almost no visible difference. In this paper an algorithm is presented that uses a 3x3x3 mask to scan the image.

For each of the 27 voxels in the mask, the distance must be defined from central voxel to other voxels. There are 6 voxels with distance equal to 1, 12 voxels with distance equal to $\sqrt{2}$ and 8 voxels with distance equal to $\sqrt{3}$. Since integer values are used in the transformation, the basic values are multiplied with 100. The mask in the form of the three two-dimensional layers is shown in Figure 4 (left).

173	141	173	173	141	173
141	100	141	141	100	140
173	141	173	173	140	172
141	100	141	141	100	140
100	0	100	100	0	99
141	100	141	141	99	140
173	141	173	173	141	172
141	100	141	141	99	140
173	141	173	172	140	172

Figure 4: The masks for standard (left) and modified (right) distance transformation.

The result of thinning in the discrete space is often not equal to actual 3-D axis that can be obtained in continuous space. In particular, the thickness of the obtained discrete axis may not be equal to one. For example, a 4x4x10 solid geometric figure does not have a unique medial axis in discrete environment. If a modified definition of a mask is introduced it is possible to obtain a medial axis that has thickness equal to one. In this work we have used the definition shown in Figure 4 (right). Distances from the central voxel to opposite voxels are not the same which artificially makes one of the voxels closer to the edge and thus the other voxel becomes a part of the medial axis. This results in a medial axis that has thickness equal to one, as desired. Every one of the thirteen axes that goes through the center has asimetric values from the opposite sides of the central voxel. The difference is very small in order to prevent the phase jump which could happen if the object is large enough. In that case larger values would be used.

Apart from the above described differences the algorithm is the same as most other distance transformation algorithms. The first step is to set every object voxel to a large number, that is larger than the real distance to the edge. For every voxel in the 3-D data the distance from the nearest background voxel is calculated in the following way:

Now that the data is transformed, a 3x3x3 neighborhood is sufficient to find the medial axis. In the

```
1:  repeat
2:    for all points in the 3-D space do
3:      if the value of the voxel is bigger the 0 then
4:        for each voxel in the 3x3x3 neighborhood
          do
5:          calculate its distance from the center and
            add it to the its value
6:          if the calculated value is smaller than the
            value of the central voxel then
7:            calculated value is the new value of the
              central voxel
8:          end if
9:        end for
10:     end if
11:   end for
12: until there are no changes in the values
```

Figure 5: The distance transformation algorithm.

3-D environment the medial axis is orthogonal to the 2-D plane in which the central point has the maximum value. For every voxel the algorithm checks if the voxel has the maximum value in any of the 2-D plane. If such a voxel is found it is than marked as the medial axis voxel. In most cases only the planes along the three basic axis must be checked. Including the diagonal axis produces many extra voxels which make it more difficult to determine the final axis along the object of interest. This is another problem of the discrete space, since in the continuous space only the three major axis would be sufficient. Nevertheless, it is always possible to use the additional planes if the results are not good enough.

The objects in 3-D environment could be very complex with many branches. Every such branch produces its medial axis which sometimes makes it difficult to find the one we are looking for. In this case the bronchical tube has two major branches and many smaller ones. It is up to the user to choose the axis he wishes to see in the fly-through animation. The best solution in this case is the use of VRML for 3-D visualization. Every point in the 3-D data is presented with the point in the VRML file (actually there are three points very close to each other for every source point which results in better visualization of smaller objects). Every other layer is in different color for better visualization. Medial axis points are presented with small spheres, each with its own number. The number can be seen by putting the cursor in the VRML browser over the sphere. The user than chooses the first and the last voxel of the desired medial axis. The shortest distance between these two

