# Simultaneous Shape Decomposition and Skeletonization

Jyh-Ming Lien*  Nancy M. Amato†

Technical Report TR05-015
Parasol Lab.
Department of Computer Science
Texas A&M University
December 12, 2005

**Abstract**

Shape decomposition and skeletonization share many common properties and applications. However, they are generally treated as independent computations. In this paper, we propose an iterative approach that simultaneously generates a hierarchical shape decomposition and a corresponding set of multi-resolution skeletons. In our method, a skeleton of a model is extracted from the components of its decomposition — that is, both processes and the qualities of their results are interdependent. In particular, if the quality of the extracted skeleton does not meed some user specified criteria, then the model is decomposed into finer components and a new skeleton is extracted from these components. The process of simultaneous shape decomposition and skeletonization iterates until the quality of the skeleton becomes satisfactory. We provide evidence that the proposed framework is efficient and robust under perturbation and deformation. We also demonstrate that our results can readily be used in problems including skeletal deformations and virtual reality navigation.

*e-mail:neilien@cs.tamu.edu
†e-mail:amato@cs.tamu.ed

# 1 Introduction

Shape decomposition partitions a model into (visually) meaningful components. Recently shape decomposition has been applied to texture mapping [39], shape manipulation [23], shape matching [33, 16, 18], and collision detection [27]. Early work on shape decomposition can be found in pattern recognition and computer vision; see surveys in [38, 48].

A *skeleton* is a lower dimensional object that essentially represents the shape of its target object. Because a skeleton is simpler than the original object, many operations, e.g., shape recognition and deformation, can be performed more efficiently on the skeleton than on the full object. The process of generating such a skeleton is called skeleton extraction or *skeletonization*. Examples of *automatic* skeleton extraction include the Medial Axis Transform (MAT) [9] and skeletonization into a *one dimensional poly-line skeleton* (or simply *1D skeleton*) [12, 28, 23].

Skeletons have been extracted from different sources, such as voxel (image) based data [50, 36, 8], boundary represented models [13, 2, 47], and scattered points [45], and for different purposes, such as shape description [40, 42], shape approximation [3, 49], similarity estimation [19], collision detection [10, 22], biological applications [1], navigation in virtual environments [26], and animation [44, 23].

Although it has been noted before that a good shape decomposition can be used to extract a high quality skeleton [28, 23] and that a high quality skeleton can be used to produce a good decomposition [27], this relationship between shape decomposition and skeleton extraction is a relatively unexplored concept, especially in 3D. Instead, when a relationship is noted, the skeletons are usually treated as an intermediate result or a by-product of the shape decomposition.

In this paper, we propose an integrated framework for simultaneous shape decomposition and skeleton extraction that not only acknowledges, but actually exploits the interdependence between these two operations. First, a simple skeleton is extracted from the components of the current decomposition. Then, this extracted skeleton is used to evaluate the quality of the decomposition. Finally, if the skeleton is satisfactory under some user defined criteria, we report the skeleton and the decomposition as our final results. Otherwise, the components are further decomposed into finer parts using approximate convex decomposition (ACD) [28, 30, 29], which decomposes a given component by 'cutting' its most concave features. Figure 1 illustrates this proposed framework and Figure 2 shows an example of the co-evolution process of the shape decomposition and skeleton extraction.

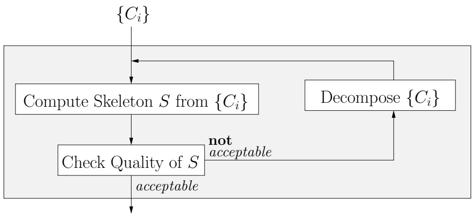



Figure 1: Simultaneous shape decomposition and skeleton extraction. The set $\{C_i\}$ is a decomposition of the input model $P$ and initially $\{C_i\} = \{P\}$.

Our proposed approach has several advantages and makes contributions as listed below.

- This recursive refinement strategy generates multi-resolution skeletons, from coarse to fine levels of detail, which is highly desirable for some applications.

- *Divide-and-conquer* algorithms which operate on the decompositions or skeletons can be more efficient because refinement is applied to the more complex regions but not to areas with less variation.

- The extracted skeleton is invariant under translation, rotation, and uniform scale, and is not very sensitive to the boundary noise and skeletal deformations.

- Our approach does not require any pre-processing, e.g., model simplification, or any post-processing, e.g., skeleton pruning, which are required by many of the existing methods, e.g., [27, 23, 47].

- Our framework is general enough to work for both 2D polygons and 3D polyhedra.

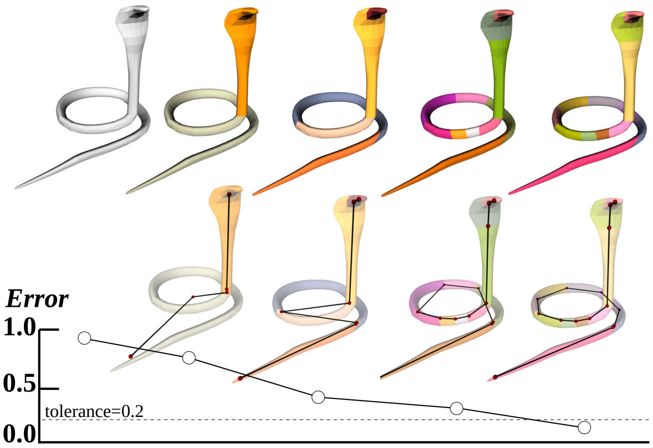



Figure 2: The skeleton (shown in the lower row) evolves with the shape decomposition (shown in the upper row).

## 2 Related Work

Both shape decomposition and skeleton extraction have been studied for decades and there is a large amount of previous work. In this review, we concentrate on recent developments most relevant to our proposed method.

**Shape decomposition**. Inspired by psychological studies, such as recognition by components [7] and the minima rule [20, 21], methods have been proposed to partition models at salient features to produce visually meaningful components. In pattern recognition, Rom and Medioni [38] partition a model into a set of tubular (generalized cylinder) shapes according to their curvature properties. As a preprocessing step for mesh generation, Sonthi et al. [32] identify closed sets (loops) of edges with required convexity and use them to decompose a model into solid parts. However, these methods work best with simple models with sharp internal angles, such as mechanical parts.

Methods that are applicable to models with general shapes also exist. Wu and Levine [48] propose a partitioning method based on a simulated electrical charge distribution on the surface of a model. Mangan and Whitaker [33] and Page et al. [35] decompose polygonal meshes by applying watershed segmentation with curvature computation. Li et al. [27] decompose polygonal meshes at critical points along skeletons obtained via model simplification. Dey et al. [16] segment a model, in $\mathbb{R}^2$ or $\mathbb{R}^3$, into *stable manifolds*, which are collections of Delaunay complexes of sampled points on the boundary. Katz and Tal [23] cluster mesh facets into fuzzy regions, carefully partition facets in those regions, and successfully produce perceptually clean cuts between decomposed components. A similar approach using a different clustering technique can also be found in [31]. Interactive methods [25, 18] that identify features via human assistance have also been shown to produce high quality and clean decompositions.

