STAG: Smart Tools & Apps for Graphics (2015)
Silvia Biasotti and Marco Tarini and Andrea Giachetti (Editors)

Skeleton Lab: an Interactive Tool to Create, Edit, and Repair Curve-Skeletons

S.Barbieri[1], P.Meloni[1], F.Usai[1] & R.Scateni[1]

[1]Universitá degli studi di Cagliari, Italy

Abstract

Curve-skeletons are well known shape descriptors, able to encode topological and structural information of a shape. The range of applications in which they are used comprises, to name a few, computer animation, shape matching, modelling and remeshing. Different tools for automatically extracting the curve-skeleton for a given input mesh are currently available, as well as inverse skeletonization tools, where a user-defined skeleton is taken as input in order to build a mesh that reflects the encoded structure. Although their use is broad, an automatically extracted curve-skeleton is usually not well-suited for the next pipeline step in which they will be used. We present a tool for creating, editing and repairing curve-skeletons whose aim is to allow users to obtain, within minutes, curve-skeletons that are tailored for their specific task.

1. Introduction

A curve-skeleton is a compact mono-dimensional representation able to encode meaningful information about both volume and topology of a shape. While the skeleton of a two dimensional shape is defined as its *Medial Axis Transform* [Blu67], which is the locus of centres of its maximal inscribed discs, this definition leads, for three dimensional shapes, to a collection of connected curves and sheets that are impractical to use in real world applications.

Constraining the skeleton of a 3D shape to be a one-dimensional structure results in a simpler and more intuitive representation, that is also easy and natural to manipulate. As of today, however, there is no unique, precise and universally accepted definition of curve-skeletons, hence different approaches for its computation have been proposed, each obtaining results with different features, characteristics and defects [Tag13].

The range of applications in which curve-skeletons are used is wide and comprises computer animation [BP07] , shape matching [HSKK01], modelling [BAS14], remeshing and quad layout extraction [ULP*ss], polycube [LZLW15] and hexahedral mesh construction [ZBG*07]. Since each of these applications requires skeletons with different properties, further processing is often required in order to obtain optimal results in the pipeline in which they will be used. Moreover it is often difficult to define algorithms to process a skeletal structure in order to reflect the required features, due to the semantic nature of the information it conveys.

Tools for interactive creation of curve-skeletons already exist also in the field of medical imaging [AJ09]). Their aim is to create from scratch curve-skeletons that can also be used as input for an inverse skeletonization algorithm, so they allow only basic operations which are not sufficient for editing an automatically extracted skeleton.

In [ULP*ss] an algorithm for computing a coarse quad-layout starting from a shape and its curve skeleton was proposed. We found that using automatically extracted skeletons often brought to sub-optimal results even using some simplification strategies. We then developed the tool we present in this paper, that have shown to be easy-to-use and practical for interactively editing curve-skeletons. It allowed us to obtain optimal results within our quad-layout computation method and we thought it can allow other researchers or practitioners to obtain, within minutes, curve-skeletons that are tailored for their specific task.

The rest of the paper is organized as follows: in Section 2 we will give a brief overview of the different algorithms for computing curve-skeletons at the state-of-the-art, interactive tools for curve-skeleton handcrafting will be discussed as well; in Section 3 we will discuss the limitation of using the current skeletonization approaches in real world applications and the reasons why the tool we are presenting in this paper could be a useful resource; in Section 4 we will give an

overview of our tool, describing what one can do with it; in Section 5 we will present an example of a specific use case, to give the feeling of the usage; in section 6 we will draw our conclusions, consider the limitations and explain what can be done in the future to improve the work.

The project page with a link to the repository containing the entire source code of the presented tool is available at http://francescousai.info/skel_lab

2. Related work

Several automatic skeletonization methods have been presented in the last two decades. A common categorisation of these methods is based on their input, which can usually be a surface mesh [JST15], a voxel grid [SJT14] or a point cloud [TZCO09]. We refer to [Tag13] for a recent survey. Nowadays a universally accepted definition of what a curve skeleton is, is still lacking. The *medial geodesic skeleton* defined in [DS06] is the only exception, however, while the theoretical definition is precise and sound, the implementation of the skeletonization algorithm provided by the authors presents some defects. One important issue regards its long computational times, since it requires computing the geodesic paths between each pair of points on the surface. There are also stability issues, due to the sensitivity of the medial axis to small perturbations of the surface, which brings to noisy skeletal paths and spurious branches, especially between nodes where the branching occurs.

