Computing Hierarchical Curve-Skeletons of 3D Objects

NICU D. CORNEA, DEBORAH SILVER, XIAOSONG YUAN, AND RAMAN BALASUBRAMANIAN

Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ 08855, USA

E-mail: cornea, silver, xiaosong, balaiitm @ece.rutgers.edu

Abstract:

A curve-skeleton of a 3D object is a stick-like figure or centerline representation of that object. It is used for diverse applications, including virtual colonoscopy and animation. In this paper we introduce the concept of hierarchical curve-skeleton and describe a general and robust methodology which computes a family of increasingly detailed curve-skeletons. The algorithm is based upon computing a repulsive force field over a discretization of the 3D object and using topological characteristics of the resulting vector field, such as critical points and critical curves, to extract the curve-skeleton. We demonstrate this method on many different types of 3D objects (volumetric, polygonal and scattered point sets) and discuss various extensions of this approach.

Keywords: *3D curve-skeleton, repulsive force field*

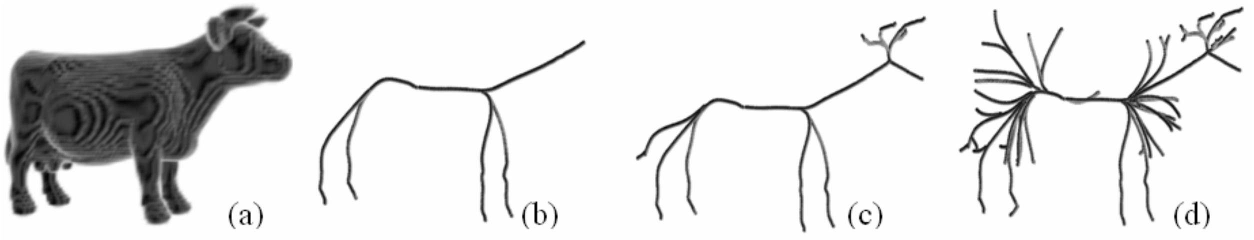

Figure 1: Curve-skeleton hierarchy for the (a) cow model: (b) core skeleton, (c) using 30% low divergence seeds, (d) using 55% low divergence seeds.

1 Introduction

A skeleton is a useful shape abstraction that captures the essential topology of an object in both two and three dimensions. It refers to a thinned version of the original object, which still retains the shape properties of the original object. In 2D, the skeleton is also referred to as the medial-axis. In 3D, the term “skeleton” has been used to describe both a medial-surface and a more line-like representation. In the grass-fire analogy given in [9] the skeleton consists of the points where different fire fronts intersect. If a fire was simultaneously started on the perimeter of the grass (the boundary of the object) the fire would proceed to burn towards the interior of the object. When two fire fronts meet each other the fire will be quenched. In 2D, the fire will quench along a curve. In 3D, the fire fronts will meet along a surface or a curve.

Recently, there has been interest in extracting a *line-like 1D skeletal-representation* from a 3D object. The line-like skeleton is also referred to as a *curve-skeleton* [38], *inverse-kinematic skeleton (IK-skeleton)* [36], or *centerline*. In this paper, we refer to it as a curve-skeleton. Curve-skeletons are useful for many different geometric tasks, such as, virtual colonoscopy and virtual endoscopy [20][45], 3D object registration and visualization [5][6][32], computer animation (both polygonal and volume animation) [8][15][16][27][36][39][42], shape matching [11][35][37], surface reconstruction [26], vessel tracking [6], and curved planar reformation [22][23]. While there is no precise definition for a curve-skeleton, there are numerous desirable properties of both the skeleton and the skeleton computation process. These properties depend upon the application that the curve-skeleton is being used for, and include some of the following: the skeleton should be *thin,* ideally 1-voxel thick, it should capture the "essential shape" of the object (*homotopy*) [24], it should be *centered* within the object, it should be *connected*, different segments of the skeleton should be distinguishable (*component*-wise differentiation), and every interior boundary point should be visible from the skeleton (*reliability [20]).*This last property is desirable for virtual colonoscopy applications. Furthermore, the skeleton computation process should be robust and insensitive to small perturbations/noise on the boundary or to rotations of the object, efficient to compute, reversible, so that the original object can be reconstructed from the skeleton and hierarchical, i.e., allows different hierarchies or level-of-detail skeletons (multiscale [33]) to be computed where each hierarchical level is included in the next level.

Many of these properties are conflicting, for example an object cannot be accurately reconstructed from a thin skeleton. The last property is an important one since a hierarchical skeleton would allow different curve-skeletons to be computed using the same process. An example of a *hierarchical curve-skeleton* is shown in Figure 1.

In this paper, we introduce the concept of a *hierarchical curve-skeleton* and present a method to extract these curve-skeletons from a general 3D object. We also demonstrate how this technique is applicable to all types of 3D objects: polygonal surface representations, volumetric datasets and scattered point sets. Extensions to mesh decomposition and animation are also discussed. The algorithm was applied to many 3D objects, some of which are shown in this paper (the others can be seen on our web site [43]). The method is based upon using a repulsive force field [2][12] and concepts from vector field topology [21] to extract families of curve-skeletons. The method automatically detects "nodes" in the skeleton, which can be used as branch points for virtual navigation [45] or as joints for animation [15][16][39][42].

2 Literature Review

There has been quite a lot of work in generating 2D medial-axes, 3D medial surfaces and 3D curve-skeletons. In this paper, we are concerned with curve-skeleton generation from a 3D object and limit our discussion to work most relevant to the approach described here (a full listing of papers can also be found at [4][43]). A 3D object can be represented either as a set of bounding polygons, scattered points on the surface of an object, or as a segmented volume. Different techniques work on different types of objects.

