Skeleton Graph Generation for Feature Shape Description

Freek Reinders, Melvin E.D. Jacobson, and Frits H. Post

Delft University of Technology
email: {k.f.j.reinders, m.e.d.jacobson, f.h.post}@cs.tudelft.nl

Abstract. An essential step in feature extraction is the calculation of attribute sets describing the characteristics of a feature. Often, attribute sets include the position, size, and orientation of the feature. These attributes are very important, but they do not provide a good approximation of the shape of a feature. For better shape description, a more sophisticated method is needed.
This paper describes a method that extracts a binary skeleton of a feature, and transforms it into a graphical representation: the skeleton-graph. This graph represents the original skeleton with controlled precision, and contains the essential topology and geometry of the skeleton. In addition, distance information is used to generate a simplified reconstruction of the original 3D feature shape, which can also be used as an iconic object for visualization.

Keywords: Feature Extraction, Shape Description, Skeleton Attributes, Graph Simplification.

1 Introduction

Feature extraction is an approach to visualization aiming at automatic recognition of important features (structures, objects, or regions) in scientific data sets. Rather than leaving the recognition of the interesting features entirely to the visual inspection by the user, this task is performed automatically. The extracted features are characterized by quantitative descriptions or attributes. The features are directly related to physical entities and phenomena studied, and thus dependent on the application. In the field of computational fluid dynamics, common examples of features are vortices, shock waves, and recirculation zones.

Most feature extraction techniques are based on a classification or segmentation of the data, to identify the parts of the data sets that belong to the features. If the data are defined on the nodes of a grid, a filter can be used that selects data items which satisfy a certain selection criterion. Other techniques for segmentation include region growing, and edge detection. The result of a segmentation of a grid data set is a binary grid in which feature nodes are marked. Adjacent marked grid nodes can then be clustered into coherent regions, and for each region certain attributes are calculated [8].

The quantification by calculating attributes is essential to the process of feature extraction. The attribute sets give a quantitative description of a feature, describing its most important characteristics, such as its position, and size. The attribute sets can be used to evaluate a feature or to compare it with other features, for instance in order

to track features in time-dependent data sets [7]. A frequently used attribute set is the ellipsoid fit around the feature. The ellipsoid provides a good indication of position, orientation, and size, but is a crude (first order) approximation of shape. Sometimes a more accurate description for shape is needed.

Skeletonization, or Medial Axis Transformation (MAT) provides a more sophisticated method to characterize the shape of a feature. The skeleton of an object can be defined as the locus of points that lie at the center relative to the object's boundary. Thus, the skeleton is a thinner version of a 3D object, which still preserves the topology and geometry of the object. Therefore, it is an efficient and compact shape descriptor of features.

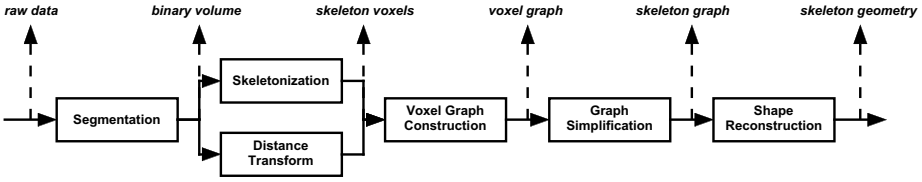


Fig. 1. Pipeline for skeleton graph generation.

This paper presents a method for skeleton attribute calculation of extracted features. Figure 1 illustrates the process of determining for these shape attributes. The input is a regular segmented data volume (a voxel grid), in which binary data represent the voxels belonging to a number of feature objects. The skeleton voxels of the objects are determined by an existing skeletonization method. The resulting voxels are then connected into a voxel graph of skeleton-nodes and edges. The graph initially contains all skeleton voxels, but it can be simplified. First, the topology is extracted by identifying special nodes such as end-nodes, junction-nodes, and loop-nodes. Next, geometry is approximated to a given tolerance by inserting extra curve-nodes and profile-nodes. Third, a reconstruction is made of the 3D shape of the original object.