Since different skeletonization approaches exist, it is difficult to compare the quality of the results of each method. Important information about the desirable properties of a curve-skeleton can be found in [CSM07].

The current trend in skeletonization is to use a contraction-based approach, which relies on contracting the 3D shape represented as a triangle mesh accordingly to its mean curvature-flow until it collapses to a monodimensional object. We refer to [SJT14, SYJT13] for a qualitative comparison of the more representative methods based on this approach. A recent contraction-based method is [TAOZ12] which is a robust skeletonization algorithm whose resulting skeletons are topology-preserving, usually well centered and smoother than previous methods.

Notable exceptions to this trend are [LGS12, LS13, KJT13] which, rather than relying on the geometric properties of the input, operate emulating human perception, synthesising the curve-skeleton of a 3D object from the curve-skeletons of a set of its 2D silhouettes defined from different points of view. This approach, while being less robust than [DS06, ATC*08, TAOZ12] presents heuristics for collapsing spurious branches and closing loops whenever two terminal nodes have intersecting maximal balls, which are beneficial for the resulting structure.

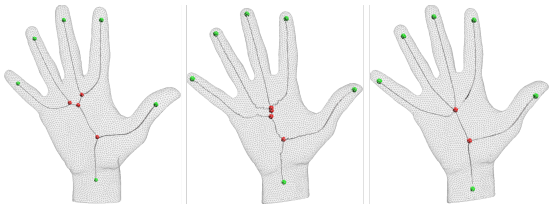


Figure 1: *A direct comparison of three automatic skeletonization methods. [TAOZ12] (left), [DS06] (center) and [LS13] (right) on the same hand model. One can notice that the terminal nodes are the same in all the three skeletonization, while the internal nodes are different in number and position. It is difficult to tell which one is the most correct.*

2.1. Skeleton editing tools

The term *Skeletonization* denotes the process of computing a curve-skeleton from a 3D shape, similarly we can define the *Inverse Skeletonization* as the process of building a 3D shape from which the input skeleton could have been extracted.

In recent years, inverse skeletonization became a quite active research field, giving rise to different works [BMW12, JLW10, HBC*10]. Commercial solutions for building 3D shapes, especially creature-like, were also developed and made publicly available [Aut, ZBr].

Almost all the aforementioned methods or commercial software, comprised an interactive skeleton creation tool. The aim of these tools is to quickly create 3D models that will have to be further refined, hence they are designed to enable the user to do only simple operations such as adding new nodes, move them around and changing their radius. ZSpheres [ZBr], which is embedded in a complete modelling tool, is more complete and complex than the others aforementioned, but on the curve-skeleton editing side it provides the same basic operations. These basic operations are not enough when a user needs to easily process a skeleton that has been computed with one of the presented automatic methods. Our tool addresses this need allowing to operate on a curve-skeleton at different scales with a set of operations that are not present in other tools and have shown to be very useful in practice.

3. Motivation

From a practical point of view all the methods presented in the previous section have specific properties and issues making the resulting skeletons difficult to use, without further processing, in the pipelines of the applicative fields presented in the introduction. An important aspect is that each of the presented methods requires some input parameters to be set in order to fine-tune how the algorithm behaves, which usually regulates its quality and especially its ability to catch small scale features. This brought us to make the following points:

- The computation of a good curve-skeleton is not a *one-click operation* since it requires a, sometime complex, parameter setting that is potentially tightly coupled to both the specific application and the specific shape; this usually leads to perform several attempts before obtaining the desired skeleton.
- Since most of the parameter setting is about detecting small scale features, it turns out that more sensitivity usually means more noise and spurious branches; on the other hand, less noise usually brings to the risk of not detecting all the meaningful features of the shape being processed.
- Given the semantic nature of the information encoded in a curve-skeleton, it results difficult, from an application agnostic point of view, to define what a meaningful feature is and how much noise can be tolerated in order to capture it.