**Skeletonization**. The Medial Axis (MA), Voronoi diagram, Shock graph and Reeb graph are common skeleton representations. Although the MA can represent a lossless shape descriptor [9], it is difficult and expensive to compute accurately in high ($> 2$) dimensional space [14]. Several ideas for approximating the MA have been proposed, e.g., using Voronoi diagram, and its dual Delaunay triangulation [2, 4, 17], of densely sampled points from the object boundary. Shock graphs [43, 15], another representation of the MA, encode the formation order and, therefore, the importance of each part of the MA. Reeb graphs, a type of 1D skeleton, extracted from various Morse functions, are a powerful tool for shape matching [45, 41, 5, 19]. Since Morse functions are defined on mesh vertices, re-meshing [19, 5] is usually needed to generate a good (accurate) skeleton.

Several methods have been proposed to extract a skeleton from the components of a decomposition [28, 23]. Skeletons can also be constructed by simplifying (contracting) a polygonal mesh to line segments [27].

**Multi-scale and multi-resolution skeletons**. Multi-scale skeletons [37, 34] consist of a set of skeletons, $S_0, \ldots, S_N$, whose union represents a complete skeleton of the model. $S_0$ is the most important part of the skeleton, representing global topology, while $S_N$ encodes local features and is sensitive to local changes. Multi-resolution skeletons [19] consist of a set of skeletons, $S_0, \ldots, S_N$, that encode topology at different levels of detail. $S_0$ will have the coarsest skeleton and $S_N$ will contain the most detailed information. This representation is desired for some applications. For instance, to extract similar items from a 3D database, a rough skeleton can be used to reject many unlikely models and incrementally refine the skeleton to get better matches. As previously mentioned, one of the features of our framework is that its recursive nature results in the construction of multi-resolution skeletons.

# 3 Preliminaries

Let $P$ be a polyhedron represented by its boundary $\partial P$ and let $H_P$ be the convex hull of $P$.

**Approximate Convex Decomposition**. A set of components $\{C_i\}$ is a *decomposition* of $P$ if their union is $P$ and all $C_i$ are interior disjoint, i.e., $\{C_i\}$ must satisfy:

$$\mathrm{D}(P) = \{C_i \mid \cup_i C_i = P \textit{ and } \forall_{i \neq j} C_i^\circ \cap C_j^\circ = \emptyset\} \ , \tag{1}$$

where $C^\circ$ is the open set of $C$.

A component $C$ is $\tau$-*approximate* convex if $C$ has *concavity* less than or equal to a tunable variable $\tau$. We use concave($C$) to denote the concavity measurement of $C$. Therefore, the $\tau$-*approximate* convex decomposition of $P$ is:

$$\mathrm{ACD}_\tau(P) = \{C_i \in \mathrm{D}(P) \mid \mathrm{concave}(C_i) \leq \tau\} \ . \tag{2}$$

We define the concavity of a vertex $x$ of $C$ as the distance from $x$ to the convex hull surface of $C$ and the concavity of $C$ as the maximum concavity of its vertices, i.e., $\mathrm{concave}(C) = \max_{x \in C}(\mathrm{concave}(x))$. ACD iteratively identifies and resolves concave features with maximum concavity. Figure 3 shows an example of this process. We refer readers to [30, 29] for details regarding ACD.

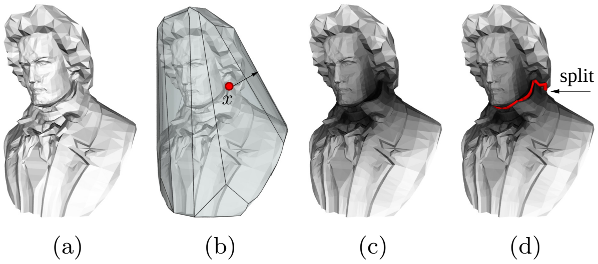



Figure 3: (a) The input model. (b) The convex hull of the input model. The concavity of $x$ is measured as the distance from $x$ to the convex hull surface. (c) The shading of the model represents concavity, i.e., darker areas have higher concavity. (d) The model is decomposed by partitioning at the high concavity region (indicated by an arrow).

**The Principal Axis**. Let $X$ be a set of points and $\ell$ be a line. We define $\mathrm{dist}(X, \ell)$, the distance from $X$ to $\ell$, as $\sum_X \mathrm{dist}(x, \ell)$, where $x \in X$. Then, the *principal axis* (PA) of a set of points $X$ is a line $\ell$ such that distance $\sum_X \mathrm{dist}(x, \ell)$ is minimized over all possible lines $\kappa \neq \ell$.

# 4 Framework

We propose a framework that simultaneously performs shape decomposition and skeleton extraction. For a given polyhedron $P$, **S**imultaneous **S**hape decomposition and **S**keleton extraction (SSS) (see Algorithm 4.1) constructs a skeleton for the model from (local) skeletons extracted from each component of a decomposition, evaluates the extracted skeleton components, and continues *refining* the decomposition and the associated skeleton components until the quality of the skeleton for each component is satisfactory, e.g., the error estimation of the skeleton for the respective component is smaller than a tunable threshold $\tau$.

There are three important sub-routines that are required by Algorithm 4.1: *Ext_Skeleton*($P$), which extracts a skeleton from a component $P$, *Error*($P, S$), which estimates the quality of the extracted skeleton, and *Decompose*($P$), which separates $P$ into sub-components when the extracted skeleton is not acceptable. We

**Algorithm 4.1** SSS($P$)

$S = Ext\_Skeleton(P)$
**if** $Error(P, S) \leq \tau$ **then**
    Report $S$ as $P$'s skeleton and report $P$ as a component
**else**
    $\{C_i\} = Decompose(P)$
    **For** each $C \in \{C_i\}$ **do** return SSS($C$)

discuss methods for skeleton extraction $Ext\_Skeleton(P)$ in Section 4.1, and methods for quality measurement $Error(P, S)$ in Section 4.2. Recall that our choice of the $Decompose(P)$ sub-routine is approximate convex decomposition.

## 4.1 Extracting Skeletons

In this section, we discuss two simple methods to extract a (*local*) skeleton from a component of a decomposition. These local skeletons can be connected to form a global skeleton of the input model. The centroid method, is very simplistic but can result in skeletons that do not represent the shape of the object. The second method, based on the principal axis of a component, is slightly more expensive to compute, but leads to improved skeletons in some cases.