The combination of skeleton and distance information provides an excellent way to approximately reconstruct the shape of the original object. The skeleton graph is "fleshed out" by wrapping spherical and conical volumes around the edges and nodes of the graph. The size of these volumes is determined by the distance to the surface, which is stored at every skeleton node. The result is a simplified geometric object, which may be used as an iconic representation describing the feature shape and which is superior to the crude approximation by a fitted ellipsoid.

The paper is organized as follows. Sections 2 through 5 describe each step in the process of skeleton graph generation. Section 6 shows a number of applications for this method. Finally, some conclusions and topics for future research are given in section 7.

2 Skeletonization

The first step is the skeletonization of the segmented binary volume. The result is a set of skeleton voxels at the center of the object. Many skeletonization algorithms have been published, especially in the image processing literature. Most algorithms are only for 2D data, but some can be extended to 3D or higher dimensions.

The skeletonization algorithms found in the literature can be classified in two categories:

- **Topological thinning methods.** These methods are based on removing voxels from the surface of the object, by identifying the so-called simple points, i.e. points that will not change the topology of the object when they are removed. Methods (2D and 3D) based on this concept are described in [4] and [6].
- **Distance transform (DT) methods.** The DT can be calculated in each voxel of the object and is equal to the minimal distance to the surface of the object. The skeleton voxels are identified as the local maxima of the DT. Some methods that extract the skeletons based on the DT are described in [5] and [9]. Methods to calculate the DT are described in [1] and [2].

The results of the two types of methods are somewhat different. Topological thinning guarantees the connectivity of the skeleton voxels, while DT-methods in general do not. For our purposes we want a method that guarantees the connectivity.

The algorithm we use is a topological thinning algorithm based on a hit-or-miss evaluation using sets of masks [3]. The mask sets can be used to manipulate a binary object in several ways, of which skeletonization is only one. A mask set consists of a number of 3x3x3 masks with zeroes, ones and "don't cares". Each mask indicates a configuration that must remain in the object. If the 3x3x3 neighborhood of a voxel matches one of the masks, it is a skeleton voxel and should remain in the object, otherwise it is removed. Thus, the object surface is peeled off iteratively until only the skeleton voxels remain.

The characteristics of a skeleton obtained in this way depends on the mask set used. We have three different mask sets to produce different types of skeletons: surface skeletons, line skeletons and point skeletons (see Figure 2). Here, we will only concentrate on the line skeletons. At each skeleton voxel also the distance information is stored, indicating the distance from the voxel to the object surface. We have used a chamfer distance transformation [1, 2], which is calculated together with the skeletonization (see Figure 1).

3 Voxel Graph Construction

After the skeleton voxels have been determined, a *voxel graph* can be constructed by connecting neighboring voxels. In the voxel graph, all skeleton voxels are nodes, and adjacent voxels are connected by edges. The basic approach for constructing the voxel graph is to traverse all voxels of the skeleton and to connect neighboring voxels. The voxel graph is a structure that is easy to manipulate and to analyze. Using the voxel graph, the number of nodes and edges can be reduced, while preserving the basic structure of the skeleton.

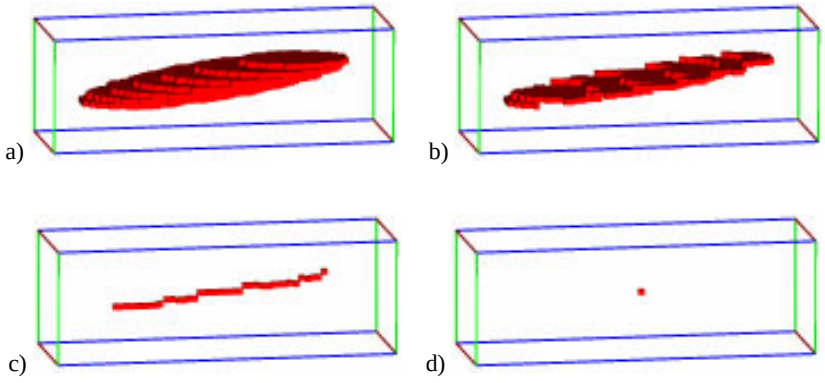


Fig. 2. Skeletonization of a binary volume, resulting in three types of skeletons, depending on the masks used: a) binary volume, b) surface skeleton, c) line skeleton, and d) point skeleton.