In [38], a method to discretely determine the curve-skeleton of surface-like objects is given. Voxels are classified as edge, inner, curve or junction using a mask, then removed and reclassified until only curve voxels remain. No hierarchy is defined and joints are not identified. Because it is discrete, the method is sensitive to noise on the boundary. A similar thinning algorithm is given in [30]. In [40], a 2.5D algorithm is described which determines a 2D skeleton for each slice of a 3D voxelized dataset and the results are then merged. This method has the ability to produce level-of-detail skeletons by tweaking the parameters but it is not a strict hierarchy and it breaks down for certain parameter values.

Many approaches are based on first computing the distance field or distance transform and then attempting to thin, cluster and/or path search to find a centerline [15][16][20][29][42][45] computing the gradient to help determine the "sink" direction [10][34]. These algorithms are very sensitive to noise and hierarchy is not determined. Furthermore, the distance field is not an ideal function for "box-like" objects.

Geometric algorithms are given by [3][26] where the Voronoi Diagram is used to help determine the medial surface. Clustering and simplification must be used to get a curve-skeleton (which was not the focus of these papers), and no hierarchy is identified.

In [41], a curve-skeleton for animation is computed from scattered points by computing a neighborhood graph. The neighborhood graph is traversed from a user-selected source, and the centers at each level form the curve-skeleton. The results vary depending upon the selection of the source point and the resulting skeleton is not always centered. In [39], a geometric method is presented which clusters the surface polygons (mesh decomposition) and then determines the IK skeleton from the polygon clusters. Levels-of-detail can be generated by adjusting the size of the polygonal surface clusters, however, the resulting skeletons will not be a strict hierarchy. In this paper, we show how the hierarchical curve-skeleton can be used to decompose the polygon mesh (the inverse of [39]).

Another set of methods attempts to compute some continuous quench function over the object and detect the extremes of that function (which should lie close or on the centerline of the object). These methods include those that use a type of field [1][2][10][12][18][25][27], radial-basis function [28] and intensity maxima [5][6][32]. Most of the above papers do not discuss a hierarchical approach to the curve-skeletonization problem. However, these methods are similar to the approach taken here and are discussed in detail below.

In [2] and [12], 2D and 3D skeletonization algorithms based upon a generalized potential field are presented. “Seed” points near the convex corners of the object are selected and the repulsive force (gradient of the potential field) is analytically derived only along a path determined using a force-following algorithm. Each path started at a seed point ends at a potential minimum detected by a major change in the repulsive force vector direction. The resulting skeleton segments are usually disconnected pieces and a separate re-connection step is necessary to assure connectivity. No hierarchy is defined and only polygonal objects can be handled.

A similar approach is taken in [44]. The visible repulsive force defined there is a special case of the repulsive force used in [2] and [12]: the Newtonian repulsive force. Additionally, the force is computed using only a few samples on the boundary (the visible set) determined by the intersection with a number of sampling rays originating at the current position on the path. The resulting skeleton is not smooth and the identified joints are not robust. In [27], the same visible repulsive force is used, but the force field is computed over the entire voxelized object. The skeleton is extracted using a combination of thinning, clustering and graph searching algorithms and it is unclear how these affect the final position of the identified joints.

3 Methodology

Without loss of generality, and to simplify the discussion in the remainder of the paper, we assume that a 3D object refers to a 3D voxelized discretization of that object. Polygonal models can be converted into volumetric objects by voxelization [14]. In Section 4.1, we also show how the method is applicable to objects represented as scattered point sets.

The methodology presented here is an enhancement and extension of the methods presented in [2][12][27][44]. We explicitly compute a repulsive force field over the entire object as in [27] (not just along the path as in [2][12][44]). The resulting 3D dataset is a *vector field*: a 3D array where each voxel contains a *vector* value (magnitude and direction). The difference from all the previously discussed approaches is that once we compute this vector field, we use its topological characteristics [13][17][21] such as critical points and low divergence points together with high curvature points on the boundary, to extract our skeleton hierarchy.

The presented methodology has a number of significant advantages over the previous methods, namely: it directly produces connected skeletons without the use of a re-connection step, it works both on un-segmented objects (where only the boundary of the object is known) and segmented objects (where the interior is known), it has the ability to produce a strict hierarchy of skeletons and it has the ability to automatically extract an "IK skeleton" that can be used for animation i.e., the joints are automatically identified.

3.1 Repulsive Force Function

Our repulsive force is defined similarly to the repulsive force of the generalized potential field [2][12]. The key idea behind the potential field approach is to generate a force field inside the object by charging the object's boundary. The basic procedure to compute the generalized potential and force for polyhedral objects is summarized in [12]. Since our algorithm operates on 3D objects represented by voxels, our boundary elements are also voxels, and we consider them to be point charges (this also simplifies the calculations involved in computing the force field). A boundary point is defined as an object voxel that has an exterior (empty) neighbor. An interior voxel (interior point) is an object voxel whose neighbors are all object voxels.

The repulsive force at a point due to a nearby point charge is defined as a force pushing the point away from the charge with a strength that is inverse proportional to a power of the distance between the point and the charge namely:

$$\overrightarrow{F_{PC}} = \frac{\overrightarrow{CP}}{R^m} \qquad (1)$$

where $\overrightarrow{F_{PC}}$ is the repulsive force at point *P*, due to point charge *C*, $\overrightarrow{CP}$ is the normalized vector from *C* to *P* which gives the direction of the force, *R* is the distance between *P* and the charge *C* and the power *m* is called the order of the force function (*m*=2 for the Newtonian force).

The force at a point *P* due to the influence of multiple point charges can be found discretely by simply summing all the forces at *P*.

$$\overrightarrow{F_P} = \sum_i \overrightarrow{F_{PC_i}} \tag{2}$$

where $\overrightarrow{F_P}$ is the resulting force at point *P* and $\overrightarrow{F_{PC_i}}$'s are the forces due to the individual point charges C_i.