**Using Centroids**. One of the easiest ways to construct a skeleton for a component $C$ (in a decomposition) is to connect the centroids of the openings, called *opening centroids*, on $\partial C$ to the centroid of $C$. These openings are generated when a component is split into sub-components during the decomposition process,

Several similar methods for extracting skeletons have been proposed [28, 23]. Although this approach is simple and generates fairly good results one of the major drawbacks of this type of skeleton is its inability to represent some types of shapes. For example, the skeleton of a cross-like model in Figure 4 extracted using its centroids is only a line segment instead of two crossing line segments. The method described next attempts to address this problem.


(using centroids) (using the principal axis)

Figure 4: This example shows a problem that arises when skeletonization is based only on the centroids. Points $b$ and $d$ are the centers of the openings and $a$, $c$ and $e$ are the centers of the components $P_1$, $P_2$ and $P_3$, respectively. This problem can be addressed using the principal axis.

**Using the Principal Axis**. In this method, we extract a skeleton from a component $C$ (in a decomposition) using the principal axis of the convex hull $H_C$ of $C$. Instead of connecting the centroids of $C$'s openings to the center of mass of $C$, we connect these centroids to the principal axis enclosed in $H_C$. Figure 5 shows an example of skeletons constructed in this manner.

Let PA($H_C$) be a line through the center of mass of $H_C$. parallel to the principal axis of $H_C$. Our method connects an opening centroid to one of the $k$ points on PA($H_C$) $\cap$ $H_C$. These $k$ points, denoted by $\mathcal{P}$, evenly subdivide PA($H_C$)$\cap H_C$ into $k+1$ line segments. The selection of the value of $k$ is based on the desired minimum skeleton link length. Let $\mathcal{P}' \subset \mathcal{P}$ be a set of points to which the opening centroids connect. Figure 5 illustrates $\mathcal{P}$ and $\mathcal{P}'$ with circles along PA($H_C$). Then, the final skeleton $S$ of $C$ contains line segments that connect the opening centroids to $\mathcal{P}'$ and line segments that connect the $\mathcal{P}'$.

To minimize the chance of getting a long skeleton with many joints, we match the opening centroids to $\mathcal{P}$ so that the cardinality of $\mathcal{P}'$ and the distances from the opening centroids to $\mathcal{P}'$ are minimized. We solve this optimization matching problem using dynamic programming. Details of how we find the optimal solution are discussed in Appendix A.

In cases where all the points in $\mathcal{P}'$ lie only on one side of the center of mass $c$ of $H_C$, e.g., $\mathcal{P}'$ in Figure 5(b), line segments that connect to the points in $\mathcal{P}'$ are not enough to represent the entire component. In such cases, the skeleton will connect $\mathcal{P}'$ with the end point of $\mathcal{P}$ on the other side of the center of mass $c$. Similarly, when $\mathcal{P}'$ contains only $c$, the skeleton will connect $c$ with the end points of $\mathcal{P}$ on both sides of $c$, e.g., the skeleton of the component $P_1$ in Figure 4 (using the principal axis).

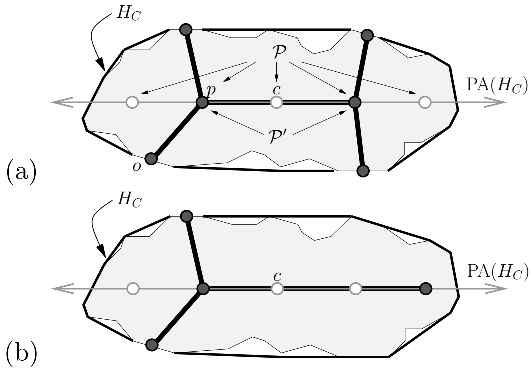



Figure 5: Using the principal axis of the convex hull $H_C$ to extract a skeleton from a component. Skeletons are shown in dark thick lines and skeletal joints are shown in dark circles and $c$ denotes the center of mass of $H_C$. (a) Opening centroids are connected to both sides of $c$. (b) Opening centroids are connected to only one side of $c$.

Figure 6 shows three skeletons: two extracted skeletons using the centroid and the principal axis methods, and one skeleton manually generated by a professional animator. One can see that the skeleton extracted using the principal axis is topologically more similar to the animator generated skeleton than the skeleton generated using the centroid method. In Section 5, we analyze the similarity of these skeletons using graph edit distance.

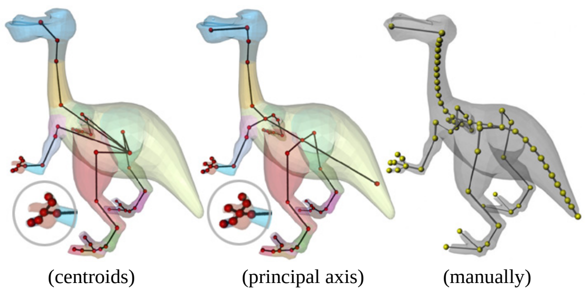



Figure 6: Notice the differences of these skeletons at the torso, the head, and the fingers.

## 4.2 Measuring Skeleton Quality

Although several criteria exist for measuring the quality of a skeleton, the general principles we adopt are that the skeleton should reside in the interior of the model and it should encode the "topology" of the model's shape. Thus, using these general criteria, our strategy of computing the quality of a skeleton $S$ is to compare $S$ with its associated component $C$. In this section, we propose three methods for measuring quality. This first method checks whether $S$ intersects $\partial C$ and the second method checks the topological representation of $S$ w.r.t. $C$. In the third method, we propose an adaptive measurement based on the volume of the component.

An important property of these three methods is that, as the decomposition becomes finer, the error of the skeleton becomes smaller. This property is justified in Appendix C. Figure 8 shows an example of extracted skeletons based on these three quality measurements.

**Checking penetration**. Our first method measures the quality of $S$ by checking whether $S$ intersects the component boundary $\partial C$. If so, the function $Error(C, S)$ returns a large number (larger than the tolerable value $\tau$). Otherwise, zero will be returned. The consequence is that $C$ will be decomposed if $\partial C \cap S \neq \emptyset$.

As seen in Figure 8, skeletonization using penetration detection stops evolving after a few iterations and does not produce skeletons that represent the dragon and the bird.

**Measuring centeredness**. In the second method, we measure the offsets of $S$ from the level sets of the geodesic distance map on $\partial C$. The value for each point in this map is the shortest distance to its closest opening of $C$. Ideally, a skeleton should pass through all connected components in all level sets. Therefore, this measurement method simply checks the number of times that $S$ does not do so. An example of this measurement is shown in Figure 7.

Let $L_C$ be all the connected components in the level sets of $C$. We define the error of a skeleton $S$ as:

$$Err(C, S) = \frac{\sum_{l_c \in L_C} f(l_c, S)}{|L_C|} \ , \tag{3}$$

where $f(l_c, S)$ returns 0 if $S$ intersects component $l_c$, and 1 otherwise, and $|L_C|$ is the total number of the connected components in $C$. Details of how we compute the level sets and $f(l_c, S)$ are discussed in Appendix B.