As can be seen in Fig. 1 different skeletonization methods give very different results even on models of moderate complexity. It is not easy to say which of the three skeletons is correct, since that all the considered methods have different approaches. Furthermore we can say that rather than evaluating if an extracted curve-skeleton is correct or not, it is worth to evaluate if it is suitable or not for the purpose and if it brings optimal results or not. We have observed that is more practical to rely on a pipeline which involves an automatic extraction method with a limited choice of standard parameter values, followed by a manual editing stage with an interactive tool like ours, rather than relying on a complete automatic pipeline in which small changes to the input parameters bring to different results in which a choice between the number of meaningful features preserved and the number of spurious features introduced is needed.

3.1. Applicative contexts

As we mentioned in the introduction we proposed in [ULP*ss] an algorithm for computing a coarse quad-layout starting from a shape and its curve skeleton. We found that optimal skeletons for this application are robust, connected and reliable [CSM07]. Strong homotopy is not required but its beneficial. Optimal results have been obtained when the skeleton had branches with a small number of nodes while still well approximating the shape, and the maximal balls of the endpoints of each branch did not intersect the other ones. In some cases, optimal results can be achieved with a little interactive editing of the automatically extracted skeleton collapsing spurious branches, handcrafting the missing ones and lowering the number of nodes retaining only the most representative ones.

Skeletons with small number of nodes are beneficial also in other application fields, such as defining kinematic skeletons, which are composed of very few nodes for each bone, that require to be carefully placed. In those applications an automatic simplification of a skeleton can potentially lead to kinematic skeletons that are not well suited for the purpose, since the nodes associated with articulations need to be preserved, but the presence of an articulation of the shape is a semantic feature that is difficult to capture algorithmically.

4. Skeleton Lab

Our tool, Skeleton Lab, allows the user both to create new skeletons and edit existing ones, hence it provides functionalities that other skeleton editors usually lack. This section will present the set of its distinguishing features, their purpose and some design choices.

We consider the skeleton as an attributed graph $G = (N, L)$ where N is the set of nodes and L the set of links between nodes. Each node $N_i \in N$ has a few attributes:

- Position in space.
- Radius of the associated maximal ball.
- Type of the node (the classification follows).
- List of its neighbouring nodes, since we do not explicitly store the links.

We classify the nodes of the skeleton as: *Joint nodes* (**Jn**) that have two incident arcs; *Leaf nodes* (**Ln**) that have only one incident arc; and *Branching nodes* (**Bn**) that have more than two incident arcs. We also allow to mark a **Jn** as an *Articulation* (**An**), that is a semantically meaningful **Jn**(e.g., a human ankle or elbow).

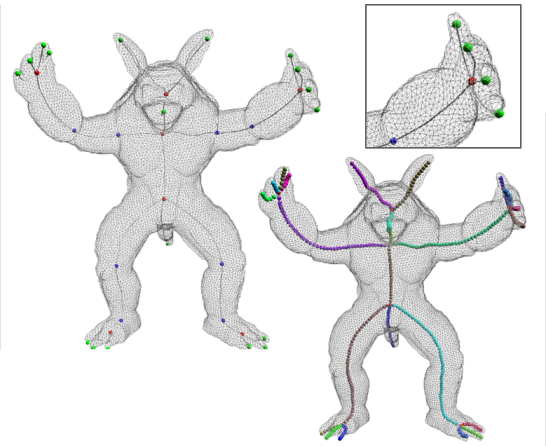


Figure 2: *A visual explanation of the terminology we use. In the left image **Bn**'s are red, **Ln**'s are green, and **An**'s are blue, while **Jn**'s are not shown. In the right image each **Branch** has a different color.*

With the term **Branch** we refer to a sequence of linked nodes starting and ending with either a **Bn** or a **Ln**. When a branch ends with a **Ln** we call it ***Ln**-ending*, while when the starting and the ending nodes are exactly the same, we have a *Looping-Branch*.

With our tool the user is allowed to visualise and manipulate a curve-skeleton and optionally visualise the triangle mesh it is connected to.

By design we have chosen to group functionalities into two different modes:

Node mode in which each operation is referred to the node or set of nodes currently selected.

Branch mode allows to directly manipulate a selected branch with operations that are meaningful only when considering an entire branch.

There are also a few operations that work on all the skeletal structure. From our experience manually editing a curve-skeleton is usually performed using a trial-and-error approach, hence undo/redo is available for both operations and selection.