We consider every boundary voxel to be a point charge and the repulsive force at each interior voxel is explicitly computed by summing the influences of all the point charges. The resulting vector field is also called a force field.

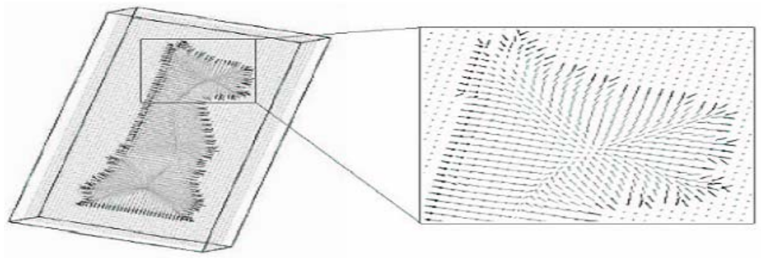

Figure 2: Repulsive force field of a 3D chess piece. The vectors at each point are shown with arrows. The middle slice of the 3D object is shown in zoom.

A high order of the force function (*m*) will cause the local boundary points to have a higher influence on a given interior point than the more distant boundary voxels, thus creating a vector field with sharper path-lines because it follows the local boundary topology more closely. A low value for the *m* parameter will produce a smoother vector field, with more rounded corners, since the vector direction at a particular point is now influenced by more boundary charges [2]. Figure 2 shows the repulsive force field along a center slice of a 3D chess piece.

Because the algorithm uses all of the boundary points for force computation, *visibility* errors can result for objects with tapered limbs (like a comb) since individual points within each prong of the comb cannot "see" boundary points on the other prongs and those points should not be considered in the force field calculation [12][2]. This can be overcome by determining the visibility with a line-of-sight calculation, which checks to see that the surface is not pierced when connecting a point to a charge with a straight line. While this is more accurate, it unfortunately increases the running time substantially and is only applicable if the surface is known. Using a high order of m also minimizes this effect [2] by reducing the influence of more distant boundary charges.

The computational complexity of the force field calculation depends on the number of object and boundary voxels: $O(No \times Nb)$ where No is the number of object voxels and Nb is the number of boundary voxels. Since Nb is a fraction of No, the computational complexity is approximately $O(No^2)$. This is the most time consuming step of the algorithm, accounting for about 98% of the total running time.

3.2 Computing the Curve-Skeleton

Given a 3D vector field, certain concepts from vector field visualization can be used to identify three different types of "seed points" in the vector field. The seeds are the starting point of "field-lines" or "stream-lines" (these lines are found by "force-following") which define our curve-skeleton segments.

3.2.1 Critical points and the "core skeleton"

Critical points are important vector field topology components and are often used in vector field visualization. These are the points where the magnitude of the force vector vanishes. In Figure 2, a critical point is visible in the middle of the head of the chess piece. In [13][17][21][31], a full discussion of the visualization of vector-field topology and the different types of critical points can be found. In what follows, we present a brief description relevant to extracting the curve-skeleton.

Critical points are difficult to locate in a vector field, particularly because they do not necessarily occur at the given sample locations, but often occur in between sampling points. A good heuristic for detecting critical points is described in [17]: a zero in the vector field occurs when all 3 components of the force vector (x, y and z) vanish, thus, if we can identify a region where each vector component changes sign, the region is a candidate for containing a critical point. In our case, the smallest region we can consider is a voxel cell; the force field value is evaluated at each of the 8 corners of a grid cell using tri-linear interpolation. Cells containing both positive and negative values for every vector component (x, y and z) are potential candidates for containing critical points. Candidate cells are recursively divided into 8 sub-cells and the candidacy test is repeated for each sub-cell. The process ends either when a cell fails the candidacy test or when the cell is too small, and still a candidate, in which case a critical point is assumed to exist at the center of the cell.

Once extracted, the critical points are classified. Different types of critical points can be identified [13][17][31] including: attracting nodes (where all vectors are pointing towards the critical point), repelling nodes (where all the vectors are pointing away from the critical point) and saddle points (where some vectors are pointing towards the critical point and others away from it). Critical points can be classified by evaluating the real and imaginary components of the eigenvalues of the Jacobian matrix of the vector-field at the critical point: a positive real part of an eigenvalue denotes the existence of a repelling direction (given by the corresponding eigenvector), a negative real part denotes an attracting direction, and an imaginary part describes a spiraling motion around the point. If all the real eigenvalues are of the same sign, the critical point is classified as an attracting (if negative) or repelling (if positive) node. The critical point is said to be a saddle if two real eigenvalues have the same sign and the third one has an opposite sign. Saddles are a special type of critical points for the purpose of extracting the curve-skeleton from a 3D object. Saddle points will occur between attracting or repelling nodes directing the flow towards one or another and can be used to connect them. Intuitively, the outward flow directed away from a saddle point, can only reach an attracting node or another saddle inside the object since the vector field was generated by the closed boundary of the object and points towards the interior.

Path-lines are "seeded" from saddles in the direction of the eigenvectors corresponding to the positive eigenvalues, which indicate the outward flow. Next, a path-line force-following algorithm is applied which stops at another critical point or when it arrives at a previously visited location. The force-following algorithm evaluates the force value at each point along a path and moves in the force direction with small steps. Samples taken along the integration path started at a saddle point form a *skeleton segment*. Skeleton segments connecting all the critical points of the force field are known as *critical-curves* [21] and form the *core skeleton*.

The core skeleton represents level 0 in the skeleton hierarchy. However, the core skeleton contains only part of the central curve-skeleton for a given object. The next level-of-detail is added by considering another important feature of the force field: the divergence.