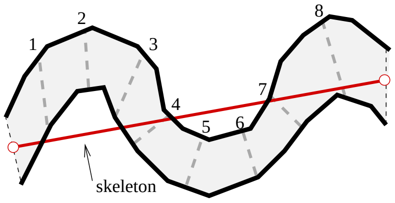



Figure 7: The error measurement for this skeleton, which intersects level sets 4, 7 and 8, is $\frac{5}{8}$.

As seen in Figure 8, skeletonization using the centeredness measurement captures the shape of the dragon and the bird better then simply using penetration detection, but it has a problem of over segmenting the tail of the bird and does not produce accurate skeletons in the dragon's and the bird's feet.

**Measuring convexity**. Our idea for the last quality measurement comes from the observation that in many cases the significance of a feature depends on its volumetric proportion to its "base". For example, a 5 cm stick attached to a ball with 5 cm radius is a more significant feature than a 5 cm stick attached to a ball with 5 km radius. This intuition can be captured by the concept of the *convexity* of a component $C$ defined as $\text{convexity}(C) = \frac{\text{vol}(C)}{\text{vol}(H_C)}$, where $\text{vol}(X)$ is the volume of a set $X$. Thus, we can define the error measurement as:

$$Err(C, S) = 1 - \text{convexity}(C) \ . \tag{4}$$

Assume that the skeleton $S$ is a good representation of the convex hull $H_C$. Then, a smaller difference between $H_C$ and $C$ means that $S$ is a better representation of $C$. Thus, although the skeleton $S$ is not included in Equation 4, $S$ is implicitly considered in terms of $H_C$.

As seen in Figure 8, using convexity produces the most realistic skeleton that captures the overall shape of the dragon and the bird and also identifies the detailed features of their feet.

## 4.3 Putting it All Together

Algorithm 4.2 shows a fleshed-out version of the proposed simultaneous shape decomposition and skeletonization framework.

Here we suggest using the principal axis, convexity and approximate convex decomposition for local skeleton extraction, quality measurement and partitioning, respectively. Algorithm 4.2 is used for all the experiments in Section 5. We would like to emphasize that the choice of these methods is made based on our own experience.

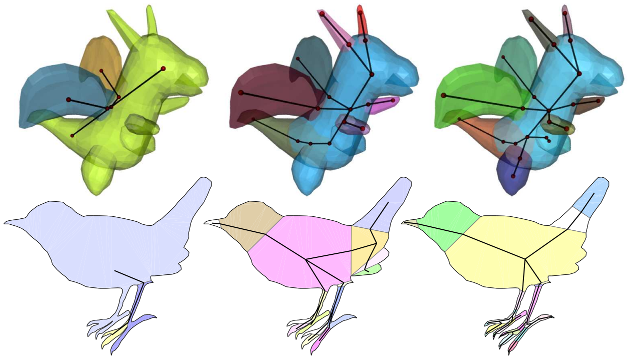



Figure 8: Final skeletons of a dragon polyhedron and a bird polygon extracted using different quality estimation functions: checking penetration, measuring centeredness, and measuring convexity. The maximum tolerable errors for centeredness and convexity are 0.2 and 0.3, respectively.

**Algorithm 4.2** $SSS_{ACD}(P)$

1: Compute a skeleton $S$ from $P$ using the *Principal Axis* of $H_P$.
2: Estimate the quality of $S$ using *convexity*.
3: **if** $S$ is acceptable **then**
4: Report $S$ as $P$'s skeleton and report $P$ as a component.
5: **else**
6: $\{C_i\} = ACD(P)$.
7: **For** each $C \in \{C_i\}$ **do** return $SSS_{ACD}(C)$

The framework is not restricted to these selected sub-routines, which can be replaced by other methods to fit particular needs of an application.

# 5 Implementation and Results

The experiments in this section are used to demonstrate the *efficiency*, the *robustness*, and several *applications* of the proposed method. The method was implemented in C++ and all these experiments are performed on a Pentium 2.0 GHz CPU with 512 Mb RAM. Seventeen decompositions and their associated skeletons are shown in Figures 8 to 13 and in Tables 1 and 2.

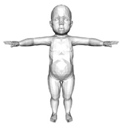
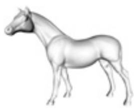
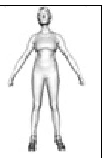
**Efficiency**. A summary of the studied models, which include several game characters, a high genus model, and two scanned models, and the skeletonization and decomposition time of these models is shown in Table 1. Table 1 shows that the processing time of SSS depends not only on the size of the model but also on the complexity of the shape. For example, even though the model in Figure 9 has the fewest triangles, its large genus (18) increases the processing time. In general, our proposed SSS method can handle models with thousands of triangles in less than a half a minute and scales well for models with tens or hundreds of thousands of triangles.

We further show that SSS is efficient by comparing our results to two recently proposed shape decomposition and skeletonization methods that have been shown to produce very promising results; see Figures 10 and 11, respectively. In both experiments, SSS generates results similar to those results reported previously but SSS can produce the shape decomposition and the skeletons about 30 times and 5 times faster than those methods reported in [23] and [47], respectively. We note that there are no well-accepted criteria to compare the quality of these decompositions and skeletons quantitatively, and therefore we do not intend to claim that our results are necessarily better.

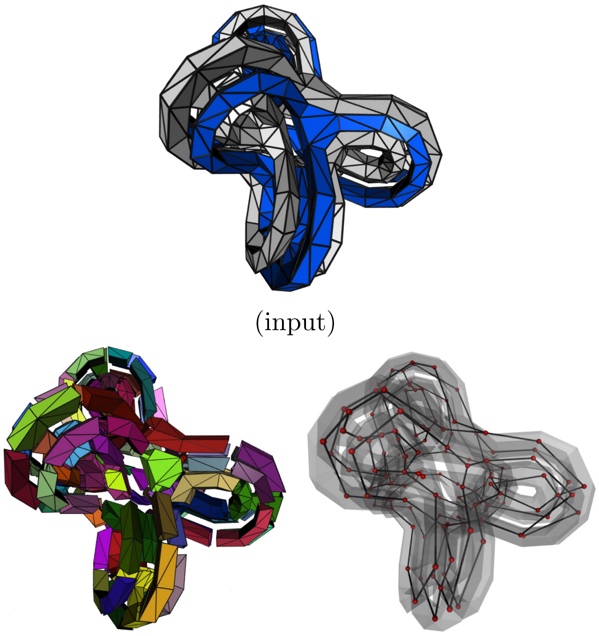
**Robustness**. In this set of experiments, we show that SSS is robust under perturbation and deformation, meaning that the shape decompositions and skeletons remain approximately the same after the input models are perturbed and deformed. The results are shown in Table 2.

