Fast skeletonization algorithm for 3-D elongated objects

Ali Shahrokni[a,b], Hamid Soltanian-Zadeh[a,b,c], Reza A. Zoroofi[a]
[a]Elec. and Comp. Eng. Department, University of Tehran, Tehran 14399, Iran
[b]School of Intelligent Systems, Inst. for Studies in Physics and Math., Tehran 19395, Iran
[c]Radiology Department, Henry Ford Health System, Detroit, MI 48202, USA

ABSTRACT

A novel one-pass 3D thinning algorithm is proposed in this paper. 3D thinning can be regarded as an essential step in many image-processing tasks such as image quantification, image compression, motion tracking, path generation, and object navigation. The proposed algorithm generates both connected and non-connected skeletons. It is faster and more accurate than currently available techniques in the literature. In addition, it adaptively removes spurious branches of the skeleton, and hence, generates a smooth and refined skeleton, representing the essential structure of 3D elongated objects such as vessels, airways, and other similar organic shapes.

Keywords: 3D thinning; skeletonization; distance transform; path finding; medial axis; image compression; and topology preserving

1. INTRODUCTION

Recent advances in three-dimensional imaging technology and availability of sophisticated computers has introduced 3-D image processing techniques as a solution to everyday life as well as specialized problems. These applications include luggage-scanning systems, quality control systems for factory products, medical image processing, automated navigation and animation generation. Due to inherent large amount of data associated with 3-D images, it is desirable to deal with certain structural features of interest of imaged objects. Therefore representing objects by their main structure, or skeleton, is a very efficient way to deal with 3-D data with respect to memory usage and data compression while highlighting structural features of the objects at hand. The desired characteristics of a skeleton are as listed below.

1. Centeredness with respect to the object boundary.
2. Independence from boundary noise and local distortions.
3. Preserving the structure of the original object (topology preserving).
4. Connectivity preserving.
5. Computational simplicity.
6. Object reconstruction capability.

Some of the above features are attractive for special applications. For example, reconstruction capability is required for image compression techniques,[1] or connectivity preserving is necessary for automatic navigation or animation control applications.

In this work, a distance transform-based 3-D thinning method has been described which possesses the above characteristics. Moreover, it is flexible to allow generation of connected or non-connected skeletons based on the application requirement. This paper is organized as follows. In Section 2, a review of previous approaches to 3-D thinning and their relations to the proposed method are presented. Section 3 discusses distance transform and voxel coding concept. Medial axis extraction and branch labeling is dealt with in Section 4. A method to refine the medial axis is described in Section 5. Reconnection of the skeletons is discussed in Section 6. Experimental results of the presented algorithm on real and synthetic elongated objects are given in Section 7. Section 8 presents the conclusions.

2. RELATED WORK

2-Dimensional thinning methods[2, 3, 4] have received major consideration and were studied extensively over the past decade. Most of the methods found in the literature on 3-D thinning are in effect extensions of 2-D methods. The approach introduced in reference[5] uses a priori knowledge of the origin of retinal vessels to detect and track the medial axis. The detection and tracking process is implemented by application of matched second order Gaussian filtering. Extended Kalman filtering is utilized in the tracking process to predict the next point on the vessel based on the knowledge of the state vector which takes into account position of the most recent visited point, orientation of the vessel and distance between consecutive points. This approach is robust and efficient in 2-D provided that a priori knowledge about the origin of objects

is available. The 3-D extension of this procedure would be practically non-efficient. Furthermore, a uniform distribution function has been used to model the intensity of vessels, which is not true in all situations.

Iterative peeling of the boundary of the object is a traditional way to thin objects. In this method, boundary voxels are deleted iteratively to reach the object skeleton. To prevent over thinning of the object, deletion of the object end points should be avoided. Furthermore, only boundary voxels whose removal does not affect the connectivity and structural topology of the object should be removed. Such voxels are called simple points.[6] Sequential simple point detection and removal is an exhaustive task for large datasets. A parallel thinning method has been described in reference[6] which detects and removes multiple voxels at each iteration using deletion templates which are designed to preserve connectivity and topology of the objects. However, these methods rely on local information to obtain skeletal points. This constraint increases the sensitivity of the algorithm to noise and local distortion.