3.2.2 Low divergence points and the “level 1 skeleton hierarchy”

The divergence of a vector field in a given region of space is a scalar quantity that characterizes the rate of flow leaving that region. Divergence is an interesting property because it measures the “sinkness” of a point [10]. A negative divergence value at a point indicates that the flow is moving mainly towards the given point. Divergence can be computed at a grid point using the formula:

$$divF = \nabla \cdot F = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

where F_x, F_y, F_z are the force field components in the x, y and z direction. Of interest are points with low divergence values, which indicate a “sink”. Either a “local min” or a threshold can be used to identify new seed points (note that the user can determine using these values how many new seed points are identified). From each of these new seed points, a new field-line is generated. These field-lines will connect to the core skeleton. By varying the divergence threshold the user varies the number of seed points selected and correspondingly the number of new skeleton segments, generating an entire hierarchy of skeletons that we call the *level 1 skeleton*. In Figure 1(b) the core skeleton of the cow is shown. In (c) and (d) varying thresholds values are applied to extract the divergence seeds. Different thresholds can be chosen based upon the application of the curve-skeleton.

3.2.3 High Curvature points and the level 2 skeleton hierarchy

Another set of seed points for skeletal segments consists of the convex corners of the object. These will create paths from the boundary to the skeleton as in [2] and generate the “tendrils” found in medial axis representations. For objects without convex corners, such as curved objects, the curvature on the boundary can be computed [19] and areas of high curvature (threshold) can be used to seed new path-lines. Local curvature maxima could also be used to determine boundary seeds. A disadvantage of boundary seeding based on the curvature is that it is affected even by small amounts of noise present on the object’s boundary. A possible solution to this problem is to consider an extended neighborhood when computing the curvature at boundary voxels, not just the face, edge and vertex neighbors [19]. For certain applications, such as virtual navigation, boundary seeds could be used differently, i.e., to specifically generate certain navigation paths. All the boundary-seeded curves connect with the core skeleton.

The skeleton segments originated at seed points on the boundary will create another level of hierarchy for each curve-skeleton computed thus far. Each of the level-1 skeletons will form the root of a new hierarchy developed by varying the number of boundary-seeded segments (via the user specified curvature threshold). We call this new hierarchy level the *level 2 skeleton hierarchy*.

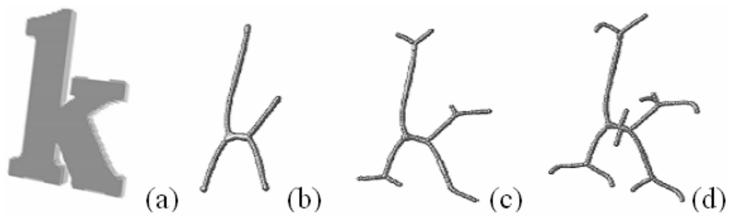

Figure 3: A 3D model of the letter K with varying curve-skeletons. (a) 3D model, (b) core skeleton using only critical points, (c) adding low divergence values (top 20%), (d) adding boundary seeds at high curvature boundary locations.

3.3 The Algorithm

The algorithm can be summarized in five steps as follows:

Step 1. Identify the boundary voxels of the 3D object as the source of the repulsive force field. If 3D polyhedral objects are used, they can be discretized onto a 3D grid by voxelization.

Step 2. Compute the *repulsive force function* at each object voxel. The function specification is given in Section 3.1. After computation, a 3D *vector-field* results.

Step 3. Detect the *critical points* of the vector field and connect them using path-lines by integrating over the vector-field. The method is described in Section 3.2.1 and produces the *core skeleton*.

Step 4. Compute the *divergence* of the vector-field at each voxel. Points with low divergence values are selected as new seeds for new skeleton segments. Varying the divergence threshold (given as a percentage, i.e., the top 20%) creates the *level 1 hierarchy* after the core skeleton.

Step 5. Compute the *curvature* at every boundary voxel and select new seed points based on a user-supplied curvature threshold, given again as a percentage of the highest curvature value in the dataset (i.e. top 30%). This adds another level of hierarchy to the core and divergence skeletons, the *level 2 skeleton hierarchy*. Note that the boundary seeds can be added either directly to the core skeleton or to a level 1 skeleton. However, a strict hierarchy is achieved only if the hierarchy levels below the current level are fixed: for example, in order to generate a strict level 2 hierarchy, the core skeleton and the number of divergence seeds must be fixed and only the number of boundary seeds is allowed to vary. The curve-skeleton extraction algorithm described here is general, and can also be used to extract a curve-skeleton from other types of quench functions such as [1][18][27][28].

3.4 Skeleton representations

The skeleton produced by the algorithm consists of a set of seed points connected by points sampled by the force-following algorithm. Alternatively, each curve sample can be mapped to the nearest grid location, creating a less smooth voxel skeleton, consistent with the discretized nature of the original 3D object. Figure 4(a) shows the curve representation and 4(c) shows the corresponding voxel skeleton of a cow model.

Another approach, useful for animation [16][42] or matching [37], is to transform the skeleton into straight-line segments by treating the critical points and the points where different skeleton segments meet as “joints” and connecting these points with straight lines. The resulting skeleton will be suitable for importing into commercial animation packages (Maya, 3D Studio Max, etc) since the joints are automatically detected (joints are the end-points of skeleton segments). In the discussion section we show how the “skinning” process (attaching surface polygons to the skeleton) can also be done automatically using the repulsive force field. Figure 4(d) shows the IK-skeleton for the cow.