Although there are no well accepted criteria to measure the differences among decompositions, we can measure

Table 1: Experimental Results

| Model |  Figure 9 |  Figure 12 |  Table 2 |  Figure 6 |  Figure 11 |  Figure 12 |  Table 2 |  Figure 2 |  Table 2 |
|---|---|---|---|---|---|---|---|---|---|
| Size | 1,984 | 3,392 | 5,660 | 6,564 | 8,276 | 11,180 | 39,694 | 48,312 | 243,442 |
| Time | 15.6 | 2.6 | 1.7 | 1.5 | 8.8 | 3.4 | 19.4 | 30.1 | 73.3 |

Size is measured as the number of the triangles of each model and the processing time is measured in seconds.



(input)

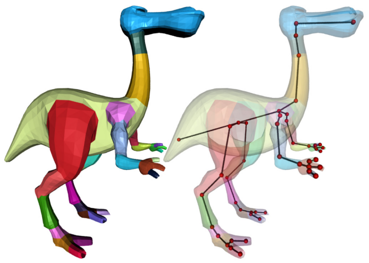
(decomposition) (skeleton)

Figure 9: This figure shows the decomposition and the skeleton of a model with 18 handles.

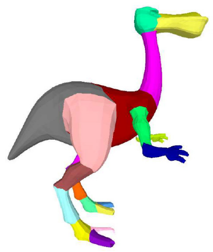
the similarity of these skeletons, e.g., using graph edit distance [11] which computes the cost of operations (i.e., inserting/removing vertices or edges) needed to convert one graph to another. In this paper, we simply associate one unit of cost with each operation.

We measure two types of distances, denoted as $D_O$ and $D_O^2$. $D_O$ is the graph edit distance from a skeleton to the skeleton extracted from the original mesh. Because removing or inserting a degree-two node does not change the topology of a graph, we are also interested in the distance, denoted as $D_O^2$, that does not count operations that create and remove degree-two nodes. Table 2 shows that $D_O$ remains small for both perturbed and deformed models and $D_O^2$ is zero for all cases.

**Applications**. The extracted skeleton can be readily used to create animations. We demonstrate this advantage by re-targeting motion captured data to the skeletons extracted using our method. In Figure 12, we show a sequence of images obtained from a skeleton-based boxing animation of a baby and a robot using motion data captured from an adult male. Note that the baby and the robot models have different body proportions and rest poses. Due to the limited space in this paper, other animations, including walking and pushing a box, are only included in the submitted supplementary materials. We use the motion captured data instead of a hand-made animation to show that the extracted skeletons are robust enough to be used by arbitrarily selected motions other

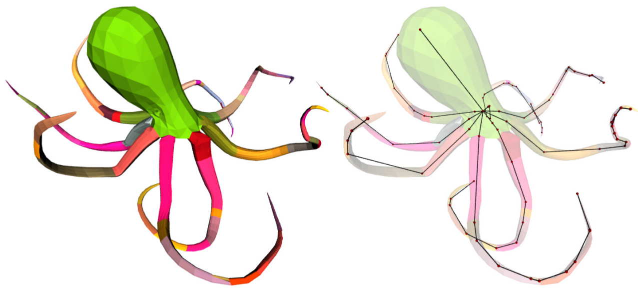


(SSS)

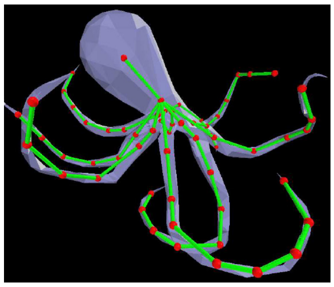


(Katz and Tal [23])

Figure 10: The decomposition with 0.7 convexity and the associated skeleton of the dino-pet model (with 6,564 triangles) are computed in 1.5 seconds whereas Katz and Tal's approach takes 57 seconds (on a P4 1.5 GHz CPU with 512 Mb RAM).



(SSS)



(Wu et al. [47])

Figure 11: The decomposition with 0.7 convexity and the associated skeleton of the octopus model (with 8,276 triangles) are computed in 8.8 seconds whereas Wu et al.'s approach takes 53 seconds (on a P4 1.5 GHz CPU with 512 Mb RAM) using a simplified version of this model (with 2,000 triangles).

than a carefully designed motion. The motions, i.e., joints angles, are manually copied from the captured data to the skeletal joints.

The extracted skeletons can also help to plan motion, e.g., for navigating in the human colon or removing a mechanical part from an airplane engine. Sampling-based motion planners have been shown to solve difficult motion planning problems; see a survey in [6]. These methods approximate the free configuration space (C-space) of a movable object by sampling and connecting random configurations to form a graph (or a tree). However, they also have several technical issues limiting their success on some important types of problems, such as the difficulty of finding paths that are required to pass through narrow passages [46]. Using sampling biased toward the joints of the extracted skeleton, we can alleviate this so called "narrow passage" problem. Figure 13 shows that the graph constructed using the skeleton-based sampling can better represent the free C-space than using the uniform sampling [24] with the same number of samples. This is because more of the skeleton-based sampling samples are placed in the narrower regions. In addition, the connections between the samples in these narrow regions can be made easily because the components of the decomposition are nearly convex.

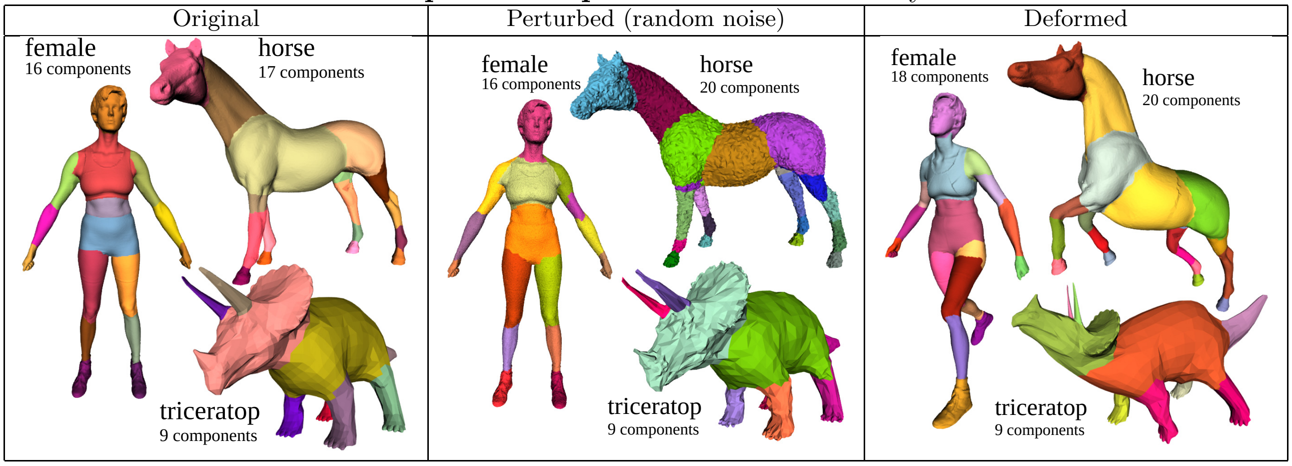
# 6 Discussion and Conclusion

In this paper, we propose a framework that simultaneously generates shape decompositions and skeletons. This framework is inspired by the observation that both operations share many common properties and applications but are generally considered as independent processes. This framework extracts the skeleton from the components in a decomposition and evaluates the skeleton by comparing it to the components. The process of simultaneous shape decomposition and skeletonization iterates until the quality of the skeleton becomes satisfactory.