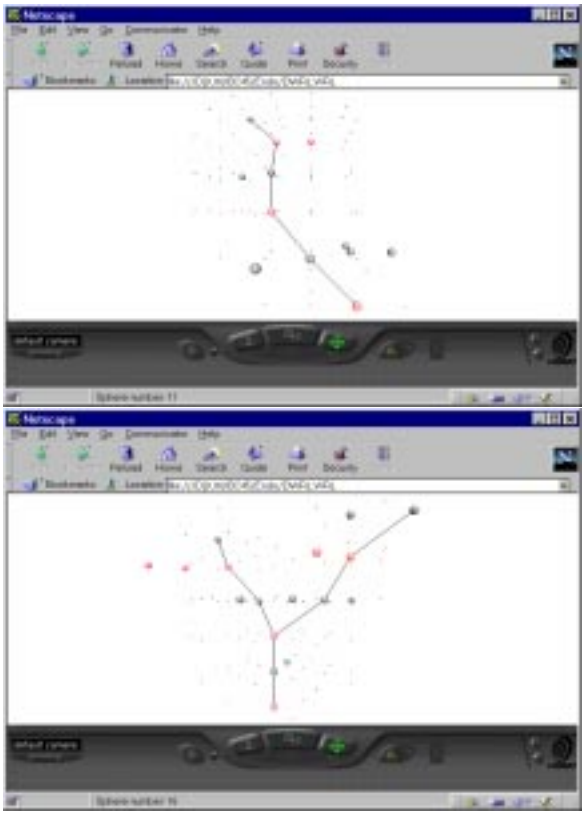

Figure 6: The upper figure shows small test objects. The lower figure shows a multiple axes example.

voxels is calculated using only the marked voxels as a path. The method eliminates every other unnecessary branch. Screenshots from the small test object are shown in the upper part of Figure 6. It is also possible to choose multiple axes which is in case of bronchial tube absolutely necessary. Examples of the multiple axes choosing is shown in the lower part of Figure 6. The obtained axis is smoothed and represented as a sequence of points which are used as camera and target points for the animation path. The camera and target points are integrated into the VRML file containing the triangulated airways. The resulting VRML model can be viewed using any VRML-enabled WWW browser, locally or over the network. Upon opening the VRML file a fly-through animation through the virtual bronchial tubes is shown. The user can also interactively travel through the virtual bronchial tubes to examine the structure in more detail.

2.3 Marching cubes triangulation

In order to create a VRML model of airways from segmented CT images, we use the marching cube sur-

face extraction algorithm [6]. The marching cubes algorithm was designed to extract surface information from a 3-D field of values.

The basic principle behind the marching cubes algorithm is to subdivide space into a series of small cubes. The algorithm then marches through the space, visiting each of the cubes, testing the corner points and replacing the cube with an appropriate set of polygons. The set of all polygons that are generated approximates the surface of the original 3-D object.

Each cube corner has a brightness value (0-255) and an interpolation is performed that is based on brightness values. Each vertex of a triangle lies on the edge of the cube so we check brightness values of two cubes vertices on the ends of that particular edge and interpolate position of the triangle vertex. (E.g. if one vertex has value of 0 and the other 255, the triangle vertex will be in the middle of the edge).

The input data are bronchial CT images. The scans do not contain only airways but also include the whole abdominal region. Int he following step the extraction of the regions of CT scans that are of interest is performed. The extraction is done by applying segmented images of CT scans over the original CT scans in a form of mask.

The triangles produced by the algorithm are represented in the VRML format. The VRML format has become a popular format for representation of 3-D virtual environments and for its transfer through the World Wide Web (WWW) over the Internet. VRML language provides the interaction with the virtual environment.

The additional flexibility of the VRML paradigm is in inclusion of Java nodes which provide user with possibility of adding their own Java code to further customize properties of their virtual environment. Java has become a popular language of the new generation because of its platform independence and network support.

2.4 VRML model

The visual layout of the user interface consists of a VRML browser window and a Java frame window. In the browser window the user can look at and examine the model or fly-through it along a predefined path. The user is able to stop and restart the fly-through at any time. The flight is very convenient, considering limitations and difficulties of moving and orienting in virtual environments, e.g. entering small objects etc.

Taking into account orientation problems a Java frame window is added in which the user can see his/hers position (the camera position). Changes

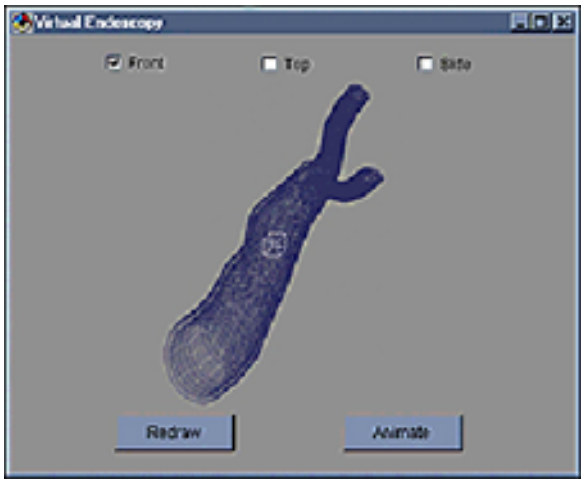

Figure 7: Wire-frame Java visualization.

in VRML browser window automatically result in changes in Java frame window. As shown in Figure 7 Java frame window shows a wire-frame model which is the best way to see inside object because of its transparency so the user can see his position while being inside the object. The user is given the choice of three views, front, top and side. The implementation of the wire-frame model is done in Java AWT.