The connection of neighboring voxels is not a straightforward process. Simply connecting the adjacent skeleton voxels causes problems as shown in Figure 3a: a "zero-loop" occurs at the junction. All three voxels at the junction have more than two edges and may be classified as a junction-node (see section 4.2), while there is in fact only one junction. Lee, Kashyap and Chu [4], solved this problem by directly classifying the junction node as it is encountered, and classifying all its neighboring nodes as regular. However, the choice of the junction node will depend on the starting point of the traversal, which will result in slightly different graphs, as illustrated in Figure 3b, c, and d.

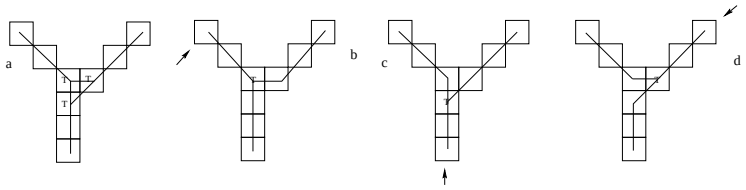


Fig. 3. Problems when constructing the voxel graph: a) zero-loop, b, c, and d) the junction node depends on the traversal starting point.

We solved the problem of zero-loops by assigning a priority ordering to the connections. In a 3D skeleton, each voxel has 26 neighbors: 6 face-connected, 12 edge-connected, and 8 vertex-connected. We give priority to the connection with the nearest neighbor, so face-connected neighbors are connected before edge-connected neighbors and these are connected before the vertex-connected neighbors. In case of a junction, this connection priority will always lead to the graph shown in Figure 3b, which we believe is the best solution.

4 Graph Simplification

After the construction of the voxel graph, the graph can be simplified by removing redundant nodes. The voxel graph initially contains all skeleton voxels as nodes, which may be a large amount. We can reduce the number of nodes and edges while still preserving the topology and geometry of the skeleton. The task is to determine which nodes should be kept and which nodes can be removed.

There are two classes of nodes that are significant and that should be retained: the *topological* nodes and the *geometric* nodes. Topological nodes are nodes that are necessary to preserve the topology, and geometric nodes are necessary to preserve the geometric shape of the object. The identification of topological nodes and their connectivity results in a topological graph, while the detection of geometric nodes results in the geometric graph. The final skeleton graph is a combination of the two graphs.

4.1 Topological Graph

The topological graph holds the basic structure of the object and can be determined by finding the topological nodes. The topological nodes are identified by counting the number of edges connected to a node in the voxel graph:

- *End node*: one edge.
- *Regular node*: two edges.
- *Junction node*: three or more edges.

The end nodes and junction nodes are the nodes that determine the topology of the object. Thus, the topological graph is created by simply removing the regular nodes, and connecting the remaining nodes by edges. However, there is a problem in case of loops.

An additional type of topological node, the *loop node*, is needed to describe loops in the graph. Figure 4 shows a number of situations with loops in the graph. In the case of Figure 4a, the removal of all regular nodes will remove all nodes. To overcome this problem a loop node is included at an arbitrary location on the loop. In the topological graph the loop node is connected to itself with an edge. A similar situation occurs in Figure 4b; the junction node is connected to itself. Therefore, this junction node is classified as a loop node. In the Figures 4c and 4d, also a loop exists, but no node is connected with itself, therefore all nodes in the loop remain junction nodes.