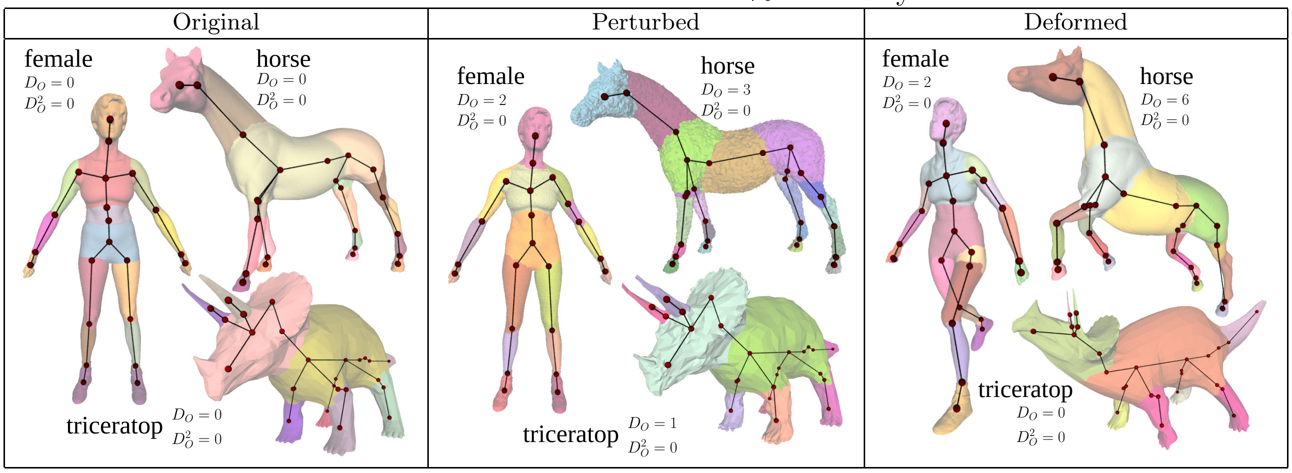
We studied two simple skeleton extraction methods, using the centroids and the principal axis, and three quality evaluation measurements, that compute penetration, centeredness and convexity, respectively. In the experiments,

Table 2: Robustness tests using perturbed and skeletal deformed meshes. $D_O$ is the graph edit distance between a skeleton extracted from a perturbed or deformed mesh and a skeleton extracted from the original mesh. $D_O^2$ is $D_O$ without counting operations on degree-2 nodes (which do not change the topology of the skeleton).

**Shape Decomposition.** 70% convexity




**Extracted Skeletons.** 70% convexity




we demonstrate that the proposed framework is efficient, robust under perturbation and deformation, and can readily be used, e.g., to generate animations and plan motion.

There are several ways to extend the current work. First, there is a need to establish a systematic framework for comparing qualities of shape decomposition and skeletons using more quantitative measuring methods and benchmarks. Although the proposed quality measurements are based on a general idea of what a good skeleton should be, more studies are needed to investigate application-specific measurement criteria that should produce better and more "comparable" results. Second, not all models, such as a bowl, can have reasonable 1D skeletons. We are interested in using the same framework to extract the approximated medial axis from the components in a decomposition based on the idea that it is easier to extract the medial axis from a convex object than from a non-convex object.

# References


[1] Nina Amenta, Sunghee Choi, Maria E. Jump, Ravi Krishna Kolluri, and Thomas Wahl. Finding alpha-helices in skeletons. Technical report, Dept. of Computer Science, University of Texas at Austin, 2002.

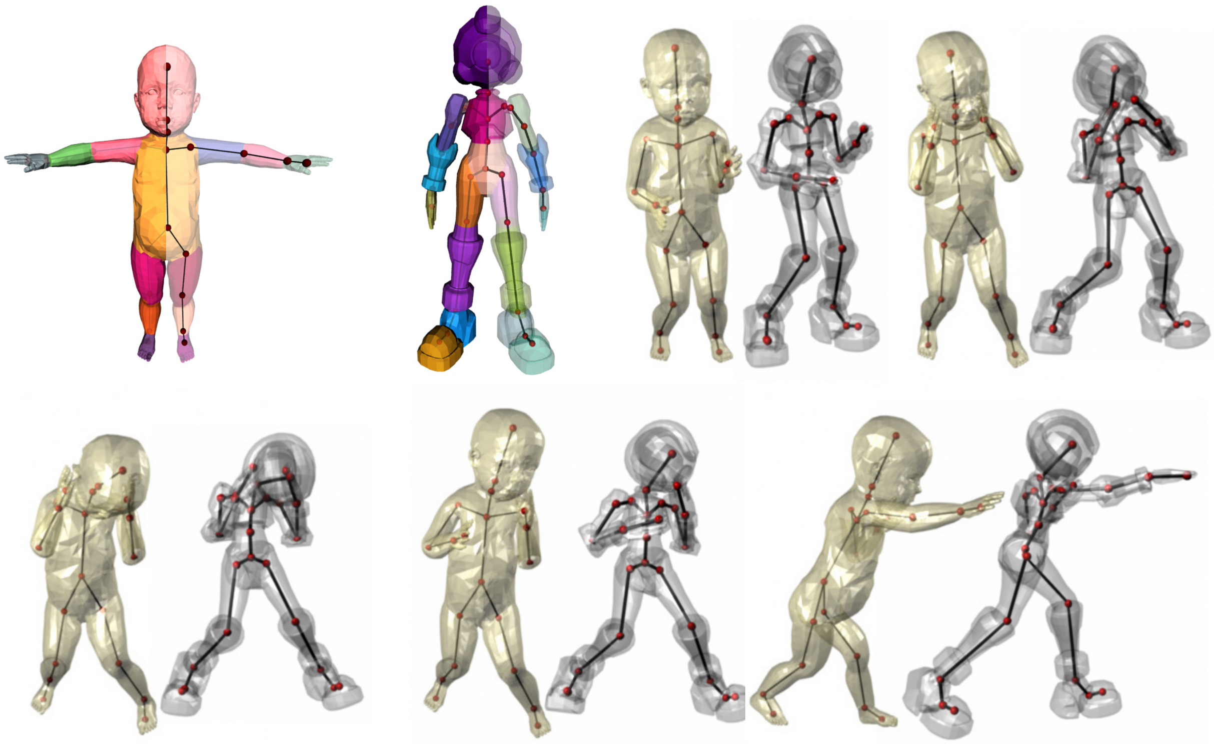


Figure 12: An animation sequence obtained from applying the boxing motion capture data to the extracted skeletons from a baby model and a robot model. The motion capture data (action number 13_17) are downloaded from the CMU Graphics Lab motion capture database. The first two figures in the sequence are the shape decompositions and the skeletons of the baby and the robot. Note that not all joint motions from the data are used because the extracted skeletons have fewer joints.