Methods based on distance transform[7, 8, 9, 10, 11] tend to determine medial points by locating voxels which lie farthest with respect to the boundary of the object on its cross section normal to the local major axis. Basically, distance transform of object voxels with respect to boundary voxels is used to determine maximal balls (disks) within a 3-D (2-D) object. The maximal balls are sphere-like sets of connected voxels with maximum radius, which are completely contained in the object but not in any other ball in the object. The radius of maximal balls is directly related to the value of the distance transform of their central voxels. It has been shown that any object can be fully represented by the set of its maximal balls.[8] Therefore, the set consisting of centers of the maximal balls and their radii (value of the distance transform at the central voxel) is sufficient to reconstruct the object. Gagvani and Silver provided an even more efficient way to obtain skeletal voxels, while retaining the reconstructiblity property, by keeping only those maximal balls which are not contained in the union of other balls. The proposed work further employs a thickness parameter, which controls the thickness of generated skeletons by setting a threshold on selection of maximal ball centers. However, the generated skeletons form a disconnected set of voxels that need further processing to be suitable for many applications. Moreover, it is not also easy to distinguish between different structural branches of the object.

The work represented in this paper is closely related to the work of Zhou and Toga.[7] They have designed a method based on the extension of traditional distance transform concept. A voxel coding technique has been introduced that transforms a given image to a distance image, where distance is measured from a single given *'seed'* voxel or a set of voxels. If boundary voxels are taken as the seed set then the resulting distance image would be the traditional distance transform of the original image. Zhou and Toga used two code fields - Boundary-Seeded (BS) and Single-Seeded (SS) - to approximate the distance of object voxels from boundary voxels and a single reference point respectively. According to reference,[7] to form the initial skeleton of the object, voxels with a local maximum SS-field code are connected to the reference point by the shortest but centered path of voxels selected by evaluation of BS-field and SS-field of intermediate voxels. Multi-branching of the object is dealt with by creation of a dynamic list of branching nodes and generating paths from these nodes in subsequent iterations of the program. The algorithm recursively uses the SPE (Shortest Path Extraction) procedure, which implements a local SS-Field to extract centerlines, refine, and connect them. This procedure works on individual voxels to perform refining. The process of extraction of centerlines, corresponding to the clusters stored in a dynamic array,[7] is also recursive and therefore time consuming. We bypass these steps by establishment of parent and child relationship during the identification of medial voxels. This would allow individualized access to different branches as a whole by means of their code for further processing including refining. Our algorithm integrates the process of skeletonization and branch labeling, rendering efficiency to the algorithm.

3. VOXEL CODING

3.1 Distance transform

Since it is neither easy nor efficient to work with the Euclidean distance, an approximate measure of distance is used. Distance transform (DT) is an extensively studied concept and is used to provide an approximation to the minimum Euclidean distance of object voxels from boundary or background voxels. Usually Euclidean distance is approximated by a set of local integer distances to measure costs of traveling from a voxel to one of its 26-neighbors. Using integer values simplifies the computations.[9] The best integer approximation of local distance for moving to a 6- 18-, or 26-neighbor is 3, 4, or 5, respectively, and is denoted as *<3,4,5>*.

3.2 Code fields

As originally mentioned in reference,[7] distance transform can be generalized to generate multiple transforms from a single image. For the purpose of skeletonization, two distinct distance transforms are applied to the input dataset. The object

boundary is used as the seed set for the first transform. The result would be the same as the traditional minimum distance field and is used to assure centeredness of the skeleton. The algorithm for the implementation of this transform is given in Figure 1. Another field is generated using a single object voxel as the origin of distance computation. Note that the algorithm for computation of SS-field is similar to the one in Figure 1 except for the seed set and the distance metric used. For the purpose of simplicity the *<1,2,3>* metric was used to compute SS-field instead of traditional *<3,4,5>* metric. SS-field and BS-Field for a test image are shown in Figures 2-*a* and 2-*b*, respectively. The intensity of the image pixels shows the value of the respective field, where black is the minimum value (used for background) and white is the maximum value. The single seed of the SS-field (also called the Reference Point) is marked with an asterisk in image 2-*b*.