4.1. Basic operations

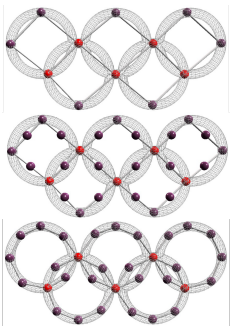
All the tools presented in Section 2 allow the user to perform just basic operations, that are: adding a new node connected to an existing one; translate and rotate in space a selected set of nodes; change the radius of their maximal ball. All these basic functionalities of a skeleton editor are all available in our tool too.

4.2. Node mode

Here we present all the possibilities offered in node mode.

4.2.1. Adding midpoint and constrained movement

Once two existing and directly connected nodes are selected, the user is allowed to create a new node connected to them and placed in the midpoint of the segment connecting them (see inset). Strictly related to this, our tool allows the user to move a selected **Jn** constraining its movement only to points in space that are linear interpolation between the position of its two neighbours, in order to fine tune its position with respect to them. These two operations have shown to be useful when creating a skeleton from scratch in a coarse to fine manner where only **Bn** and **Ln** are created at first and branches are progressively modelled adding **Jn**'s.



4.2.2. Removing nodes

Removing one or more nodes is an operation with a number of different cases that have to be taken in account. In fact, depending on which nodes the user is deleting, the process will be different.

When removing a single node there are three possible cases depending on the type of the node that is going to be removed:

- **Ln** the node is removed without any further operations.
- **Jn** the node is removed and the two nodes linked to it are then connected each other.
- **Bn** the user is allowed to choose between two possibilities. The first one is to delete all the links of the node. Since we do not allow to create separate components, this means that all the **Ln**-ending branches connected to the **Bn** will be removed. Other branches, the ones connected to other **Bn**, will be kept untouched except for the fact that removing one of their endpoint, they will then end with an **Ln**. The second possibility is to transfer all the links of the **Bn** that is going to be deleted, to another node selected by the user. The node on which the links are going to be transferred must be a neighbour of them removed one.

It is important to note that the tool allows to handle isolated nodes or branches but does not allow to explicitly create them. This is a design choice, since a curve-skeleton is always a single connected component we do not allow to create disconnected components, but we allow to load a disconnected skeleton in order to repair it interactively. When a user tries to remove a set of nodes in *Node Mode*, the operation is performed as a sequence of single node removals. When **Bn**'s are involved all the **Ln**-ending branches that are connected to them are traversed and their nodes removed.

4.2.3. Copy and paste

Especially when creating a skeleton from a scratch, it can often happen that a user wants to replicate a part of the skeleton that he is editing. For this reason our tool allows to copy and paste a set of connected nodes. When copying, the user is required to choose a *Source* node, when pasting a *Destination* one.

The *Source* node has to be chosen from the ones that are being copied, the *Destination* node can be any node of the skeleton. When pasting the copied nodes, *Source* will be merged into *Destination* in order to keep a single connected component. *Destination* will be modified only inheriting all *Source*'s copied neighbours. It is important to notice that copy and paste are disjoint operations, and, as such, one can perform them in different moments.

4.2.4. Creating and breaking links

Since we represent the curve-skeleton as a graph, we allow the user to explicitly manipulate the graph's connectivity in two ways:

- Merge two nodes creating a looping-branch.
- Break a link between two nodes. We use the Dijkstra's algorithm to check if breaking the link will create two separate components, if so we prevent the operation.

4.3. Branch mode

The operations we illustrated so far are functions operating on nodes. We here describe the operations work on whole branches of the skeleton.

4.3.1. Pruning

Pruning only works on **Ln**-ending branches. It allows the users to delete one or more nodes of a branch starting from the **Ln** of the selected bone.

4.3.2. Remove

If the branch is a **Ln**-ending one, it will be removed without further operations, if both of its endpoints are **Bn** they will be merged into one of the two. This operation is particularly useful when repairing skeletons since one of the most common defects is the presence of a lot of short spurious branches.

4.3.3. Resampling

Resampling allows the users to change the number of nodes of a branch. It is the most complex operation one can perform on branches. It is done using arc-length parameterisation using the method presented in [Hec94] with slight modifications in order to take into account both position and radii of the involved nodes. In this way, the three-dimensional structure of the branch is maintained and the radii of the new nodes are approximated from the original ones. Both sub-sampling and super-sampling are allowed, but while the second one is a smoothness preserving operation, the first one is not.

