The Visual Computer

Extracting skeletal curves from 3D scattered data

Anne Verroust[1], Francis Lazarus[2]

[1] INRIA Rocquencourt, Domaine de Voluceau, B.P. 105, 78153 Le Chesnay Cedex, France
[2] I.R.C.O.M.-S.I.C. SP2MI, Boulevard 3, Teleport 2, B.P. 179, 86960 Futuroscope Cedex, France

We introduce a method for extracting skeletal curves from an unorganized collection of scattered data points lying on a surface. These curves may have a treelike structure to capture branching shapes such as blood vessels. The skeletal curves can be used for various applications ranging from surface reconstruction to object recognition.

Key words: Visualization – Skeletal curves – Cylindrical decomposition – Generalized cylinders – Reconstruction

Correspondence to: A. Verroust

1 Introduction

The description of objects with cylindrical shapes has been extensively used in the computer vision community [1, 32, 33, 39]. In a first approximation, it is often convenient to model a shape as a generalized cylinder. This is particularly relevant in the context of reconstruction of anatomical shapes from CT data or a scanner [28, 31, 38]. Numerous works regarding object description use treelike structures [26, 27, 35] or cylindrical representations [10]). What makes the use of cylindrical descriptions and generalized cylinders so popular is their relative simplicity. A generalized cylinder is parameterized by a spin curve together with a set of cross sections [1]. Depending on the application, the recovery of such parameters usually requires less work than a more general description of a shape.

Note that a spin curve can be used as a skeleton of an implicit surface model [7, 11, 13, 30]: the surface of the object is defined as an isosurface of a field function generated by the skeleton. In [12, 17], the authors propose an implicit reconstruction of shapes from a set of 3D scattered points using the Voronoï graph of the data points to build a geometric skeleton. Our method is an alternate skeleton construction. It avoids the computation of the 3D Voronoï graph.

Other applications start with volume images so that the traditional medial axis transform can be applied [28].

As opposed to voxelized data, we start with an unorganized cloud of points sampled on a surface. A 3D curve must be specified inside the cloud of points in order to use a cylindrical model and to represent the surface. When the set of 3D points is given as a set of 2D planar scans, Burdin et al. [10] use the curve defined by the centroids of each 2D section. However, this implies a consistent orientation of the cross-sectional planes relative to the position of the object. Figure 1 shows two orientations of the cross-sectional planes. Clearly, the configuration of Fig. 1a is not adapted. For some objects, because of their bent shape, it is even impossible to find a suitable orientation for the sectional planes.

Although essential, the issue of finding a suitable axis in order to design a generalized cylinder has rarely been discussed. In practical cases, this task can be tedious, as mentioned by Nazarian and colleagues [26]. These authors propose two methods to automate the determination of the axis.

The Visual Computer (2000) 16:15–25

- The first one consists in a recursive subdivision of the set of data points. At each step the subsets of data points are split by a plane perpendicular to their main axes of inertia and passing through their barycenters. The resulting axis is a polyline composed of main axes of parts of the initial set of data points (Fig. 2a). The authors recognize that their method does not always work, and they show a counterexample where the main axis of inertia (Fig. 2b) cannot be used to find the central axis.
- The second method works with a set of data points given by planar 2D scans. It uses the triangulation computed by Boissonnat's method [9] as an approximation of the tubular shape.

In this paper, we present a method that automatically computes a set of 3D curves in a cluster of points. When the points lie on a surface, these curves may be used as axes for a cylindrical approximation of the surface. These curves may form a tree structure if the surface resembles a "tree of generalized cylinders". The method works independently of the bending of the surface. As a consequence, it returns a correct result for the shape shown in Fig. 2b. As for other reconstruction algorithms [2, 5, 24], we set some conditions on the input cluster of points.

1. The data points should be sampled on a certain (unknown) surface.
2. The sampling should be done in such a way that the distance between neighbor points is small with respect to the width of the tubular parts of the surface.

The computation of the axes is based on a decomposition of the surface to be reconstructed into level sets of a scalar function. In [21–23] we apply a similar decomposition to a polyhedral surface. In this work a simplified discrete version of the Morse theory was developed (see [19, 20, 36, 37] for use of Morse theory in graphics).
In the present case, we are dealing with a cluster of points, and no topological information, such as an adjacency relationship, is available. We cannot apply the method of [21–23], which uses this information to build the decomposition of the surface.
Here, our goal is to compute the axes rapidly rather than to reconstruct the surface. We follow the main steps of [21, 22] and adapt them to our particular case. The scalar function used in the decomposition