3.3 Selection of reference point

The Reference Point (RP) will be the starting voxel of the final skeleton. RP plays a fundamental role in the results of skeletonization algorithm. Therefore, considerable attention should be paid to its selection. Automatic selection of RP should guarantee that the selected point is close to the projected ends of the object. An automatic approach has been introduced in reference[7] that selects the last coded object voxel in an arbitrarily seeded SS-field as the RP. This method is too time consuming. We used a raster scan to detect 2-D connected components starting from the first frame to the last. The area of all connected components found in the first non-empty frame is computed and compared to a predefined threshold. If the greatest area was higher than the threshold, then the search for connected components is stopped and a voxel belonging to the component with the greatest area is selected as the RP. This voxel is selected so that it lies approximately on the major axis of the 2-D component. This is done by computation of the bounding box of the component and moving along the leftmost line of the bounding box-if its width is greater than its height, and the topmost line otherwise. The first object voxel found on this line will be marked to be the RP. This method of RP selection is fast and allows automatic object selection and segmentation. It also facilitates further processing and automatic skeleton generation for multiple objects. However, there is possibility of selecting RP on the minor axis if the object cross-section is too complex and twisted.

-Initialize the "Seed Set" with the object boundary points
-Set distance transform of object voxels to ∞, background voxels to a negative value, and Seed voxels to zero
While Seed Set is not empty do:
For all voxel in the Seed Set do:
-Get the distance transform of its neighboring object voxels.
-Compare this value with the new DT value of this voxel based on proximity of the current seed voxel.
-If the new value is smaller than the current value, then change the value of distance transform at this voxel to the new value and add the voxel to the seed set.
Until no DT value is changed

Figure 1. Algorithm for computation of BS-field.

4. MEDIAL AXIS EXTRACTION

4.1 Cluster graph

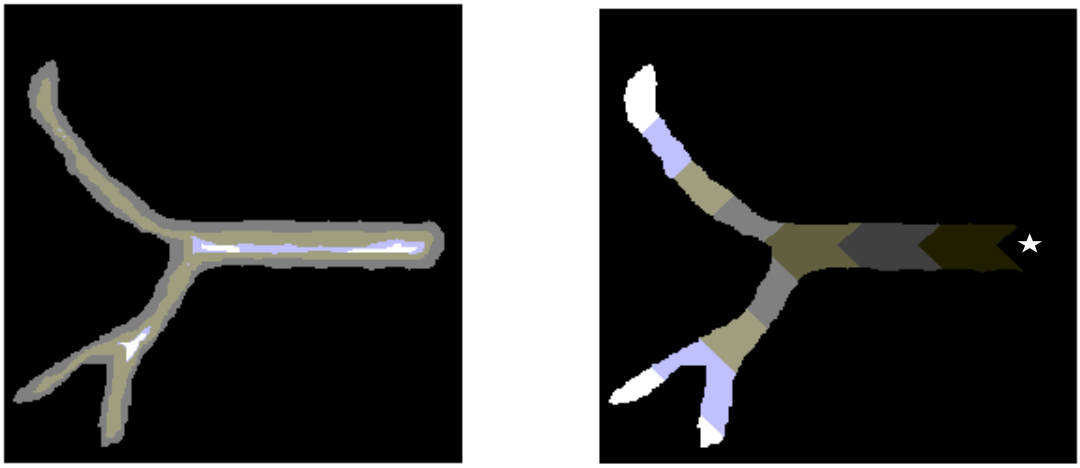
SS-field provides the basis for representing the 3-D object as a directed graph. This feature of SS-field is also very useful for object segmentation; because only those voxels that are connected to the reference point will be SS-field coded. A set of connected voxels having the same SS-field is called a cluster and interconnection of these clusters form a directed graph.[7]

4.2 Medial point extraction

A voxel can be selected as a representative for every cluster in the segmented object. To ensure centeredness of the cluster representative, among cluster voxels a voxels with the maximum value of BS-field will be selected. This voxel is called the medial voxel of each cluster.[7] If there are more than one such voxels, closest voxel to the center of mass of the cluster will be selected. The basic idea of skeleton generation is to obtain and connect the medial points of all object clusters. This set is shown to give the skeleton of the object.[7]

A different approach from Zhou's has been taken to extract the medial paths (sets of medial points). The algorithm is summarized in Figure 3. A path is defined as the smallest set of object voxels connecting two points. A medial path

associated with each path is derived from the original path by replacing its voxels by the medial point of the corresponding clusters. We perform the search for medial points starting from the reference point and continue along the cluster graph in an SS-field value ascending order. In each iteration of the algorithm, the clusters corresponding to the step-by-step increasing SS-field value are located and their medial points are identified and saved in a list of skeletal voxels. A code is associated with each of the medial points, which determines to which branch of the object they belong. The details of this procedure are discussed in the next section.