[2] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2-3):127–153, 2001.

[3] D. Attali, P. Bertolino, and A. Montanvert. Using polyballs to approximate shapes and skeletons. In *Proceedings of International Conference on Pattern Recognition (ICPR'94)*, pages 626–628, 1994.

[4] D. Attali and J.-O. Lachaud. Delaunay conforming iso-surface; skeleton extraction and noise removal. *Computational Geometry: Theory and Applications*, 19(2-3):175–189, 2001.

[5] Marco Attene, Silvia Biasotti, and Michela Spagnuolo. Re-meshing techniques for topological analysis. In *Proc. of the Shape Modeling International (SMI'01)*, pages 142–151, May 2001.

[6] J. Barraquand, L.E. Kavrakiand J.C. Latombe, T.Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *Int. J. of Rob. Res*, 16(6):759–774, 1997.

[7] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychol. Rev.*, 94(2):115–147, 1987.

[8] Ingmar Bitter, Arie E. Kaufman, and Mie Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[9] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

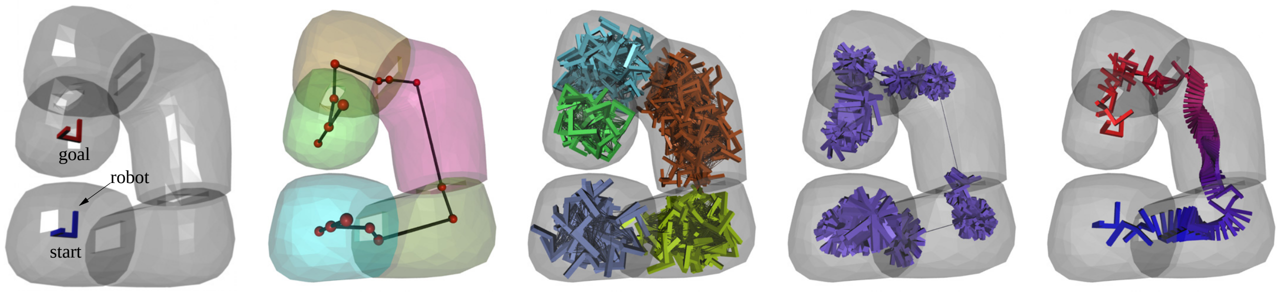
[10] Gareth Bradshaw and Carol O'Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *Proceedings of the ACM SIGGRAPH symposium on Computer animation*, pages 33–40. ACM Press, 2002.

[11] H. Bunke and A. Kandel. Mean and maximum common subgraph of two graphs. *Pattern Recogn. Lett.*, 21(2):163–168, 2000.

[12] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popovic. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics*, 21(3):586–593, 2002.

[13] Jen-Hui Chuang, Chi-Hao Tsai, and Min-Chi Ko. Skeletonization of three-dimensional object using generalized potential field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1241–1251, 2000.

[14] Tim Culver, John Keyser, and Dinesh Manocha. Exact computation of the medial axis of a polyhedron. *Comput. Aided Geom. Des.*, 21(1):65–98, 2004.




(a) (b) (c) (d) (e)

Figure 13: Narrow passage motion planning. (a) A difficult motion planning problem in which the robot is required to pass through four narrow windows to move from the start configuration (bottom) to the goal configuration (top). (b) A shape decomposition and a skeleton of the environment using the proposed method. In (c), uniform sampling results in a poor representation of the robot's free C-space. Uniform sampling samples 350 uniformly distributed collision-free configurations and connects these samples into a graph. In this example, there are five connected components in the graph. In (d), the skeleton-based sampling results in a better representation of the robot's free C-space. The skeleton-based sampling samples 350 collision-free configurations around the joints of the skeleton and connects these samples into a graph. In this example, the graph has one single connected component. (e) A collision-free path found by connecting the start and the goal configurations to the graph generated using the skeleton-based sampling.


[15] C. M. Cyr and B. B. Kimia. 3d object recognition using shape similarity-based aspect graph. In *ICCV'01*, 2001.

[16] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In *Proc. Workshop on Algorithms and Data Structures*, pages 25–36, 2003.

[17] Tamal K. Dey and Wulue Zhao. Approximate medial axis as a voronoi subcomplex. In *ACM Symposium on Solid Modeling and Applications*, pages 356–366, 2002.

[18] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, 2004.

[19] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, 2001.

[20] D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[21] D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63:29–78, 1997.

[22] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics (TOG)*, 15(3):179–210, 1996.

[23] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961, 2003.

[24] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[25] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Intelligent mesh scissoring using 3d snakes. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04)*, pages 279–287, October 2004.

[26] Tsai-Yen Li, Jyh-Ming Lien, Shih-Yen Chiu, and Tzong-Hann Yu. Automatically generating virtual guided tours. In *Proc. IEEE Computer Animation (CA)*, pages 99–106, May 1999.

[27] Xuetao Li, Tong Wing Toon, and Zhiyong Huang. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 35–42, 2001.

[28] J.-M. Lien and N. M. Amato. Approximate convex decomposition. Technical Report TR03-001, Parasol Lab, Dept. of Computer Science, Texas A&M University, Jan 2003.

[29] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. Technical Report TR05-001, Parasol Lab, Dept. of Computer Science, Texas A&M University, Jan 2005.

[30] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. In *Proc. 20th Annual ACM Symp. Computat. Geom. (SoCG)*, pages 17–26, June 2004.

[31] Rong Liu and Hao Zhang. Segmentation of 3d meshes through spectral clustering. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04)*, pages 298–305, October 2004.

[32] Yong Lu, Rajit Gadh, and Timothy J. Tautges. Volume decomposition and feature recognition for hexahedral mesh generation. In *Proc. 8th International Meshing Roundtable*, pages 269–280, October 1999.

[33] Alan P. Mangan and Ross T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[34] R.L. Ogniewicz and O. Kubler. Hierarchic voronoi skeletons. *Pattern Recognition*, 28(3):343–359, 1995.