A special case for resampling occurs when the branch contains some **An**. Since **An**'s convey semantically meaningful information, those branches are resampled preserving both position and radii of **An**'s. Furthermore each branch can be sub-sampled at least to have $E + \#\mathbf{An}$ nodes, where E is the number of distinct endpoints of the branch.

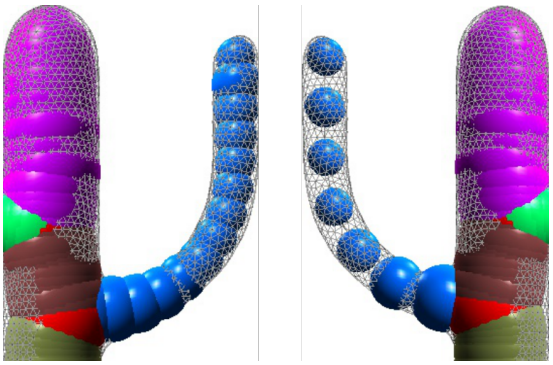


Figure 3: *An example of resampling. The blue branch was subsampled from 54 to 7 nodes. The right image is mirrored to better show the results.*

Resampling is a very useful functionality, due to the fact that automatically extracted skeletons are often composed by a very large amount of nodes and practical applications usually do not need that resolution (e.g., [DS06] produces skeletons with hundreds of nodes even for simple models).

4.4. Skeleton-wide operations

We here present a few operations that allow to manipulate the entire skeleton when a mesh is also loaded. In order to perform all the following operations we keep an AABB tree of the loaded mesh. If the mesh it's not a triangular mesh it is triangulated before building the AABB tree.

4.4.1. Fix the external nodes

When a mesh is loaded along with a skeleton it is possible to check if some of the skeleton's nodes are outside of the mesh and move them inside. This can be useful since most of the skeletonization algorithms cannot guarantee that all the nodes are internal to the mesh; even the more robust ones as [TAOZ12] with particular values of the parameters can lead to branches crossing the mesh boundaries instead of flowing inside them.

For each node N_i we find its closest face of the mesh F with normal N_F . If N_i is outside the mesh we calculate its new position as

$$N_i - [d(N_i, F) + \varepsilon] * N_F$$

where d is the Euclidean distance and ε a small positive number.

4.4.2. Center the skeleton and update the maximal balls

One of the desired properties of a curve-skeleton is centeredness, which is the property of being medial to the shape it describes.

As stated in [CSM07], *perfect centeredness* is achieved when the curve-skeleton lies on the medial surface and it is centered with respect to it. However *perfect centeredness* could be undesirable due to the sensitivity of the medial axis to small perturbations on the object's surface. Moreover achieving such result would require as input both the curve-skeleton and its corresponding medial axis, while state-of-the-art algorithms do not provide such output and some methods do not compute the medial axis at all.

Starting from the quantification of *approximate centeredness* proposed in [CSM07] we have implemented the following algorithm.

For each skeleton's node N_i we denote with $\overrightarrow{N_i}$ the local direction of the skeleton computed as :

$$\overrightarrow{N_i} = \begin{cases} \overrightarrow{N_jN_i} & \text{if } N_i \text{ is } \textbf{Ln} \\ \overrightarrow{N_iN_k} & \text{if } N_i \text{ is } \textbf{Bn} \\ \overrightarrow{N_jN_i} + \overrightarrow{N_iN_k} & \text{if } N_i \text{ is } \textbf{Jn} \end{cases}$$

where N_j and N_k are, respectively, the predecessor and the successor of N_i in the branch they belong to.

We cast n uniformly distributed radial rays with origin in N_i and orthogonal to $\overrightarrow{N_i}$ computing their intersections with the object surface. For each ray R_h we consider only the intersections obtained pinching the object from the inside and we store the intersection I_h that is nearest to N_i. We also discard all the pair of opposite rays for which at least one valid intersection is not found.