(a) (b)

Figure 2. (a) BS-field and (b) SS-Field of an object. The colors denote the value of the transform. RP is marked with an asterisk.

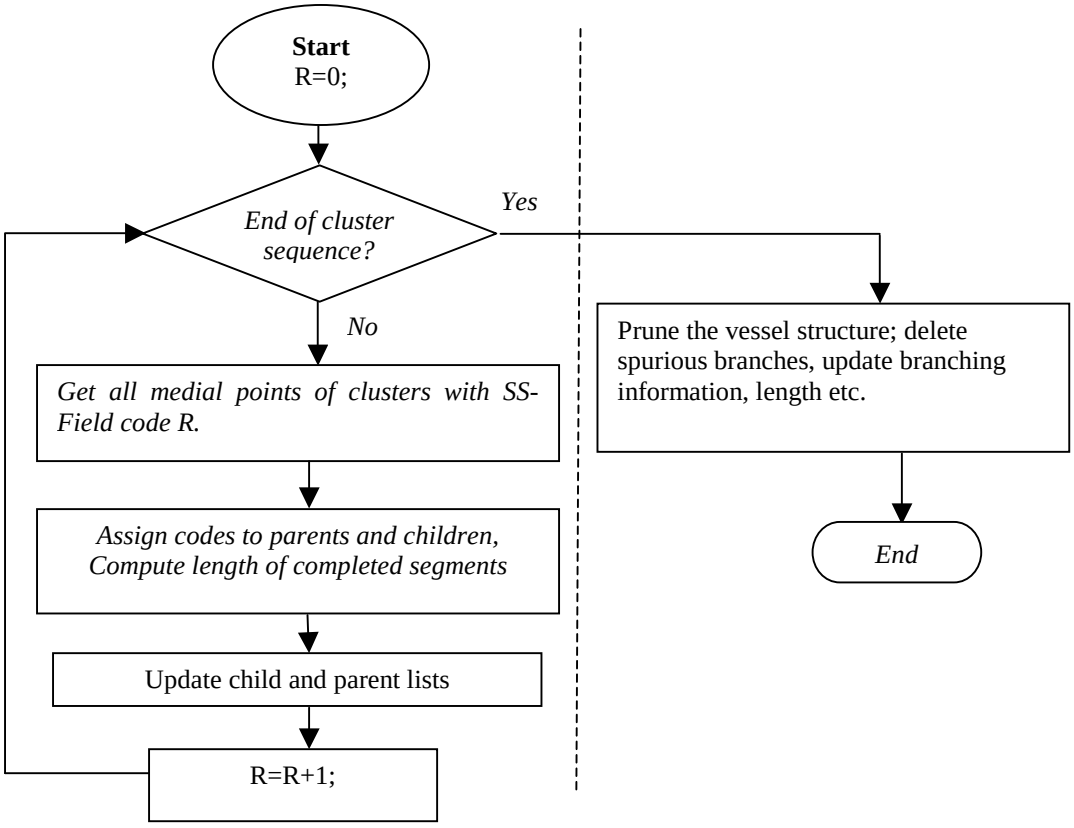


Figure 3. Block diagram of the thinning algorithm, the left part is the initial skeleton generation algorithm and the right part is the refining process.

4.3 Parent and child relationship

The 16-bit code associated with each medial point found consists of two segments, assuming the object has no more than 256 nodes. The 8-bit head and tail segments denote, respectively, starting and ending node of the branch on which the medial voxel is located. A list called "*ParentList"* is created which initially holds RP as the single parent of the subsequent medial points. Any newly found medial point corresponding to the next value of SS-field is saved in a list called "*ChildList"*. Then the parent of each voxel in this list is selected from voxels in the *ParentList*. Euclidean distance is used to find the nearest voxel. This simplifies the process and is accurate because of the small distances between these points. After all the voxels on the *ChildList* have been assigned parents, one of the following conditions will occur for a parent voxel and its children.