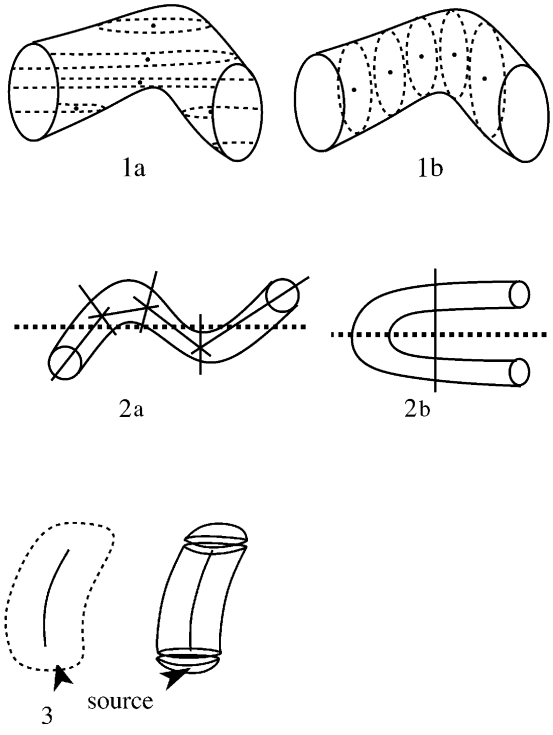


Fig. 1. Two sets of 2D sections sampled on a surface

Fig. 2. a Recursive subdivision using main axes of inertia (Nazarian & al); the first main axis is shown in *dashed line*. The main axes of inertia and the perpendicular splitting planes for the second level of recursion are shown in *plain lines*; **b** The first axis of inertia and the perpendicular splitting plane are not suitable for the subdivision algorithm

Fig. 3. The source point and the corresponding decomposition

is an approximation of a geodesic distance function to a given point. This function is determined by a single point from the set of data points called the *source point*. The source point may be interactively designated by the user. It may also be automatically selected by a simple heuristic as explained at the end of Sect. 3. Intuitively, the source point corresponds to a polar extremity of the shape, as shown in Fig. 3. Once the decomposition into the level sets of a scalar function is done, a single axis is obtained for each cylindrical part associated with it.

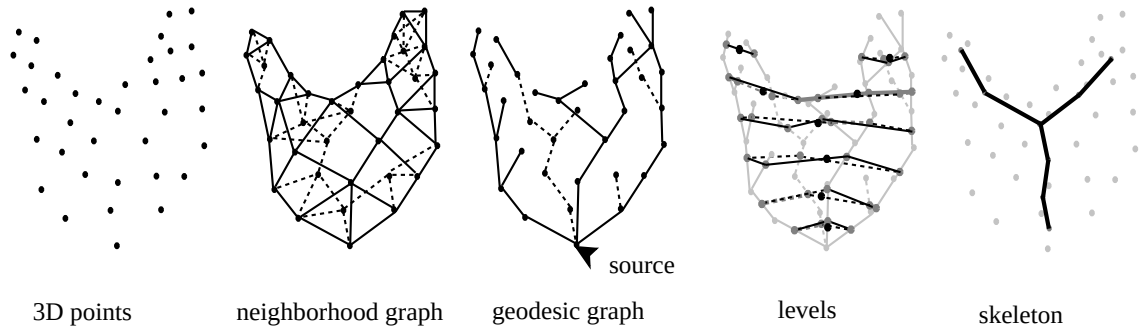


Fig. 4. The main steps of our method

2 Overview

The 3D curves are computed in four steps, as illustrated in Fig. 4. Given a set of data points:

1. We build a *neighborhood graph* of minimum connectivity number n. This is a nonoriented graph whose vertices are the input data points. The edges have the form (v, v'), where the vertex v' or the vertex v is one of the n nearest neighbors of v or v', respectively. The subgraphs of the neighborhood graph induced by a point and its neighbors may be nonplanar and, in general, it will not be possible to fit a suitable polygonal surface to this graph.
2. From this neighborhood graph and a source point, we construct a distance map and an oriented subgraph called the *geodesic graph*. The geodesic graph is obtained by computing shortest paths of the neighborhood graph between all vertices and the source point. The distance map is simply the length of these shortest paths. The construction of the neighborhood graph, the distance map, and the geodesic graph are described in detail in Sect. 3.
3. We compute k level sets of the distance map using the neighborhood graph, where k is a user-defined value. The level sets are composed of points at a constant distance from the source. We connect these points on a proximity criterion and identify the connected components. We also compute the centroids of each of these connected components.
4. The axes are finally obtained by connecting the centroids of successive levels as previously computed. When the levels contain several connected components, the connection of the centroids are based upon the paths of the geodesic graph.

Steps 3 and 4 are discussed in Sect. 4 and examples are provided in Sect. 5.

3 The geodesic graph

We are given a set of data points $X = \{x_1, \ldots, x_m\}$ sampled on an unknown surface $\mathcal{S}$ and an integer n. The neighborhood graph $\mathcal{N}_{(X,n)}$ is an undirected graph defined as follows:

1. The vertices of $\mathcal{N}_{(X,n)}$ are the points of X.
2. A pair (x_i, x_j) is an edge of $\mathcal{N}_{(X,n)}$ if either x_i is one of the n nearest points of x_j in X or x_j is one of the n nearest points of x_i in X.