The number of loops is an important variable in the topological graph, because it may provide a cue for comparison of two skeletons. It can be calculated with the *Euler formula*:

$$V - E + F = C - H \tag{1}$$

with V the number of nodes, E the number of edges and F the number of faces. $C - H$ form the so-called *Euler number*, with H the number of holes in an object and C the number of connected objects in a scene. Because we only work with lines F can be set to 0, and because the graph is always a single object, C can be set to 1. Substitution gives the number of loops (holes) in the graph:

$$H = 1 - V + E \tag{2}$$

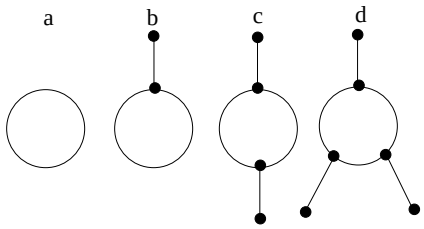


Fig. 4. Loops in the topology of the skeleton.

4.2 Geometric Graph

The geometric graph refines the edges in the topological graph by determining certain geometric nodes. The topological graph is not a sufficient approximation of the shape, especially if the shape is strongly curved. Geometric nodes are key points for describing the shape of the skeleton. The geometry can vary in two ways: the skeleton line is curved, or the profile of the object surface is curved. Hence, we distinguish the following two types of geometric nodes:

- *Curve nodes*: where the skeleton line bends.
- *Profile nodes*: where the surface profile changes.

The geometric nodes are inserted on the edges of the topological graph. For each edge the regular nodes are traversed, and curve or profile nodes are inserted. Insertion of geometric nodes depends on geometric tests, which will be described below. If a new node is inserted, the two new edges are handled recursively in the same way until all geometric tests are satisfied.

Curve nodes

The test for finding the curve nodes uses the maximum distance d_{max} between the voxel nodes and the edge, and compares d_{max} to a distance threshold T_{dist} (see Figure 5). A curve node is added at the location of the maximum when $d_{max} > T_{dist}$. Two new edges are created and both are tested recursively. The process terminates when all intermediate nodes fall within T_{dist} of the corresponding edges.

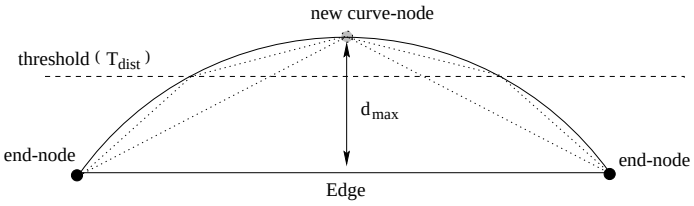


Fig. 5. Finding the curve nodes recursively.

The distance threshold provides a measure of precision for the approximation of the original voxel graph. A zero threshold results in a geometric graph that is equal to the

original voxel graph. When the threshold is very large, no curve nodes are inserted and the topological graph is not refined.

Profile nodes

The test for finding the profile nodes uses the distance transform (DT) in a similar way as the test for finding the curve nodes. The DT is known at each voxel node and can be highly variable. If a linear profile is assumed along an edge, the actual values of the corresponding voxels will differ from this assumption. The profile test finds the voxel with the maximum distance to this linear profile and tests this distance to a threshold T_{prof} (see Figure 6). A new profile node is inserted when the threshold is exceeded and the two resulting edges are tested recursively.

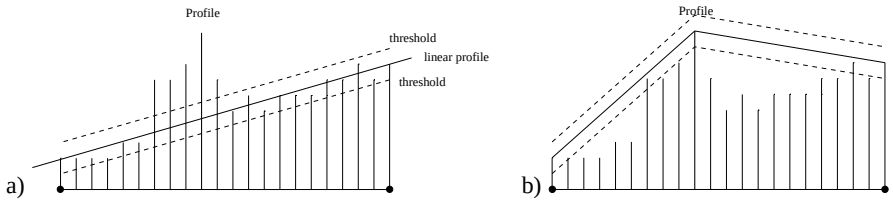


Fig. 6. Finding the profile nodes recursively: a) first step, and b) second step.