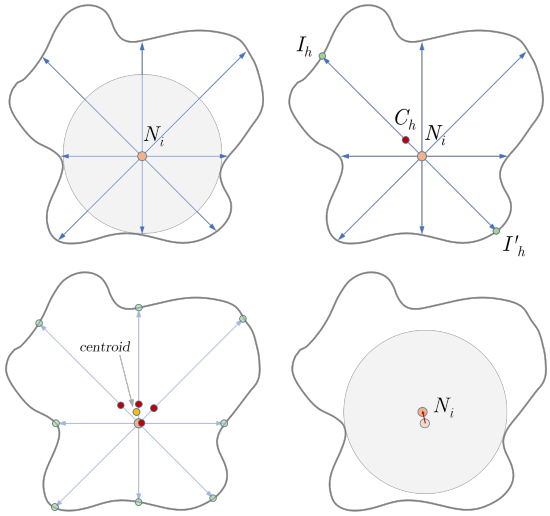


Figure 4: *A section of the tubular branch where the node N_i is centered. On the top left we show the maximal ball and some of the rays casted; on the top right we show how to compute the midpoints; on the bottom left we show how we compute the centroid; on the bottom right the new position of N_i and the new maximal ball are shown.*

For each pair of opposite rays (R_h, R'_h) we calculate C_h as the midpoint of their intersections (I_h, I'_h), then calculate the new position of N_i as the centroid of all the C_h's and the new maximal ball radius as the average semi-distance of all the intersection pairs. If N_i is a **Bn** we calculate its new position and radius, respectively, as the centroid of the positions and mean radius calculated separately for each incident branch. The presented procedure works quite well even if its results are dependent to the original node's position and the skeleton directions. We observed that a single iteration of the procedure is usually not enough to obtain optimal results, however since convergence cannot be guaranteed, it is up to the user to iterate the procedure until the results are satisfactory. Figure 4 shows an usage example of the re-centering procedure.

5. Results and Discussion

In this section we show how some automatically extracted curve-skeletons have been processed with our tool in order to be used as input in the pipeline presented in [ULP*ss].

Figure 6 shows how we edited one of the most challenging skeletons we have faced, in order to compute the quad layout of the model of the *Stanford Dragon*. The skeleton originally had 1050 nodes and, as can be seen in the close-up, some of the branches of the horns where disconnected. The resulting skeleton was composed of 193 nodes, 6 spurious branches were removed during the process. Complete editing, from a) to c) took about 15 minutes.

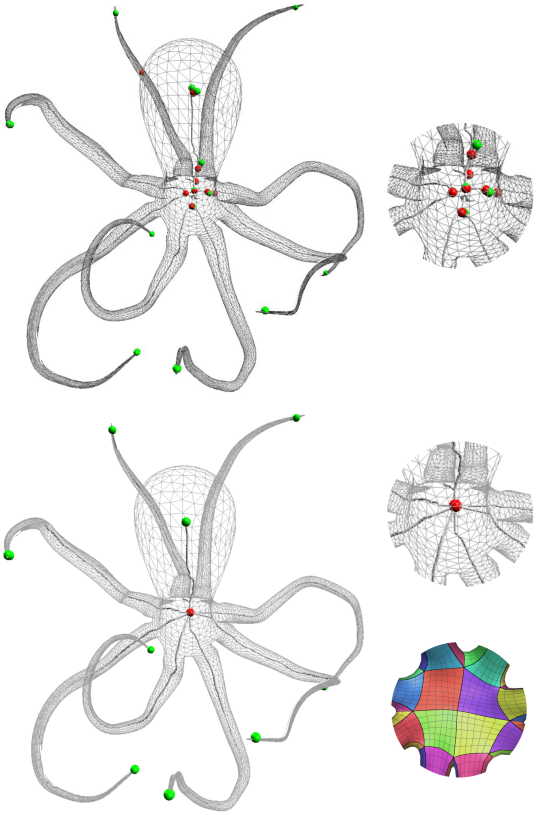


Figure 5: *Skeleton of an octopus model extracted using [TAOZ12] (top) and its edited version (down) with all the spurious branches collapsed. Close-ups on right side of the image show how the unnecessary **Bn** 's were removed in order to obtain the quad layout depicted on the lower right.*

Another example showing how some skeletonization methods, while producing correct results, may fail at conveying the correct semantic information can be seen in Figure 5. Since a human would usually describe an octopus as a living being composed of a big head and 8 tentacles starting from the bottom side of the head, we edited the skeleton removing all the spurious branches inside the head and inside one of the