In order to compute the n nearest points of each vertex, we use a subdivision of the space into voxels. We subdivide the enclosing box of the data points into a regular grid of voxels and associate each data point with the voxel containing it. The n nearest neighbors of a data point x are searched for in two steps. We first visit the voxels close to the one containing x, progressively moving away until n points $(v_1, \ldots, v_n)$ are found. Then, we set $d = \max_{1 \le i \le n}(\text{distance}(v_i, x))$, the maximum distance between x and these n points. As we have already found n points $(v_1 \ldots, v_n)$ located at a distance less or equal to d from x, we are sure that the n nearest neighbors of x belong to the sphere $S(x, d)$ with center x and radius d. We search all the data points belonging to $S(x, d)$, and we extract the n nearest points on this set, using a quick sort on the distance values. The number n should

be chosen so that $\mathcal{N}_{(X,n)}$ is a connected graph. This choice depends on the regularity of the sampling step of the data points on the surface. In the example presented in Fig. 9 we used $n = 5$. When the data points are not uniformly spread over the surface, larger values may be necessary. For instance, in the case of 2D planar scans, where the resolution is smaller inside the scan planes than between the cross-sections, we found that a value of $n = 10$ was necessary to obtain a connected graph. This was the case for the blood vessels shown in Fig. 11.

The selected source point x_s and the neighborhood graph are used to compute the geodesic graph $\mathcal{G}_{(X,n,x_s)}$. The geodesic graph is composed of geodesic paths of the neighborhood graph emanating from the data points and joining the source point. The length of an edge of the neighborhood graph is taken as the euclidean distance between its endpoints. We use Dijkstra's algorithm [15] to efficiently compute these paths and their length. Precisely, Dijkstra's algorithm computes the geodesic distance from any point to the source, as well as a pointer to the next point in a geodesic path joining the source. Note that several geodesic paths from a point to the source may exist. We simply use the path returned by Dijkstra's algorithm. The distance map introduced in Sect. 2 is defined for every data point to be its geodesic distance to the source. It is extended to the interior of the edges of the geodesic graph with a linear interpolation of the endpoint distance values of the edges.

As we supposed that the edge of the neighborhood graph should lie on the surface, the geodesic paths previously computed are intended to approximate geodesic paths on the surface. Note that the computation of geodesic paths for polyhedral surfaces have been the subject of numerous publications [25, 29]. However, we cannot apply those methods, since our data points do not have a polyhedral structure.

Note that $\mathcal{G}_{(X,n,x_s)}$ is a connected subgraph of $\mathcal{N}_{(X,n)}$ since all the geodesic paths share the source point. $\mathcal{G}_{(X,n,x_s)}$ is actually a tree since Dijkstra's algorithm selects only one path from any vertex to the source. An example of a geodesic graph is shown Fig. 6 for a set of data points registered on a femur (data courtesy of J. Menon, IBM T. J. Watson Research Center).

The source point x_s may be selected by the user or automatically computed by the following two-step heuristic:

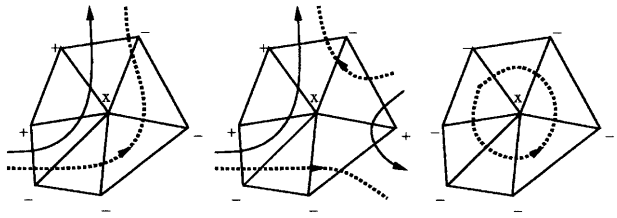


Fig. 5. The leftmost vertex is regular, and the other are critical (the + (respectively −) associated to a vertex indicates that the geodesic distance between the vertex and the source is greater (respectively lower) than the geodesic distance between the source and x)

1. A point x_0 is chosen at random (e.g. the first point in the data set).
2. x_s is defined as the farthest point from x_0 on the neighborhood graph.

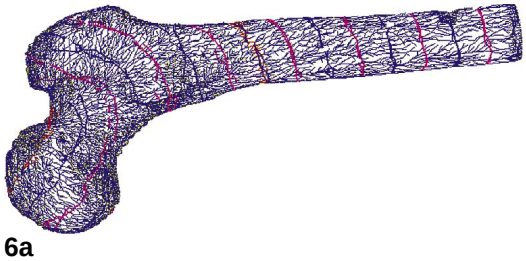
When the object has a "tubular structure", even with branches, the farthest point from x_0 is a point located at the extremity of a tubular part of the surface. This point can be obtained as the last visited vertex when Dijkstra's algorithm runs a first time with source point x_0.

4 Levels and branches

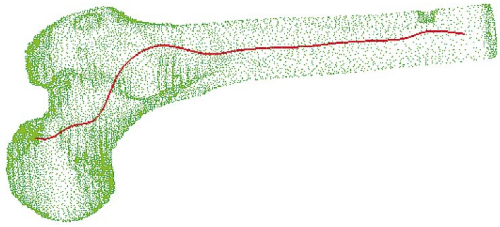
After computing the geodesic graph and the distance map, we extract k level sets. Intuitively, each level set corresponds to a set of "sections" of the decomposition of the surface given by the set of data points $X = \{x_1, \ldots, x_m\}$. The number k is given by the user to specify the number of points used to approximate the longest axis. In practice, k can be chosen as the ratio of the distance of the farthest point to the average edge length. The k levels $d_1, \ldots, d_k$ are uniformly distributed over the interval $[\alpha d_{\max}, \beta d_{\max}]$, where $d_{\max}$ is the extremum of the distance map and α and β are two percentage coefficients. We used $\alpha = 3\%$ and $\beta = 97\%$ in Fig. 12. The ith level set $\mathcal{S}_i$ is the set of points on the neighborhood graph whose (extended to edge interiors) distance is d_i.