Again, the threshold is a measure for shape approximation of the original voxel graph, but this time for the profile of the surface of the object. Also, with a zero threshold the geometric graph is equal to the voxel graph, and with a very large threshold the topological graph remains unchanged.

Figure 7 shows the skeleton graph which results after graph simplification. In the figure the original segmentation is visualized with a transparent iso-surface, and the skeleton graph is visualized with spheres connected by lines. The spheres are located at the node positions and have a radius equal to the DT. In general the skeleton graph gives a good approximation of the shape. However, some objects are cut by the system boundary which results in a flat shape, in these cases the skeleton graph gives a less accurate approximation. The cross-section of a flat object has an elliptical contour, while we assume a circular contour with a radius equal to the distance transform.

5 Shape Reconstruction

The shape of the original object can be approximated by adding a volume to every edge in the skeleton graph. The resulting 3D shapes can be used for an iconic visualization of the features [8]. Several types of geometric objects can be used for "fleshing out" the skeleton graph. The topology is viewed best with an open geometry with for instance lines and spheres, such as shown in Figure 7. A solid structure of spherical and conical volumes gives a clear visualization of the volume of the objects, see Figure 8a). Some variations and intermediate representations to this are possible, one is shown in Figure 8b).

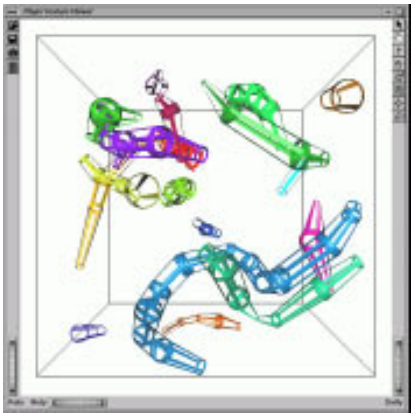
a)  b) 

Fig. 8. Skeleton surface reconstruction using solid spheres and cones, or a more open geometry.

In another approach a cubic Hermite interpolating tube is calculated through each path (from end node to end node) in the geometric graph. The interpolation results in a smooth tube, see Figure 9. The thickness of the tube is taken equal to the distance information at the nodes. This way, a smooth surface is drawn representing the feature objects.

6 Applications

The skeleton generation procedure was applied to a dataset with turbulent vortex structures, obtained from a fluid dynamics simulation[1]. The dataset consists of 128^3 grid nodes with vorticity data. After segmentation, 18 objects were found with a total of 119262 voxels. Skeletonization reduced this number of voxels to 966, and after graph simplification only 87 nodes remained. The file size of the skeleton graph is about 2.3 Kb, while the original data file was 8.0 Mb, a reduction factor in the order of 1000. Figures 7, 8, and 9 show different visualizations of this application.

We also applied the skeleton reconstruction to similar simulations of turbulent vortex structures with a higher Reynolds number. In total three similations were obtained with 128^3 grid nodes with vorticity data. A higher Reynolds number results in more complex and smaller vortex structures, and therefore the skeleton data is somewhat larger. Table 1 shows the reduction in percentages, from segmented volume (V) to voxel nodes (VN), from voxel nodes to geometric nodes(GN) and from volume to geometric nodes, the last column shows the file size of the skeleton file (SF) in Kb (N.B. the file size of the raw data is 8.0 Mb).

[1] Data courtesy D. Silver and X. Wang of Rutgers University.

dataset	V→VN	VN→GN	V→GN	SF (Kb)
Simulation I	74.3%	81.0%	95.1%	12.3
Simulation II	79.7%	76.5%	95.2%	2.7
Simulation III	89.1%	91.1%	99.9%	2.3

Table 1. Reduction percentages.

7 Conclusions and Future Research

In this paper we have presented a method that generates skeleton graphs describing the shape of feature objects in a binary segmented volumes. The skeleton graph is a set of feature attributes that signifies a significant data reduction, while still preserving a good approximation of the original shapes.