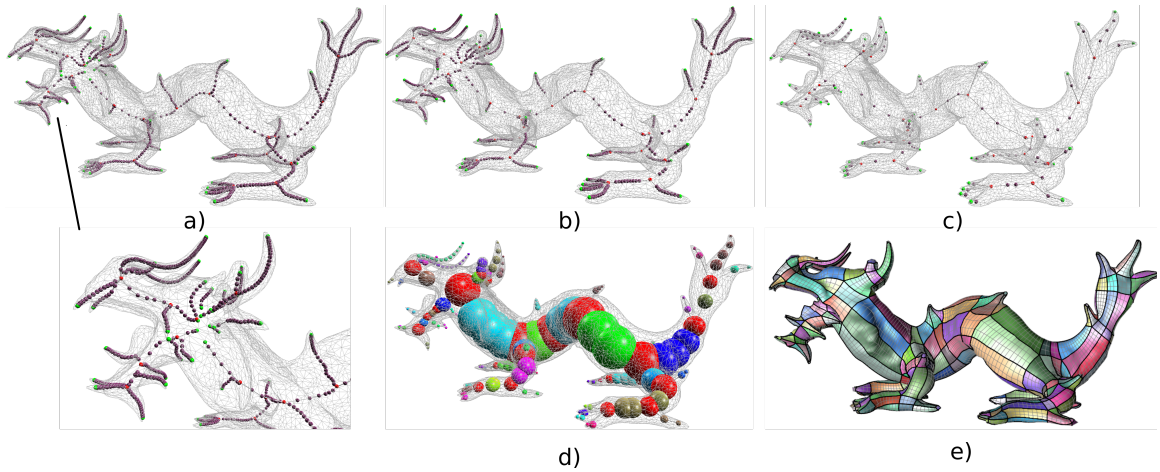


Figure 6: *Editing process of a complex skeleton extracted with [DS06], a) has been computed with a number of disconnected branches (close-up) due to a buggy implementation or erroneous parameter setting. The skeleton was first reconnected b), and simplified c). In the second row we show the skeleton with its associated maximal balls d) and the quad-layout obtained with it e).*

tentacles. The original skeleton contained 23 branches while the result shown on the right of Figure 5, obtained in less than 4 minutes, contained the 9 branches one would expected. It is also important to note that using the original skeleton the resulting quad-layout would have been completely wrong, while the edited version brought to an optimum. In our experiments the tool have shown to be versatile enough to be used for both quick-fixes and complex editing and repairing tasks.

Our tool relies on the Qt Framework, libQGLviewer [lib] and CGAL [CGA] (we use their AABB tree implementation), and it is able to load curve-skeletons produced by the available implementations of [TAOZ12, DS06, LS13].

6. Concluding remarks

In this paper we presented a novel tool for the interactive processing of curve-skeletons in order to make them fit the application in which they will be used. We observed that the evaluation of what is a good curve-skeleton is strictly dependent to the application in which it will be used and none of the existing skeletonization methods can be used for every application always obtaining optimal results. This is not a defect of the proposed approaches since the topological and semantic information that a skeleton encodes are not easy to capture algorithmically, and fine-tuning the parameters required by the skeletonization methods is often necessary but not resolutive. While developing the method presented in [ULP*ss] we have observed that in order to obtain optimal results from the pipeline, an effective and practical approach was to automatically extract the curve-skeleton, choosing the method we experienced to be the best suited for the considered shape using standard parameter values, and process it with the presented tool in order to optimally fit our pipeline requirements.

Future work

Since we consider our tool a useful instrument for researchers and practitioners we plan to make it publicly available and extend its functionalities with both interactive operations and heuristics such as simmetry planes management and automatic spurious branch collapsing.