Next, we need to partition each level set $\mathcal{S}_i$ into subsets corresponding to the different branches of the surface.

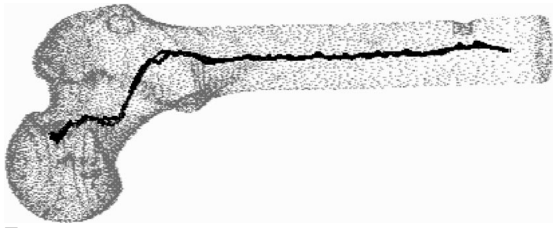
Here, we cannot proceed as in [21, 22] where each vertex of the polyhedron can be classified as regular or critical according to the configuration of distances of its neighbors (see Fig. 5). In the polyhedral case, the decomposition is obtained by cutting through the



6a



6b



7

Fig. 6. a 15 level sets computed on the geodesic graph of a femur; **b** The corresponding axis

Fig. 7. Different number of levels used to compute the axis of the femur : 30 (*grey curve*) and 90 (*black curve*)

level sets associated with the critical points. Moreover, the adjacency between faces of the polyhedron can be used to extract the level sets of the geodesic distance efficiently.

In our case this adjacency relationship is not available. As a result, points cannot be classified as regular or critical, which would in turn provide the branching points and the partition of each level set corresponding to the various branches. To obtain this partition, we first construct the neighborhood graph $\mathcal{N}_{(\mathcal{S}_i,2)}$, linking every point of $\mathcal{S}_i$ with its two closest neighbors. Then, we remove from $\mathcal{N}_{(\mathcal{S}_i,2)}$ the edges longer than a certain length w_i. We determine w_i using condition 2 on the data points: the distance between points is small with respect to the width of the tubular parts. In practice, we use the median of the distribution of the distance values of the second nearest neighbor of each point inside $\mathcal{S}_i$ to compute w_i. Finally, we partition this graph into its connected components. We also compute the centroid of each connected component by taking the barycenter of its points.

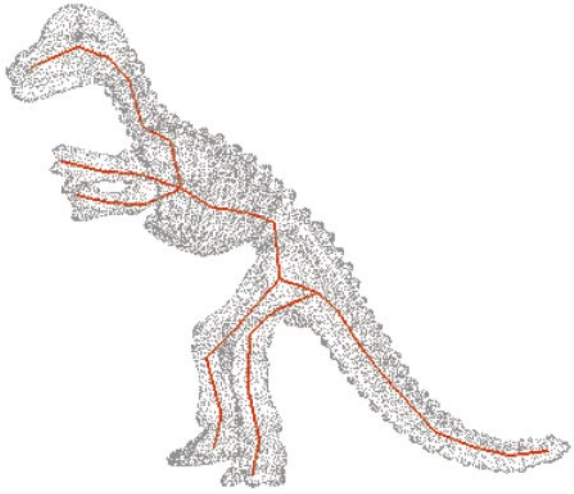
The skeletal curves are constructed by connecting the centroids of successive level sets. When the level sets have several connected components, we use the geodesic paths to connect the centroids. If a level set $\mathcal{S}_i$ is partitioned into l_i connected components $(\mathcal{S}_{i,1}, \ldots, \mathcal{S}_{i,l_i})$ or if the previous level set $\mathcal{S}_{i-1}$ was partitioned into l_{i-1} connected components, then for each $j \leq l_i$, we select a point of $\mathcal{S}_{i,j}$ and follow its geodesic path to the source point until an edge containing a point of $\mathcal{S}_{i-1}$ is encountered. This point belongs to the connected component of $\mathcal{S}_{i-1}$, which is either associated with the branch containing $\mathcal{S}_{i,j}$ or located at the birth of this branch.

Eventually, we may approximate the axes with spline curves as shown for the femur in Figs. 6 and 7. We use a simple fitting procedure from [16, 34] to compute these splines from the connected centroids.

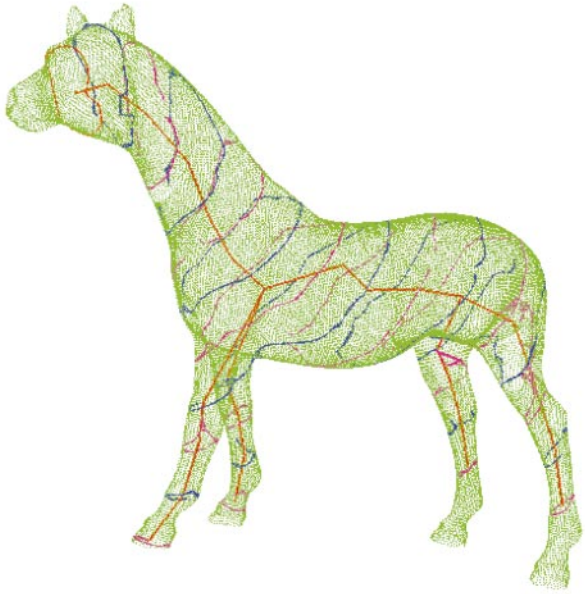
5 Results

We applied our skeletonization method on the set of examples presented in Figs. 8 – 12:

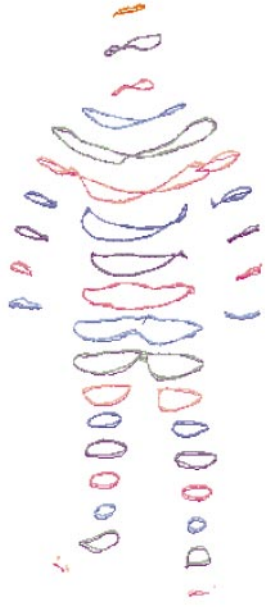
- Figure 8 shows two models of a dinosaur and a horse. These two models and the man in Fig. 9 are public domain models that can be found on the Web pages of Cyberware (http://www.cyberware.com/models/index.html). The source point of the dinosaur is located at the end of its tail.