Java and VRML coexist in the same virtual environment and cooperate between themselves. In fact, the virtual environment calls JVM (Java Virtual Machine) that interprets Java byte codes. The user is not aware of this and simply sees two windows that can interact as shown in Figure 8. It would be very difficult to develop this functionality in VRML without Java. The block diagram showing the actual hierarchy is shown in Figure 9.

VRML is a dynamical language and it is possible to alter the scene by adding or removing scene entities at run time. This feature can be used for altering surface of objects and their position, e.g. coronary artery or bronchus reacts when its pushed or touched (with catheter).

3 Conclusion

In this paper, the methods and procedures are presented that are used to generate interactive visualization of medical 3-D structures. The procedure performs analysis of CT medical images based on a neural network and extracts the volume of interest. A medial axis of the volume is computed to serve as the animation path for fly-through animation. The path

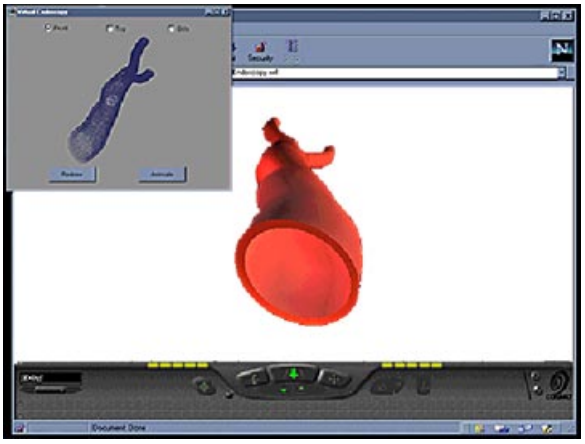

Figure 8: Graphical user interface written in Java and VRML.

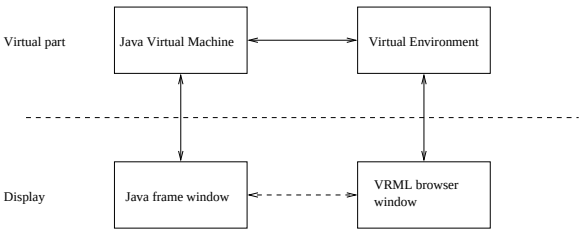

Figure 9: The organization of the Java/VRML environment interaction.

and the surface model are integrated into a VRML model of the structure. The structure can be interactively explored using a standard WWW browser that is VRML-enabled. The procedure is useful both for educational and diagnostic purposes. The software is developed on a SUN Ultra 1 workstation in Java and VRML languages.

References

[1] G. Burdea and P. Coiffet. *Virtual Reality Technology*. Wiley, 1994.

[2] J. M. Rosen, H. Soltanian, R. J. Redett, and D. R. Laub. Evolution of virtual reality: From planning to performing surgery. *IEEE Engineering in Medicine and Biology*, 15(2):16–22, 1996.

[3] J. A. Waterworth. Virtual reality in medicine: A survey of the state of the art. URL: http://www.informatik.umu.se/~jwworth/medpage.html.

[4] W. J. Greenleaf. Developing the tools for practical VR applications. *IEEE Engineering in Medicine and Biology*, 15(2):23–30, 1996.

[5] J. Wernecke J. Hartman. *The VRML 2.0 Handbook*. Addison Wesley, 1996.

[6] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21:163–169, 1987.

[7] A. Watt. *3-D Computer Graphics, 2nd ed.* Addison Wesley, 1993.

[8] R. Stuart. *The Design of Virtual Environments*. McGraw-Hill, 1996.

[9] D. Flanagan. *Java in a Nutshell*. O'Reilly, 1997.

[10] S. Haykin. *Neural Networks*. Prentice-Hall, 1994.

[11] S. Loncaric and D. Kovacevic. A method for segmentation of CT head images. In *Proceedings of the 9th Int'l Conference on Image Analysis and Processing*, pages 388–395. IAPR, 1997. Florence.

[12] G. Borgefors. Distance transformation in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.