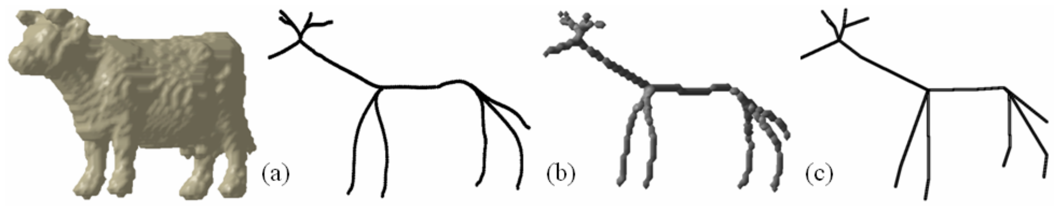

Figure 4: Skeleton representations for the cow model (a): curve skeleton (b), voxel skeleton (c) and straight-line skeleton (d).

4 Results and Discussion

The algorithm was tested on many different volumetric datasets containing various 3D objects both volumetric and voxelized polygonal objects. In Figure 5, the core skeleton and a divergence-based skeleton (using the top 20% of divergence values) of a few sample objects are shown. More objects can be seen at [43].

The divergence and the curvature thresholds should be chosen based on the application of the resulting skeleton. There are no optimal values for these parameters, as different applications may require skeletons of different complexity. The simplest skeleton our algorithm can extract is the core skeleton, using no divergence or curvature seeds. If a more complex skeleton is desired, the user can increase the number of divergence seeds, or the number of curvature seeds or both, keeping in mind that curvature seeds are very sensitive to noise on the boundary of the object.

The order of the repulsive force function (m) has a dominant role in the skeleton outcome. It shapes the vector field, determining the location of critical points and divergence seeds. While the force function is object dependent, experimental results showed that a high order (m>5) produced stable results. The results shown in Figure 5 used m=6.

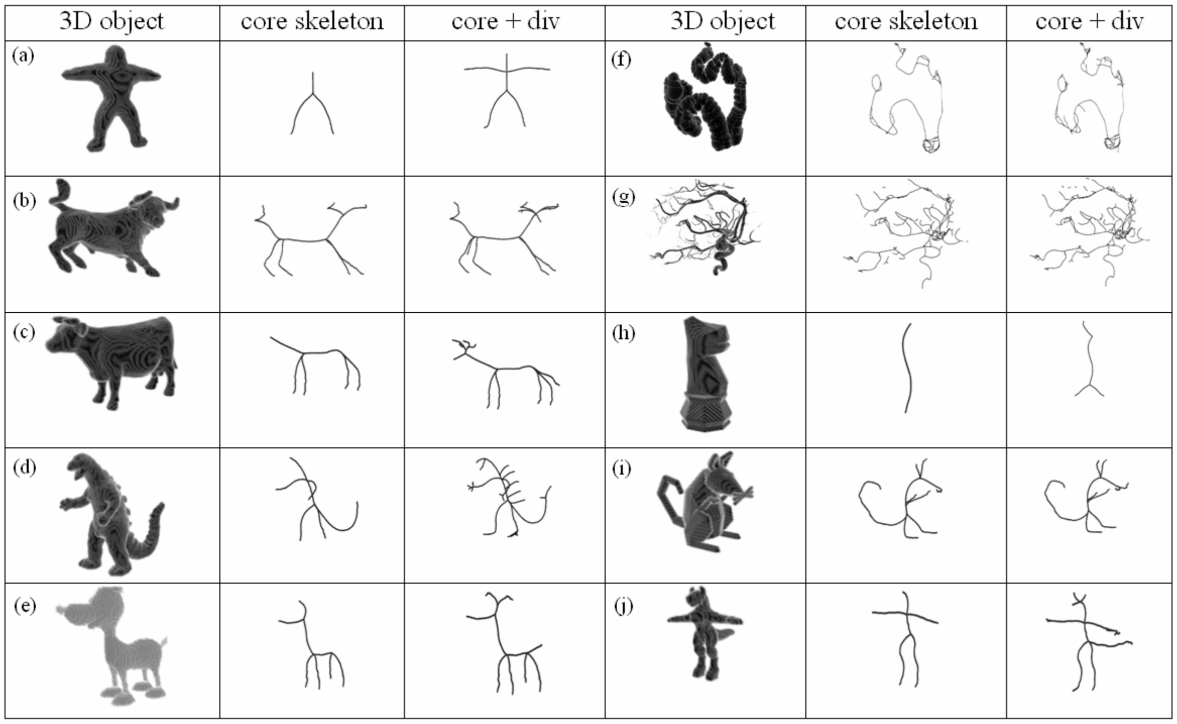

Figure 5: Some 3D objects and their skeletons. The first column shows the core skeleton, the second shows the skeleton with 20% of the low divergence points.

Resolution/discretization is an important external factor affecting the performance of the algorithm and changing the accuracy of the solution, especially for scattered point sets. Clearly a 4^3 grid will yield a less accurate result than a 60^3 grid. For the results in this paper (Figure 5), the voxelized datasets are all less than 70^3, except for the *colon* (Figure 5(f)) and *angio* (Figure 5(g)) datasets, which were on the order of 200^3 (these were original voxel datasets). A 1 or 2 voxels thick object region will not have enough resolution to properly compute the repulsive force field in that region (this is a common simulation issue and standard mutiresolution solutions can be employed [7]). For such regions, the repulsive force field is extremely sensitive to even small boundary perturbations, resulting in discontinuities of the flow pattern inside the object's boundaries. As a result, the interior skeleton can become disconnected just because the force-following algorithm becomes stuck in these perturbed regions of the force field. To overcome this problem, one can either increase the resolution of the voxelization or pad the object with a number of extra layers of voxels (dilation), making the object thicker. Padding produces a smoothing effect of the object's boundary, which makes it suitable for noisy objects, but can also merge object features that are very close to each other.