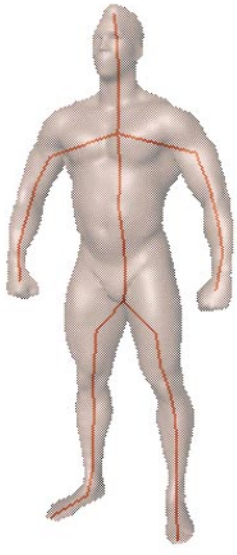
8a

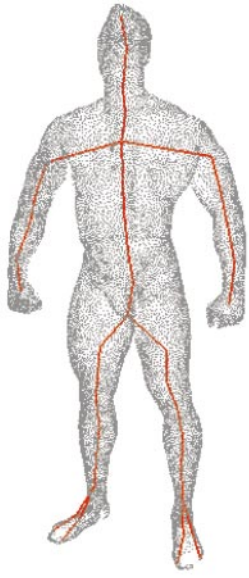


8b



9a

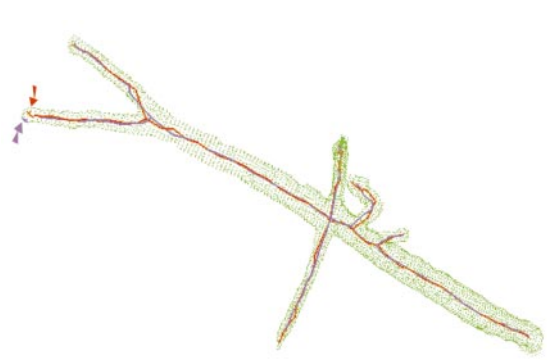




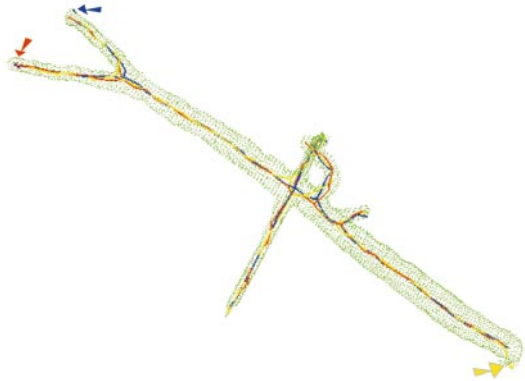
9b

Fig. 8. a The skeletal curves of the dinosaur are built from 20 level sets; **b** 15 level sets and the corresponding skeletal curves on a horse

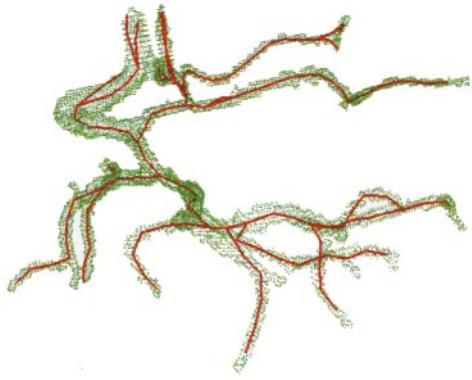
Fig. 9. a 18 level sets and the corresponding skeletal curves on a man described by a set of scattered data points or by a polyhedron (*the darkest lines*); **b** The curves and their associated object representation



10a



10b



11

Fig. 10. a Two skeletal curves associated to two neighbor points; **b** Three skeletal curves associated to three different source points
Fig. 11. Two ramifications of blood vessels

The horse was scanned from many orientations, resulting in 210 linear scans consisting of a series of scans of various regions (views) of the object. The source point is located at the extremity of its jaw.
Here, the branches are correctly located, and a treelike axial structure is built.

- For the man in Fig. 9, we tried two versions of our skeletonization method, as we had the polyhedron associated to the surface: the version described in [23], which works on polyhedra and the version described in this paper with the same source point located on the top of the man's head and, as a set of data points, the vertices of the polyhedral description. One can see that the two sets of contours are very similar, so that the use of the neighborhood graph instead of the polyhedral description of the surface is sufficient to compute the level sets.
The main difference is in the branches and is due to the irregular distribution of the vertices on the man's feet. The branches corresponding to the arms and the legs are nearly the same, and, as there is a lack of vertices on the top of the feet, two different connected components are built and a branching node is created at each foot.