(i) A parent voxel has no children voxels.
In this case the parent voxel is a branch-ending node. It is possible to compute the length of this branch by counting the number of medial voxels having the same code as the parent code. This length is saved in a list called "*CodeList"* containing branch labels and their lengths. This list is later used to refine the skeleton.

(ii) A parent voxel has exactly one child voxel.
This happens for the middle voxels along an object branch. Therefore, only the parent code should be passed to the single child and the algorithm is resumed by selection of the next higher SS-field value and the corresponding clusters.

(iii) A parent voxel has more than one children voxels.
This is the case when a parent voxel is a branching node and multiple branches emanate from a single stem. Therefore, new codes should be generated for the new branches (nodes). The code is formed by attaching a new 8-bit tail code -which is equal to the number of nodes located so far plus one- to the end of the tail code of the parent segment. Therefore, the head segment of each branch code identifies its parent node. The new codes are added to the *CodeList* for future pruning and feature description.

Finally, the *ParentList* is replaced by the *ChildList* and the *ChildList* is reset. The algorithm is repeated until the clusters associated with the maximum value of SS-field are processed.

5. SKELETON REFINEMENT

Due abrupt discontinuity of cluster voxels at the ending parts of the object, small False Positive Branches (FPB) appear at these areas. To remove such branches, we impose two conditions to detect and delete excessive skeletal "hairs". A branch segment is qualified for deletion if *1)* its length is smaller than a predefined threshold and *2)* it doesn't possess any child branches.[12] Both of these conditions can be checked by inspection of the *CodeList* (which contains inheritance and length information). Working with the *CodeList* makes the refining a very fast process. This is due to the fact that instead of working with the voxel information, the algorithm deals with a very small list of codes and lengths of the branches. Furthermore, it is possible to control the complexity of the final skeleton by appropriate selection of the single length threshold in condition *1* above. The refinement process is explained below.

5.1 Branch removal

In this step the codes of the qualified branches are removed from the *CodeList* by checking the conditions explained in the previous paragraph. A *threshold* is chosen by inspection of the *CodeList* to determine the length of the spurious-looking branches.

5.2 Re-labeling of the branches

In this step the *CodeList* is re-organized to update parent and child relationships and lengths. That is, starting from the root branch (RP node), the labeling is updated to cancel the effect of removed branches in the labeling process.

5.3 Repeat steps 1 and 2 until no more branches are deleted.

5.4 Trace back

The final *CodeList* corresponds to the remaining branches of the skeleton. The code of these branches might have changed during the pruning and renaming process. Therefore, a correspondence must be established between the original and final

labels of the remaining branches. This is accomplished by tracing back through intermediate stages of pruning and renaming.