References

[AJ09] ABEYSINGHE S. S., JU T.: Interactive skeletonization of intensity volumes. *The Visual Computer 25*, 5-7 (2009), 627–635. 1

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton Extraction by Mesh Contraction. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 44:1–44:10. 2

[Aut] AUTODESK 123D:. http://www.123dapp.com/. 2

[BAS14] BÆRENTZEN J. A., ABDRASHITOV R., SINGH K.: Interactive Shape Modeling Using a Skeleton-mesh Co-representation. *ACM Trans. Graph. 33*, 4 (July 2014), 132:1–132:10. 1

[Blu67] BLUM H.: A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form* (1967), 362–380. 1

[BMW12] BÆRENTZEN J., MISZTAL M., WEŁNICKA K.: Converting skeletal structures to quad dominant meshes. *Computers & Graphics 36*, 5 (2012), 555 – 561. Shape Modeling International (SMI) Conference 2012. 2

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 1

[CGA] CGAL, COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY:. http://www.cgal.org. 7

[CSM07] CORNEA N. D., SILVER D., MIN P.: Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (May 2007), 530–548. 2, 3, 5

[DS06] DEY T. K., SUN J.: Defining and Computing Curve-skeletons with Medial Geodesic Function. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 143–152. 2, 5, 7

[HBC*10] HIJAZI Y., BECHMANN D., CAZIER D., KERN C., THERY S.: Fully-automatic Branching Reconstruction Algorithm: Application to Vascular Trees. In *Proceedings of the 2010 Shape Modeling International Conference* (Washington, DC, USA, 2010), SMI '10, IEEE Computer Society, pp. 221–225. 2

[Hec94] HECKBERT P. S.: Graphics Gems IV. Academic Press Professional, Inc., San Diego, CA, USA, 1994, ch. Bilinear Coons Patch Image Warping, pp. 438–446. 5

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 203–212. 1

[JLW10] JI Z., LIU L., WANG Y.: B-Mesh: A Modeling System for Base Meshes of 3D Articulated Shapes. *Computer Graphics Forum (Proceedings of Pacific Graphics) 29*, 7 (2010), 2169–2178. 2

[JST15] JALBA A., SOBIECKI A., TELEA A.: An Unified Multi-scale Framework for Planar, Surface, and Curve Skeletonization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on PP*, 99 (2015), 1–1. 2

[KJT13] KUSTRA J., JALBA A., TELEA A.: Probabilistic View-based 3D Curve Skeleton Computation on the GPU. 237–246. 2

[LGS12] LIVESU M., GUGGERI F., SCATENI R.: Reconstructing the Curve-Skeletons of 3D Shapes Using the Visual Hull. *Visualization and Computer Graphics, IEEE Transactions on 18*, 11 (Nov 2012), 1891–1901. 2

[lib] LIBQGLVIEWER:. http://libqglviewer.com/. 7

[LS13] LIVESU M., SCATENI R.: Extracting curve-skeletons from digital shapes using occluding contours. *The Visual Computer 29*, 9 (2013), 907–916. 2, 7

[LZLW15] LIU L., ZHANG Y., LIU Y., WANG W.: Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design* (2015), 162–172. 1

[SJT14] SOBIECKI A., JALBA A., TELEA A.: Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recognition Letters 47* (2014), 147 – 156. Advances in Mathematical Morphology. 2

[SYJT13] SOBIECKI A., YASAN H., JALBA A., TELEA A.: Qualitative Comparison of Contraction-Based Curve Skeletonization Methods. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, Hendriks C., Borgefors G., Strand R., (Eds.), vol. 7883 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 425–439. 2

[Tag13] TAGLIASACCHI A.: Skeletal Representations and Applications. *CoRR abs/1301.6809* (2013). 1, 2

[TAOZ12] TAGLIASACCHI A., ALHASHIM I., OLSON M., ZHANG H.: Mean Curvature Skeletons. *Comp. Graph. Forum 31*, 5 (Aug. 2012), 1735–1744. 2, 5, 6, 7

[TZCO09] TAGLIASACCHI A., ZHANG H., COHEN-OR D.: Curve Skeleton Extraction from Incomplete Point Cloud. *ACM Trans. Graph. 28*, 3 (July 2009), 71:1–71:9. 2

[ULP*ss] USAI F., LIVESU M., PUPPO E., TARINI M., SCATENI R.: Extraction of the Quad Layout of a Triangle Mesh Guided by its Curve-Skeleton. *ACM Trans. Graph.* (in press). 1, 3, 6, 7

[ZBG*07] ZHANG Y., BAZILEVS Y., GOSWAMI S., BAJAJ C. L., HUGHES T. J.: Patient-specific vascular {NURBS} modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering 196*, 29–30 (2007), 2943 – 2959. 1

[ZBr] ZBRUSH ZSPHERES:. http://pixologic.com/zbrush/features/zspheres/. 2