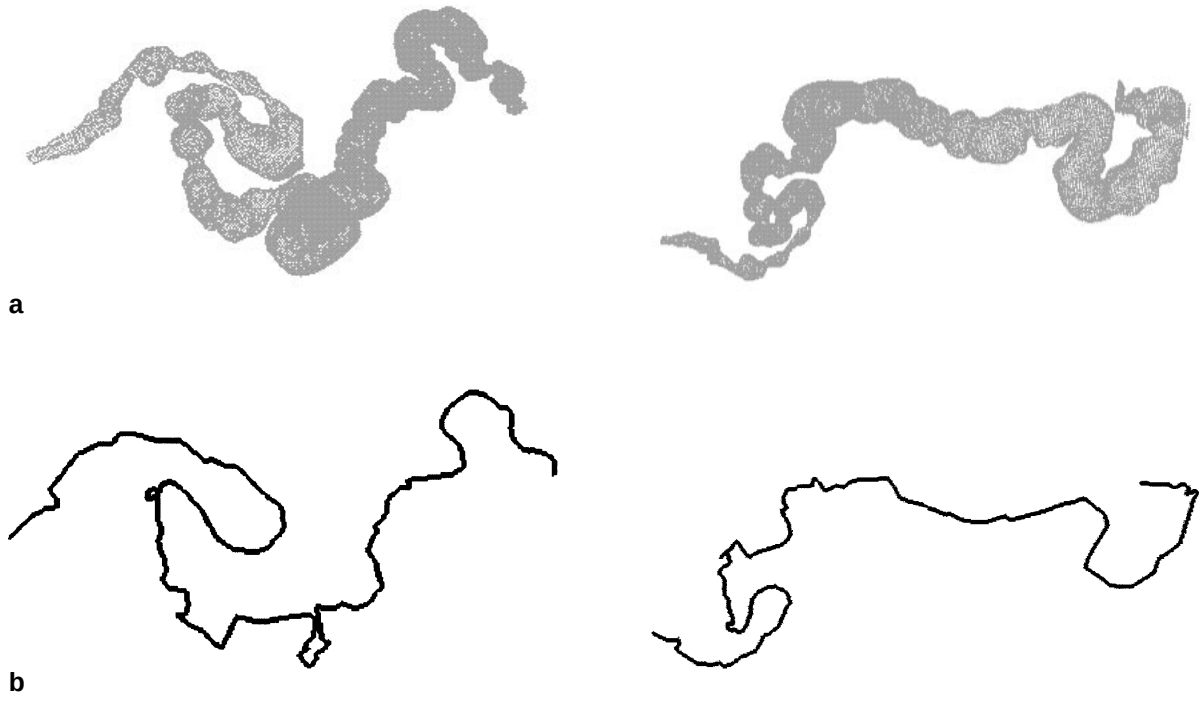


Fig. 12. a A colon. **b** Its axis (two different views)

This example confirms the validity of our approach when the data points are regularly distributed on the surface of the object.

- The aorta presented in Fig. 10 is described by a set of 2D scans. These data come from the National Library of Medicine Visible Human Male (http://www.mayo.edu/bir/Models/Visible_Human_Male/VRML/index.html). Our method is hardly sensitive to the location of the source points: in Fig. 10a, two neighbor points and in Fig. 10b, three points located at different extremities of the tubular parts of the surface have been used to compute the axial structures. The difference between their corresponding axial structures is negligible.
- We obtained good results with our method for the two blood vessels of Fig. 11 (data courtesy of J. Feldmar). These data sets are especially challenging since they contain many vessels close to each other.
- The axis of the colon presented in Fig. 12 cannot be found with Nazarian's subdivision method [26]. It is described by a set of 2D scans and comes from the National Library of Medicine Visible Human Male.

Table 1 shows the CPU times required to compute the axial structure for the models shown in Fig. 6 to 12. We ran nonoptimized code on an Indy. The computing time depends mainly on the number of data points and on the minimum connectivity number of the neighborhood graph. In fact, the computation of the neighborhood graph is one of the most expensive parts of our method. This is confirmed by the CPU times measured on the two versions of our algorithm for Fig. 9:

– When we have a polyhedral description of the surface (cf. [23]), the computation of the axial structure is done in 10.45 s.
– When we have only a set of data points scattered on the surface, we compute several neighborhood graphs: the first one is for the whole set of data points and the other are computed at each level, to partition the level set into con-

Table 1. Statistics for Fig. 6 to 12

Figure	Number of data points	Number of levels	Minimum conectivity number of the neighborhood graph	CPU time in seconds on an Indy
6	16 781	15	5	27.83
7	16 781	30	5	33.20
7	16 781	90	5	44.83
8	21 500	15	5	52.21
8 (horse)	48 486	15	6	132.61
8	14 071	20	5	54.90
9	21 500	18	5	62.21
9 (polyhedron)	21 500	18	-	10.45
10a,b	7688	50	5	40.01
10a	7688	50	5	39.54
10b(middle)	7688	50	5	35.09
10b(right)	7688	50	5	35.91
11a	11 807	60	10	28.85
11b	13 582	50	8	45.49
12	55 573	150	5	235.56

nected components. In this case the computation is done in 62.21 s.

To improve this computation time, we need to speed up the computation of the k nearest neighbors (with, for instance, the method presented in [14]).