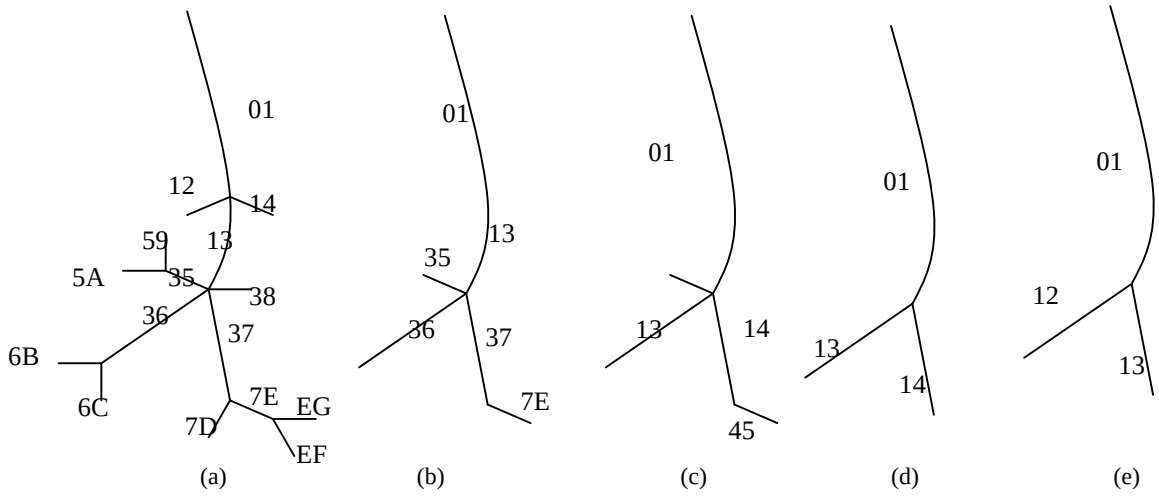


Figure 4. The pruning process. The vessel structure shown in (a) is first refined to give the structure in (b), this structure is then relabeled as shown in (c), this process is repeated until no changes in the structure occur (e).

5.5 Final code structure

The final *CodeList* is formed by storing the final codes and the respective new lengths, which are obtained by summing the length of the original branch segments corresponding to each final code. The code of the remaining voxels is also updated in the set of skeletal voxels. The above procedure is demonstrated in Figure 4.

6. SKELETON CONNECTION

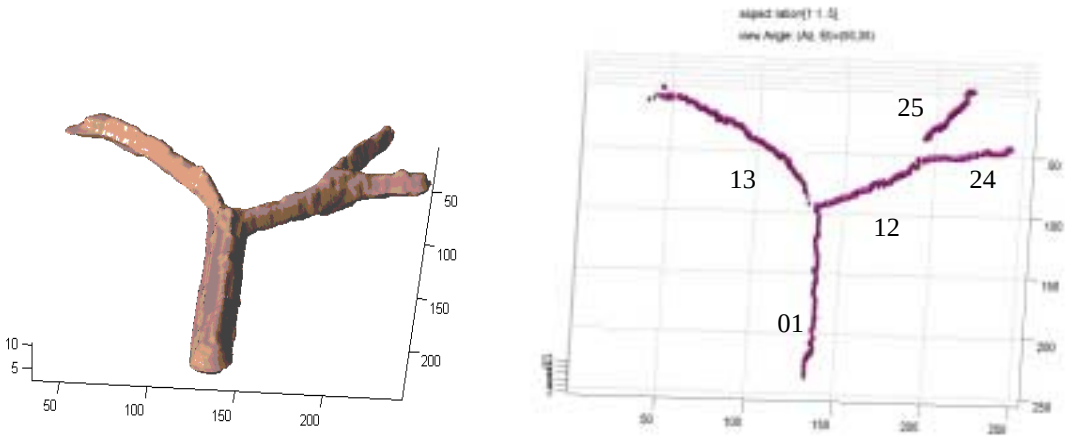
For two consecutive and disconnected skeletal voxels, A and B, the shortest path connecting these voxels is obtained by the SPE procedure. Then the medial points of the local SS-field clusters along this path are found and added to the skeletal voxels to create a connected path.[1] The branch label associated with these new voxels is the same as the code associated with B. This is to ensure correct labeling specially at bifurcations.

7. RESULTS

We have tested and evaluated the method described in this paper, using both simulated and actual images of complicated 3-D elongated shapes. Results showed accuracy of the method in extracting the object medial axis. Furthermore, the algorithm showed strong resistance against noise and boundary distortion because it is not based on local templates and takes into account global structure of the object. In addition, the BS-field can be used to visualize the object using the isosurface rendering techniques. Figures 5-6 show 3-D view of the elongated objects and their skeletons. The analytical branch information corresponding to Figure 5 is listed in table 1.

Table 1. Features of the branches of the object shown in Figure 5.

Branch Code (left is the decimal and right is the binary code)		Branch Length (in voxels)
01	00000000 00000001	157
12	00000001 00000010	101
13	00000001 00000011	166
24	00000010 00000100	69
25	00000010 00000101	44



(a) (b)

Figure 5. (a) A simulated vessel and (b) its skeleton. Note branch labels are shown in Decimal base (The right digit is the head and the left digit is the tail label of each segment.