The method works in four stages. First, the skeleton voxels of the objects are determined by an existing skeletonization algorithm. Then, neighboring skeleton voxels are connected into a voxel graph representation with nodes and edges. The nodes represent the voxels and the edges represent the connectivity between voxels. Using the distance transform, also the minimal distance to the surface is known in each voxel. Then, the voxel graph is simplified by recognizing topological nodes (junction nodes, end nodes, and loop nodes) and geometric nodes (curve nodes and profile nodes). The simplification can be controlled by two threshold variables, controlling the precision of the skeleton graph approximation. Finally, the skeleton graph is used to reconstruct a geometry that approximates the surface of the segmented objects.

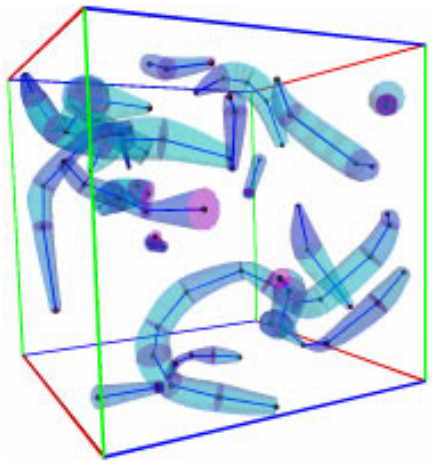


Fig. 9. Skeleton surface reconstruction using hermite tube icons.

The applications with the turbulent vortex structures showed that this method approximates the original surface very well. The skeleton attributes provide a good topological and shape description in a very condensed way. Data reductions in the order of a 1000 were obtained. Figures 7, 8, and 9 show some nice visualizations of this application.

The turbulent vortex structures normally have a worm-like shape which can be very well approximated by this method, however sometimes the method fails. The cross-sections of the worms have a circular contour with a minimal radius equal to the DT. When the shape is more flat the contour has an elliptical shape and the circular approximation can be poor. Two solutions can overcome this problem: 1) for each skeleton voxel create an elliptical approximation, 2) use the surface skeletons (Figure 2b) and do a similar

simplification as described in this paper. The implementation of these two solutions is a topic for future research.

Another topic for future research is using the skeleton graph information for feature comparison and tracking [7] in time-dependent data. The skeleton graph can be collected in successive time steps and stored as feature data. For tracking purposes, a metric needs to be defined for the correspondence between two skeleton graphs.

Acknowledgments

This work is supported by the Netherlands Computer Science Research Foundation (SION), with financial support of the Netherlands Organization for Scientific Research (NWO).

References

1. G. Borgefors. Distance Transformations in Arbitrary Dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321–345, 1984.
2. P. E. Danielsson. Euclidean Distance Mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.
3. P. P. Jonker and A. M. Vossepoel. On Skeletonization Algorithms for 2, 3,.. N Dimensional Images. In D. Dori and A. Bruckstein, editors, *Proc. Shape, Structure and Pattern Recognition '94*, pages 71–80, Nahariya, Israel, Oct. 4-6 1995. World Scientific Singapore.
4. T. C. Lee, R. L. Kashyap, and C. N. Chu. Building Skeleton Models via 3-D Medial Surface/Axis Thinning Algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, November 1994.
5. F. Leymarie and M. D. Levine. Simulating the Grassfire Transform using an Active Contour Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):56–75, January 1992.
6. S. Logbregt, P. W. Verbeek, and F. C. A. Groen. Three-Dimensional Skeletonzation: Principle and Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:75–77, January 1980.
7. F. Reinders, F.H. Post, and H.J.W. Spoelder. Attribute-Based Feature Tracking. In E. Gröller, H. Löffelmann, and W. Ribarsky, editors, *Data Visualization '99*, pages 63–72. Springer Verlag, 1999.
8. T. van Walsum, F.H. Post, D. Silver, and F.J. Post. Feature Extraction and Iconic Visualization. *Trans. on Visualization and Computer Graphics*, 2(2):111–119, 1996.
9. Y. Xia. Skeletonization via Realization of the Fire Front's Propagation and Extinction in Digital Binary Shapes. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(10):1076–1086, October 1989.