6 Conclusion

We have presented a new technique to associate an axial structure with a set of scattered data points. The interaction is reduced to the selection of a source point and can be automated with a simple heuristic. The main steps for the construction of the axes are:

1. The computation of the neighborhood and geodesic graphs, using Dijkstra's algorithm
2. The computation of geodesic levels and the location of branching nodes.

Our method is rapid and robust with respect to the location of the source point. The "quality" of the resulting skeletal curves mainly depends on the appropriate choice of the minimum connectivity number of the neighborhood graph.
Our skeletonization algorithm relies on the location of the source point. This means that the axial structure found by the algorithm is not unique for a given object. The axial structure is actually defined for a pair (object, source point).

The question that arises from this observation is how can we extract interesting axes for objects with tubular shapes? We should note that the precise definition of a tubular shape is not clear (except for the trivial case of a curve extrusion along an axis). As a consequence, there is no precise definition for the axis of a tubular shape. The medial axis, for instance, does not always produce what we would expect from a tubular object. As opposed, to our skeletons, the medial axis is in general composed of pieces of curves, as well as pieces of surfaces with many ramifications [8, 18]. It can be simplified to obtain a wire-frame skeleton, fixing some thresholds [3, 4] or by an automatic process [6]. In both cases, the whole process may also be expensive as the simplification process works on the medial axis of the object represented by the set of sampled data points.
There are many objects, however, for which the definition of an axial structure is intuitively so obvious that we could draw this axis without any hesitation. Blood vessels are good examples of such objects. Our initial question is now turned into the more pragmatic question: how does this skeletonization algorithm provide an efficient way for computing axes in the structure of blood vessels? The only point we have to worry about is selecting the source point, since the computation is entirely automatic, once this is done. On one hand, we see in Fig. 10a that the algorithm is fairly insensitive

to the precise location of the source point. This is due to the continuity of the distance function with respect to the source location. On the other hand, we see in Sect. 3 that the source point can be selected with a simple heuristic. Figure 10b shows that this heuristic works well in practice: extremities on different branches give rise to a nearly identical axis.

The work in progress include attempts to model more precisely the location of the branching nodes. A more robust construction of the neighborhood graph using an adaptive number of neighbors based on the local density of data points is also under consideration.

References

1. Agin GI, Binford TO (1976) Computer description of curved objects. IEEE Trans Comput C-25(4):439–449
2. Amenta N, Bern M, Kamvysselis M (1998) A new Voronoi-based surface reconstruction algorithm. (SIGGRAPH ’98), Comput Graph 32: 415–421
3. Attali D, Montanvert A (1997) Computing and simplifying 2D and 3D semicontinuous skeletons of 2D and 3D shapes. Comput Vision Image Understanding 67: 261–273
4. Attali D (1995) Squelettes et graphes de Voronoi 2D et 3D. Thèse de doctorat en sciences, Université Joseph Fourier Grenoble, France
5. Attali D (1997) R-regular shape reconstruction from unorganized points. ACM Symposium on Computational Geometry, Nice, France, ACM Press, New York, pp 248–253
6. Bloomenthal J, Lin C (1999) Skeletal methods of shape manipulation. Shape Modeling International’99, Aizu-Wakamastu, Japan, IEEE Computer Society, Los Alamitos, CA, pp 44–47
7. Bloomenthal J (1995) Skeletal Design of Natural Forms. PhD Thesis, University of Calgary, Calary, Canada
8. Blum H (1973) Biological shape and visual science (part I). Theor Biol 38: 205–287
9. Boissonnat J-D (1988) Shape reconstruction from planar cross sections. Comput Vision Graph Image Processing 44:1–29
10. Burdin V, Roux C, Lefèvre C, Stindel E (1996) Modeling and analysis of 3-D elongated shapes with application to long bone morphometry. IEEE Trans Med Imaging 15: 79–91
11. Bloomenthal J, Shoemake K (1991) Convolution surfaces. (SIGGRAPH ’91) Comput Graph 25:251–256
12. Bittar E, Tsingos N, Gascuel MP (1995) Automatic reconstruction of unstructured 3D data: combining a medial axis and implicit surfaces. (Eurographics ’95) Comput Graph Forum 14(3):457–468
13. Bloomenthal J, Wyvill B (1990) Interactive techniques for implicit modeling. (1990 Symposium on Interactive 3D Graphics) Comput Graph 24:109–116
14. Callahan PB, Kosaraju SR (1992) A decomposition of multi-dimensional point-sets with applications to *k*-nearest-neighbors and *n*-body potential fields. STOC ’92, Proceedings of the 24th Annual ACM Symposium on the Theory of Computing, Victoria, Canada, ACM Press, New York, pp 546–556
15. Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to Algorithms. McGraw-Hill, New York, NY
16. Eilers PH (1994) Smoothing and interpolation with finite differences. Graphics Gems IV, Academic Press, Boston, pp 241–250
17. Ferley E, Cani-Gascuel M-P, Attali D (1996) Skeletal reconstruction of branching shapes. Implicit Surfaces’96, pages 127–142, Eindhoven, October 1996 (also in Comput Graph Forum 16(5):283–293)
18. Goldak JA, Yu X, Knight A, Dong L (1991) Constructing discrete medial axis of 3D objects. Int J Comput Geom Appl 1:327–339
19. Hart JC (1998) Morse theory for implicit surface modeling. In: (ed) Mathematical Visualization, Springer, Heidelberg, pp 257–268 (Reviewed paper presented at VisMath ’97, Berlin, 1997)
20. Kergosien YL (1992) Topology and visualization: from generic singularities to combinatorial shape modelling. In: Shinagawa Y, Kunii TL (eds) International workshop on modern geometric computing for visualization, Tokyo, Japan, pp 31–54
21. Lazarus F (1995) Courbes, cylindres et métamorphoses pour l’image de synthèse. Thèse de doctorat en sciences, Université PARIS VII, Orsay, France, http://www-sic.univ-poitiers.fr/ lazarus/publications.html#these
22. Lazarus F, Verroust A (1996) Décomposition cylindrique de polyèdre et courbe squelette. Revue Internationale de CFAO et d’Infographie 11:1–10
23. Lazarus F, Verroust A (1999) Level set diagrams of polyhedral objects. SMA’99 The 5th ACM Symposium on Solid Modeling and Applications, Ann Arbor, MI, ACM Press, New York, pp 130–140
24. Mencl R, Müller H (1998) Graph-based surface reconstruction using structures in scattered points sets. Computer Graphics International’98, Hannover, Germany, IEEE Computer Society Press, Los Alamitos, CA, pp 298–311
25. Mitchell JSB, Mount DM, Papadimitriou CH (1987) The discrete geodesic problem. SIAM J Comput 16:647–668
26. Nazarian B, Chédot C, Sequeira J (1994) Interactivity and Delaunay triangulation for the reconstruction of tubular anatomical structures. The 16th annual conference of IEEE/Engineering in Medicine and Biology Society Baltimore, IEEE Computer Society Press, Los Alamitos, CA, pp 125–132
27. Nazarian B, Chédot C, Sequeira J (1996) Automatic reconstruction of irregular tubular structures using generalized cylinders. MICAD’96 – Revue Internationale de CFAO et d’Infographie 11:11–20
28. Näf M, Kübler O, Kikinis R, Shenton ME, Székely E (1996) Characterisation of 3D organ shapes in medical image analysis using skeletonization. Workshop on Mathematical Methods on Biomedical Image Analysis. IEEE Patt And Machine Intell, San Francisco, CA, IEEE Computer Society Press, Los Alamitos, pp 139–150
29. Opitz K, Pottmann H (1994) Computing shortest paths on polyhedra: applications in geometric modeling and scientific visualization. Int J Comput Geom Appl 4:165–178