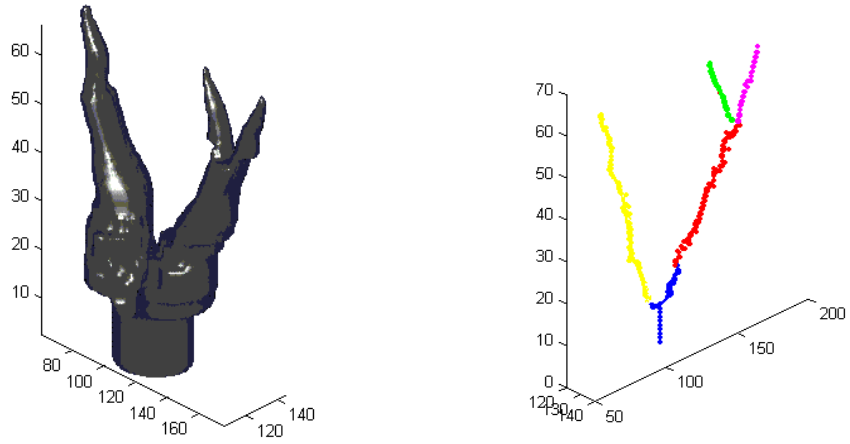


Figure 6. Another simulated 3-D object and its skeleton, different branches are marked by different colors.

8. CONCLUSION

We have developed a novel method to extract the medial axes of 3-D binary elongated objects and refine them. The proposed method successfully extracts skeletons. The medial axis is centered with respect to the object boundary and conveys both major and parameter-controlled branches of the object. The skeletons of different branches of the object are independently accessible for further analysis and processing. The method is robust in the sense of insensitivity to noise. Moreover, it is implemented in an understandable and structured way and runs fast. The proposed method can be used for other applications such as image compression and morphological feature extraction.

ACKNOWLEDGMENTS

This work was supported in part by the School of Intelligent Systems, Institute for Studies in Physics and Mathematics, Tehran, Iran.

REFERENCES

1. Renato Kresch, David Malah, "Skeleton-Based Morphological Coding of Binary Images," *IEEE Transactions on Image Processing*, vol. 7, no. 10, October 1998.
2. D.J. Sheehy et. al., "Shape Description by Medial Surface Construction," *IEEE Trans. Visual. And Computer Graph*. vol. 2, no. 1, Mar. 1996.
3. Computation of skeleton by partial differential equation, Denis Pasquignon, *Proc. Image Processing* 1995, pp. 239-241.
4. One-pixel width image skeleton generation using mathematical morphology, C.K. Lee, Y.W. Pang, *Proc. workshop on Nonlinear Digital Signal Processing* 1993, pp. 6.1_7.1-6.1_7.5.
5. Zana, F.; Klein, J.-C., "Robust segmentation of vessels from retinal angiography," *Digital Signal Processing Proceedings*, 1997. DSP 97., 1997 13th International Conference on Volume:2, Page(s): 1087 -1090 vol.2.
6. C. Min Ma and Milan Sonka, "A Fully Parallel 3D Thinning Algorithm and Its Applications," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 420–433, Nov. 1996.
7. Y. Zhou and Arthur W. Toga, "Efficient Skeletonization of Volumetric Objects," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, 1999, pp.196-209.
8. N. Gagvani and D. Silver, "Parameter-Controlled Volume Thinning," *Graphical Models and Image Processing*, vol. 61, no. 3, pp. 149-164, May 1999.
9. G. Borgefors: On Digital Distance Transforms in Three Dimensions, *Computer Vision and Image Understanding*, vol. 64, no. 3, November 1996, pp. 368-376 .
10. Olivier Cuisenaire, "Distance Transformations: Fast Algorithms and Applications to Medical Image Processing," Ph.D. thesis, Oct. 1999, Communications and Remote Sensing Laboratory, Université Catholique de Louvain, Belgium.
11. Sanniti di Baja, G., Thiel, E., Skeletonization Algorithm Running On Path-Based Distance Maps, *IVC* (14), 1996, pp.47-57.
12. Kensaku, Mari, et. al. "Automated Anatomical Labeling of the Bronchial Branch and its Application to the Virtual Bronchoscopy System," *IEEE Transactions on Medical Imaging*, vol. 19, no. 2, Feb 2000.