[35] D. L. Page, A. F. Koschan, and M. A. Abidi. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *Proceedings of the 2003 Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 27–32, 2003.

[36] Roman M. Palenichka, Marek B. Zaremba, and Universite du Quebec. Multi-scale model-based skeletonization of object shapes using self-organizing maps. In *16 th International Conference on Pattern Recognition (ICPR'02)*, pages 10143–10147, 2002.

[37] H. Rom and G. Medioni. Hierarchical decomposition and axial shape description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):973–981, 1993.

[38] Hillel Rom and G. Medioni. Part decomposition and description of 3d shapes. In *Proc. International Conference of Pattern Recognition*, pages 629–632, June 1994.

[39] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 146–155, 2003.

[40] D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Shape description by medial surface construction. *IEEE Trans. Visualizat. Comput. Graph.*, 2(1):62–72, March 1996.

[41] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Comput. Graph. Appl.*, 11:66–78, September 1991.

[42] Yoshihisa Shinagawa and Tosiyasu L. Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11(6):44–51, 1991.

[43] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. In *ICCV*, pages 222–229, 1998.

[44] Marek Teichmann and Seth Teller. Assisted articulation of closed polygonal models. In *Proceedings of the Eurographics Workshop*, pages 254–254, 1998.

[45] A Verroust and F. Lazarus. Extracting skeletal curves from 3d scattered data. In *International Conference on Shape Modeling and Applications*, pages 194–201. IEEE Computer Society, 1999.

[46] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Motion planning for a rigid body using random networks on the medial axis of the free space. In *Proc. ACM Symp. on Computational Geometry (SoCG)*, pages 173–180, 1999.

[47] Fu-Che Wu, Wan-Chun Ma, Ping-Chou Liou, Rung-Huei Laing, and Ming Ouhyoung. Skeleton extraction of 3d objects with visible repulsive force. In *Computer Graphics Workshop 2003, Hua-Lien, Taiwan*, 2003.

[48] Kenong Wu and Martin D. Levine. 3d part segmentation using simulated electrical charge distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1223–1235, 1997.

[49] Geoff Wyvill and Chris Handley. The "thermodynamics" of shape. In *Proc. of the Shape Modeling International (SMI'01)*, pages 2–8, May 2001.

[50] Yong Zhou and Arthur W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.

# Appendix

# A Construct skeleton from the Principal Axis

Here, we show how a local skeleton can be computed using the principle axis. Our goal is to find a match $M : O \to \mathcal{P}$ from the opening centroids $O$ to the points $\mathcal{P}$ on the principle axis so that the total length of the match and the number of the matched points (joints) in $\mathcal{P}$ is minimized. We let the score function $F$ of a match $M$ be defined as

$$F(M) = s_1 \cdot |M| + s_2 \cdot J(M) \ , \tag{5}$$

where $|M|$ and $J(M)$ are the length and joint size of match $M$, and $s_1$ and $s_2$ are user specified scalars. In this paper, $s_1$ and $s_2$ are constantly set as 10 and 1, resp. A brute force approach to find an optimal solution will take $O(|\mathcal{P}|^{|O|})$ time, where $|\mathcal{P}|$ and $|O|$ is the number of vertices in $\mathcal{P}$ and $O$, respectively. This exponential time complexity is in general impractical for most applications.

The main idea of finding the optimal match is to group opening centroids $O$ and each group will connect to a point in $\mathcal{P}$. After knowing how $O$ is grouped, it takes $O(|\mathcal{P}||O|)$ time to find solution.

Grouping $O$ can be done using dynamic programming. An observation enables us to group $O$ is that two centroids are likely to be grouped when their closest points in $\mathcal{P}$ are close. Thus, we first sort $O$ according to their closest points of $\mathcal{P}$ and then group sorted $O$. A dynamic programming is shown in Algorithm A.1 to group $O$. In Algorithm A.1, we use $G[i, j]$ to denote the optimal solution for the sub-problem $\{O_i, \cdots, O_j\}$ and use $< G_i G_j >$ and $G_i G_j$ to denote the joint of two groups $G_i$ and $G_j$ with and without merging $G_i$ and $G_j$ to one group.

**Algorithm A.1** Optimal Matching($O$, $\mathcal{P}$)

**for** $i \in \{1, \cdots, |O|\}$ **do**
  $G[i, i] = O_i$
**for** $l \in \{2, \cdots, |O|\}$ **do**
  **for** $i \in \{1, \cdots, |O| - l + 1\}$ **do**
    $j = i + l - 1$
    $G[i, j] =< O_i \cdots O_j >$
    $score = F(G[i, j], \mathcal{P})$ {$F$ is defined in Eqn. 5}
    **for** $k \in \{i \cdots, j - 1\}$ **do**
      $s = F(G[i, k]G[k + 1, j], \mathcal{P})$
      **if** $s_1 < score$ **then**
        $G[i, j] = G[i, k]G[k + 1, j]$
        $score = s_1$

## B Compute level sets and centeredness

A level set of a component $C$ in a decomposition is a set of points on the surface $\partial C$ of the component with the same geodesic distance to the closest opening of $C$. A connected component in a level set is a list of connected edges, which usually forms a loop on $\partial C$. A level set can have one or multiple connected component(s).

These level sets can be computed, similar to the construction process of a Reeb graph [42], by flooding the entire $\partial C$ from the boundaries of the openings of $C$. In each iteration of this flooding process, the wavefronts will propagate from the visited vertices to unvisited vertices via incident edges.

To compute centeredness, we need to know how well a skeleton $S$ intersect the level sets of $C$, i.e., we need the function $f(l_c, S)$ use in Eqn 3, which returns zero if $S$ intersect the level set $l_c$. The function $f(l_c, S)$ can be implemented by simply checking the intersection between each line segment of $S$ and the triangulation of $l_c$.

## C ACD increases skeleton quality

In this section, we show that the error measurements of a skeleton described in Section 4.2, i.e., penetration, centeredness, and convexity, decreases as the input model get decomposed. This is a critical property, which allows the SSS framework to terminate.

**Lemma C.1.** *Let $S$ be the skeleton of a polyhedron $P$ and let $S'$ be the skeleton of the components of the* ACD *of $P$. The error estimation of $S'$ must be smaller than the error estimation of $S$ measured using penetration, centeredness, and convexity defined in Section 4.2.*

*Proof.* We show that all error measurements become zero if the input model is convex. For penetration, because any two points inside the convex object must not intersect its boundary, a skeleton will never penetrate the object. For the same reason, the skeleton must not be 'outside' of any level set of a convex component. Finally, because the convexity of a convex object is one, its error must be zero.

Since ACD decomposes $P$ into more convex components than $P$ is after each iteration, the error measurements of $S'$ will be closer to zero than $S$ for all three types of measurements. □