30. Pasko A, Savchenko V (1997) Projection operation for multidimentional geometric modeling with real functions. In: Klein R, Strasser W, Rau R (eds) Geometric modeling: theory and practice the state of the Art, Springer, Berlin Heidelberg New York, pp 197–205
31. Pisupati C, Wolf L, Mitzner W, Zerhouni E (1995) A central axis algorithm for 3D bronchial tree structure. International Symposium on Computer Vision, Coral Gables, Fla. IEEE Computer Society Press, Los Alamitos, CA, pp 259–264
32. Soroka B, Andersson R, Bajcsy R (1981) Generalized cylinders from local aggregation of sections. Patt Recogn 13:353–363
33. Shani U, Ballard DH (1984) Splines as embedding for generalized cylinders. Comput Vision Graph Image Processing 27:129–156
34. Schneider PJ (1990) An algorithm for automatically fitting digitized curves. In: Glassner A (ed) Graphics Gems 1. Academic Press, San Diego London, pp 612–620
35. Sequeira J, Ebel R, Schmitt F (1993) Three-dimensional modeling of tree-like anatomical structures. Comput Med Imaging Graph 17:333–337
36. Shinagawa Y, Kunii TL, Kergosien YL (1991) Surface coding based on Morse theory. IEEE Computer Graphics and Applications 11:66–78
37. Takahashi S, Ikeda T, Shinagawa Y, Kunii TL, Ueda M (1995) Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. (Eurographics '95) Comput Graph Forum 14(3):181–192
38. Williams JP, Wolff LB (1995) Form from function: a vector field based approach to the analysis of CT images of the vascular tree. Workshop on Physic-based Modeling in Computer Vision, Cambridge, Mass. IEEE Computer Society Press, Los Alamitos, CA, pp 70–77
39. Zerroug M, Nevatia R (1996) Volumetric descriptions from a single intensity image. Int J Comput Vision 20:11–42

ANNE VERROUST is a Senior Researcher at INRIA Rocquencourt, France. She obtained her Thèse de 3^e cycle in Computer Science and her Thèse d'Etat in Computer Science from the University of Paris-sud, France, in 1983 and 1990. Her research interests include geometric modeling, animation, computer, graphics, and computer aided design.

FRANCIS LAZARUS is an Assistant Professor of Computer Science at the University of Poitiers, France. He received his PhD in Computer Science in 1995 from the University of Paris VII, France. He was a postdoctoral researcher at the IBM T.J. Watson Research Center, Yorktown Heights, NY, from 1996 to 1997, and he coauthored the IBM VRML 2.0 binary standard proposal. His research interests include geometric modeling, geometry compression, computer animation, and three-dimensional morphing.