The time to determine the skeleton is dominated by the repulsive force field computation. It ranged from about one minute for most of the datasets to about a half an hour for the *colon* dataset (Figure 5(f)) (on a standard PC). The computational complexity of the repulsive force field is $O(n^2)$ (see Section 3.1). Speedup of the computation can be achieved by sampling the boundary (i.e., using only a fraction of the boundary points for force field generation) or by using a less accurate distance computation in place of Euclidean distance.

Currently, the entire force field is computed and stored in memory. Although the memory requirement is linear in the size of the original volume, for massive datasets, where the size of the vector field can easily reach gigabytes, out-of-core solutions need to be investigated.

4.1 Scattered Point Sets

For objects from scattered point sets where the surface is not known (such as from scanners) the skeletonization algorithm described here can still be used. Although a discrete grid is needed for the computation of the repulsive force field, it is not necessary to know what is inside vs. outside the object or to have the exact boundary of the object specified. Only a set of surface points is necessary to compute the vector field. The sample points are mapped to a grid (Eulerian approach [26]) and the force field is computed on the *entire* grid (inside and outside the object, since these distinctions are not known a-priori). The curve-skeleton is then determined using the described algorithm. Many curve-skeletons will result: the correct interior one, which will be connected, and a number of exterior skeletons, which will hit the boundary of the grid and can be removed [26]. In Figure 6 we show the core and level-1 skeletons for a horse dataset represented by scattered points on the surface. If the sample points on the object's surface are not dense enough, critical points (saddles) will occur between the sample points. These critical points are artificially created by the sampling process and should not be considered as part of the core skeleton. They can be removed by imposing a minimum distance constraint between a critical point and a surface sample.

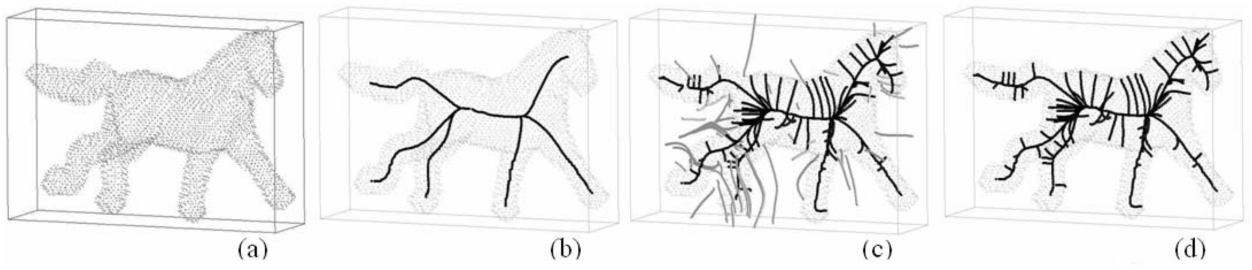

Figure 6: Curve-skeleton from scattered points: (a) scattered point set, (b) "core" skeleton. (c) skeleton inside (in black) and outside the object (in gray) using top 20% divergence points as seeds. (d) "inside-only" skeleton obtained from (c) by removing skeleton segments touching the bounding box.

4.2 Hierarchical Mesh Decomposition

The computed skeleton can also be used to generate a hierarchical mesh decomposition as in [39], taking advantage of the fact that we can differentiate between different segments of the skeleton. Starting from every boundary point of the object and following the field-lines with the same force-following algorithm used to extract the skeleton, we can attach each boundary point to the first skeleton segment encountered during the force-following process. Figure 7(b) shows the boundary decomposition of a cow model. The method generates a decomposition of the volumetric object into logical parts (component-wise differentiation), corresponding to the skeleton segments and it is similar to the "skinning" process of an animation character, where the surface polygons are attached to the skeleton segments manually.

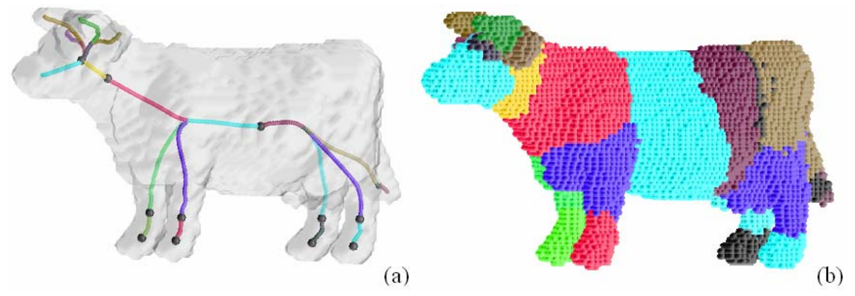

Figure 7: Skeleton for animation and object decomposition. (a) Cow IK skeleton; each segment in a different color (b) object decomposition (points on the boundary are color coded by their associated skeletal segment).

The resulting skeleton is very suitable to animation tasks. The algorithm can produce straight-line skeletons and automatically identify the joints as the critical points or the points where two skeleton segments meet. In the case of a polygonal model, each surface polygon will be attached to one or more skeleton segments by tracing the field-lines starting at each of the polygon's vertices. If all vertices are attached to the same segment, the entire polygon is assigned to that segment and if vertices are assigned to different segments, a weight is assigned to each segment based on distance or other factors for a smoother animation.

The extracted curve-skeleton has many of the desirable properties described in the introduction. The force-following algorithm used to discover new skeleton segments guarantees that the skeleton is *one-voxel thick*. *Homotopy* however is not guaranteed and in fact can be compromised by inadequate resolution, producing disconnected skeletons even for connected objects. A post processing step could be used to connect the disjoint parts. *Centeredness* is also not guaranteed since each interior voxels is influenced by all boundary voxels even if they are not visible from that point. If visibility of a boundary point from the interior voxel is taken into account, then the resulting skeleton is more centered. Given the adequate resolution, c*onnectivity* of the core skeleton is guaranteed by the topology of the repulsive force field. It can be shown that between every pair of neighboring attracting nodes (where the force following stops), there must be a saddle point (because the force vector must change sign as we move from one attracting node to the other). These saddles are the seeds of our core skeleton and the skeleton segments seeded there will connect the attracting nodes. Connectivity of the level 1 and level 2 skeletons is guaranteed because the force following algorithm stops only when reaching a previously visited location. *Component-wise differentiation* was demonstrated above by object decomposition. *Reliability* (seeing all the boundary points from the skeleton) is a post-process and it can be checked using the methods described in [20]. The algorithm is *robust to noise* on the object's boundary because a central core is detected. In Figure 8(d), the skeleton of a 3D object with noise is shown. The levels of hierarchy will be different for the object with noise and without but this can be controlled by varying the threshold used for the divergence values or by padding the object. Quantitative comparisons of the skeleton can be made by either comparing critical points and/or the entire path lines.

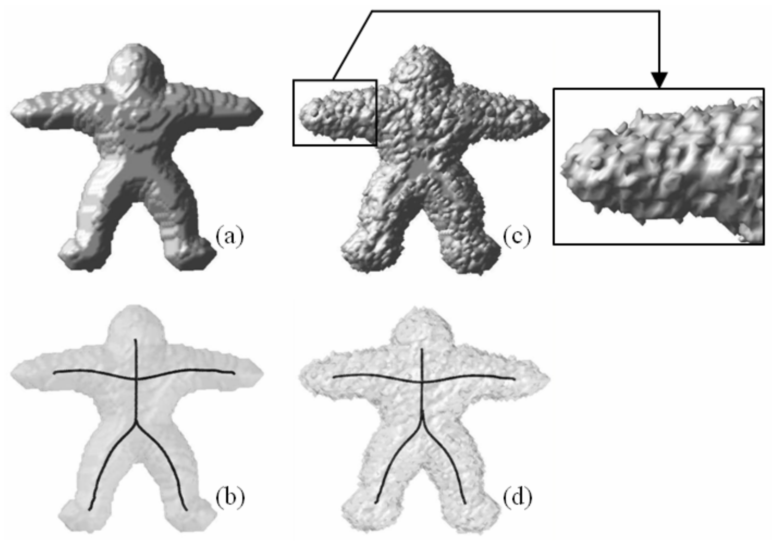

Figure 8: Object (a) and skeleton (b). Object with artificially added noise (c) and corresponding skeleton (d).

Although the algorithm presented here is not as fast as the distance field-based approaches, the resulting curve-skeleton is much cleaner. Reconstruction of the original object is generally not possible from curve-skeletons because they retain only the core of the original shape.

5 Conclusions

In this paper, we have presented a robust algorithm to extract the curve-skeleton from any 3D object, polygonal, voxel or scattered point set. It is based upon computing a repulsive force field and using vector-field topology to extract a family of hierarchical curve-skeletons. The curve-skeleton hierarchy that results from the method described can be tailored to many different applications including virtual navigation, animation, and matching.

Acknowledgements

We gratefully acknowledge the support of the National Science Foundation (ITR 0118760) and (ITR 0082634). We would like to thank J. Dwoskin for the implementation of the voxelization code.

The algorithm was implemented in C++ and can be downloaded from [43]. The online program takes as input a small number of parameters: the order of the repulsive force function *m*, a divergence threshold and a growth factor. The growth factor is used to overcome resolution problems as described above.

References

1. Abdel-Hamid G, Yang Y (1994) Multiresolution Skeletonization: An Electrostatic Field-Based Approach. Proc. IEEE International Conference on Image Processing (1): 949-953.
2. Ahuja N, Chuang J-H (1997) Shape Representation Using a Generalized Potential Field Model. IEEE Trans. Pattern Analysis and Machine Intelligence, 19(2):169-176.
3. Amenta N, Choi S, Kolluri R-K (2001) The Power Crust. Proc. of 6th ACM Symp. on Solid Modeling, pp 249-260.
4. Medial Axis/Surface Transform Website http://www.cs.princeton.edu/~min/meshclass/.
5. Aylward S-R, Jomier J, Weeks S, Bullitt E (2003) Registration and Analysis of Vascular Images. International Journal of Computer Vision, 55 (2-3): 123 - 138.
6. Aylward S-R, Bullitt E (2002) Initialization, Noise, Singularities, and Scale in Height Ridge Traversal for Tubular Object Centerline Extraction. IEEE Trans. Medical Imaging 21(2): 61-75.
7. Berger M, Oliger J (1984) Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. Journal of Computational Physics, 53:484-512.
8. Bloomenthal J (2002) Medial Based Vertex Deformation. Proc. ACM SIGGRAPH/Eurographics Symposium On Computer Animation, pp 147-151.
9. Blum H (1967) A Transformation for Extraction New Descriptors of Shape. Models for the Perception of Speech and Visual Form, MIT Press, pp 362-380.
10. Bouix S, Siddiqi K (2000) Divergence-Based Medial Surfaces. Proc. European Conf. on Computer Vision, pp. 603-618.
11. Brennecke A, Isenberg T (2004) 3D Shape Matching Using Skeleton Graphs. Simulation und Visualisierung (SimVis), SCS European Publishing House, Erlangen, San Diego, pp 299-310.
12. Chuang J-H, Tsai C, Ko M-C (2000) Skeletonization of Three-Dimensional Object Using Generalized Potential Field. IEEE Trans. Pattern Analysis and Machine Intelligence, 22(11):1241-1251.
13. Dickinson R-R (1991) Interactive Analysis of the Topology of 4D Vector Fields. IBM Journal of Research and Development, 35(1/2):59-66.
14. Fang S, Chen H (2000) Hardware Accelerated Voxelization. Computers and Graphics, 24(3):433-442.
15. Gagvani N, Silver D (1999) Parameter Controlled Volume Thinning. Graphical Models and Image Processing, 61(3):149-164.
16. Gagvani N, Silver D (2001) Animating Volumetric Models. Graphical Models, 63(6):443-458.
17. Globus A, Levit C, Lasinski T (1991) A Tool for Visualizing the Topology of Three-Dimensional Vector Fields. Proc. IEEE Visualization, pp 33-40.
18. Grigorishin T, Yang Y-H (1998) Skeletonization: An Electrostatic Field-Based Approach. Pattern Analysis and Applications, 1:163-177.
19. Hameiri E, Shimshoni I (2002) Estimating the Principal Curvatures and the Darboux Frame from Read 3D Range Data. 1st International Symposium on 3D Data Processing Visualization and Transmission, pp. 258-267.
20. He T, Hong L, Chen D, Liang Z (2001) Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs. IEEE Trans. Visualization and ComputerGraphics, 7(4):333-342.
21. Helman J-L, Hesselink L (1991) Visualizing Vector Field Topology in Fluid Flows. IEEE Computer Graphics and Applications, 11(3):36-46.
22. Kanitsar A, Fleischmann D, Wegenkittl R, Felkel P, Gröller M-E (2002) CPR: Curved Planar Reformation. Proc. IEEE Visualization, pp 37-44.

23. Kanitsar A, Wegenkittl R, Fleischmann D, Grőller M-E (2003) Advanced Curved Planar Reformation: Flattening of Vascular Structures. Proc. IEEE Visualization, pp 43-50.
24. Kong T-Y, Rosenfeld A (1989) Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing, 48(3):357-393.
25. Leymarie F, Levine M (1992) Simulating the Grassfire Transform using and Active Contour Model. IEEE Trans. on Pattern Analysis and Machine Intelligence, 14(1):56-75.
26. Leymarie F (2003) 3D Shape Representation via the Shock Scaffold. Ph.D. Thesis, Brown University, Providence, USA.
27. Liu P, Wu F, Ma W, Liang R, Ouhyoung M (2003) Automatic Animation Skeleton Construction Using Repulsive Force Field, Proc. IEEE 11th Pacific Conference on Computer Graphics and Applications, pp 409.
28. Ma W, Wu F, Ouhyoung M (2003) Skeleton Extraction of 3D Objects with Radial Basis Functions. Proc. Shape Modeling International, pp 207-215.
29. Manzanera A, Bernard T, Preteux F, Longuet B (1999) Medial faces from a concise 3D thinning algorithm, Proc. IEEE International Conference on Computer Vision, pp 337-343.
30. Palagyi K, Kuba A (1999) Directional 3D Thinning using 8 Subiterations. Proc. Discrete Geometry for Computer Imagery, Lecture Notes in Computer Science 1568:325-336.
31. Perry A-E, Chong M-S (1987) A Description of Eddying Motions and Flow Patterns Using Critical-Point Concepts. Ann. Review of Fluid Mechanics, 19:125-155.
32. Pizer S-M, Fritsch D-S, Yushkevich P-A, Johnson V-E, Chaney E-L (1999) Segmentation, Registration, and Measurement of Shape Variation via Image Object Shape. IEEE Trans. Medical Imaging, 18(10):851-865.
33. Pizer S, Siddiqi K, Szekely G, Damon J, Zucker S (2003) Multiscale Medial Loci and Their Property. International Journal of Computer Vision 55(2-3):155-179.
34. Schirmacher H, Zockler M, Stalling D, Hege H (1998) Boundary Surface Shrinking - A Continuous Approach to 3D Center Line Extraction. Proc. Image and Multidimensional Digital Signal Processing, pp 25-28.
35. Sebastian T, Klein P, Kimia B (2002) Shock-Based Indexing into Large Shape Databases. Proc. European Conference on Computer Vision, 3:731-746.
36. 3D Studio Max (3DSMax), Discreet, www.discreet.com.
37. Sundar H, Silver D, Gagvani N, Dickinson S (2003) Skeleton Based Shape Matching and Retrieval. Shape Modeling and Applications, pp 130-138.
38. Svensson S, Nystrom I, Sanniti di Baja G (2002) Curve Skeletonization of Surface-like Objects in 3D Images Guided by Voxel Classification. Pattern Recognition Letters, 23 (12):1419-1426.
39. Tal A, Katz S (2003) Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts. ACM Trans. on Graphics, 22(3):954-961.
40. Telea A, Vilanova A (2003) A Robust Level-Set Algorithm for Centerline Extraction. Eurographics/IEEE/TCVG Symp. on Visualization, pp 185-194.
41. Verroust A, Lazarus F (2000) Extracting Skeletal Curves from 3D Scattered Data. The Visual Computer, 16(1):15-25.
42. Wade L, Parent R-E (2002) Automated Generation of Control Skeletons for Use in Animation. The Visual Computer, 18:97-110.
43. Vizlab Website, Rutgers, The State University of New Jersey: http://www.caip.rutgers.edu/vizlab_group_files/vizlab.html.
44. Wu F-C, Ma W-C, Liou P-C, Liang R-H, Ouhyoung M (2003) Skeleton Extraction of 3D Objects with Visible Repulsive Force. Computer Graphics Workshop 2003, Hua-Lien, Taiwan.
45. Zhou Y, Kaufman A, Toga A-W (1998) Three-dimensional Skeleton and Centerline Generation Based on an Approximate Minimum Distance Field. The Visual Computer, 14:303-314.