Int. J. Appl. Math. Comput. Sci., 2020, Vol. 30, No. 1, 123–132
DOI: 10.34768/amcs-2020-0010

CURVE SKELETON EXTRACTION VIA K–NEAREST–NEIGHBORS BASED CONTRACTION

JIANLING ZHOU [a], JI LIU [a,*], MIN ZHANG [a]

[a]College of Computer Science
Chongqing University
No. 174 Shazheng St., Shapingba, Chongqing 400044, China
e-mail: liujiboy@cqu.edu.cn

We propose a skeletonization algorithm that is based on an iterative points contraction. We make an observation that the local center that is obtained via optimizing the sum of the distance to k nearest neighbors possesses good properties of robustness to noise and incomplete data. Based on such an observation, we devise a skeletonization algorithm that mainly consists of two stages: points contraction and skeleton nodes connection. Extensive experiments show that our method can work on raw scans of real-world objects and exhibits better robustness than the previous results in terms of extracting topology-preserving curve skeletons.

Keywords: curve skeleton, points contraction, point cloud, k nearest neighbors.

1. Introduction

The curve skeleton is a 1D representation which captures the essential topological invariant of 3D models. Due to its property of topology preserving (Cornea *et al.*, 2007), the curve skeleton has been widely applied in animated cartoon, shape registration, 3D reconstruction, and many other fields. The unorganized point cloud is the most often used model for capturing the shape of real-world objects. Therefore, extracting the curve skeleton from an unorganized point cloud is an important research topic in computer graphics.

The challenges in skeleton extraction are mostly discussed in three aspects: noise, heavy data occlusions and non-uniform points distribution. Until now a large number of algorithms have been proposed to overcome these challenges (Sharf *et al.*, 2007; Tagliasacchi *et al.*, 2009; Huang *et al.*, 2013). Their effort makes it possible to extract the correct skeleton from a point cloud with large amounts of missing data and noise. However, we notice that there are some problems with the existing work. First, there still remains some space for improvement in the extract topology-preserving skeleton on low-quality point clouds. On the other hand, it takes too much running time for existing algorithms to extract a curve skeleton from the point cloud which has too many points.

In this paper, we introduce our key observation that, once the point cloud has a uniform density, the local center, which is obtained via optimizing the sum of the distances to k nearest neighbors, shows good robustness to noise and missing data. Meanwhile, such a contraction process takes fewer iterations than L_1-median based optimization. Based on such an observation, we propose a fast and novel skeletonization algorithm consisting of two steps. First, in order to obtain a set of uniformly distributed points, we down-sample the point cloud using a 3D grid and re-sample it using the weighted locally optimal projection (WLOP) method (Huang *et al.*, 2009); then we quickly contract the sample points into a skeletal point cloud. Second, we down-sample the skeleton points and connect the skeleton nodes according to the result of confidence computation.

To demonstrate the effectiveness and robustness of the proposed method, we test it and recent techniques on extensive point cloud models. The visual assessment shows that our method extracts a topology-preserving and well-connected curve skeleton from point cloud models that contain noise, incomplete data, and complex structures. Our method outperforms the latest techniques on some challenging point cloud models. Additionally, the actual experiment results show that our method

*Corresponding author

Table 1. Comparison of several skeletonization methods in terms of six aspects. The second column indicates that the input point cloud is required to be pre-processed, including computing normals, re-sampling, etc. The third column indicates that the method aims at processing tree-like point clouds. The fourth column indicates that the input point clouds should satisfy the shape prior knowledge that local regions are generally cylindrical. The three rightmost columns indicate that the robustness of the method to low-quality point clouds. In these three columns the symbol ■ indicates that the method is very robust and □ that it is moderately robust.

Method	Preprocess	Shape		Robustness		
		Tree-like topology	Cylindrical	Non-uniform density	Noise	Missing data
LS (Verroust and Lazarus, 2000)	7	3	7	□	□	□
GR (Bucksch *et al*., 2010)	7	7	3	□	■	□
BSG (Livny *et al*., 2010)	7	3	3	□	□	□
LC (Cao *et al*., 2010)	7	7	7	□	□	□
ROSA (Tagliasacchi *et al*., 2009)	3	7	3	□	□	■
L_1 (Huang *et al*., 2013)	7	7	7	□	■	■
DF (Song *et al*., 2018)	3	7	3	■	□	□

possesses better computing efficiency compared with the recent work.

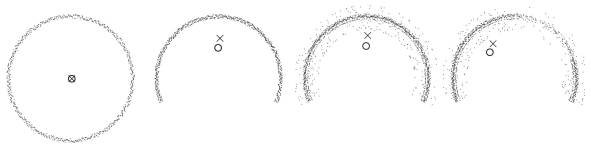
2. Related work

There are many algorithms which aim at extracting a curve skeleton from the 3D model. The latest survey on the skeleton is by Tagliasacchi *et al*. (2016). We will only introduce the related work that extracts the skeleton for mesh models and unorganized point cloud models.

Some researchers focus on extracting skeletons from the mesh or the closed shape. Levet and Granier (2007) use a variational implicit surface to approximate the silhouette curve that is sketched by the user, and find the local minima of the implicit function to generate the skeleton. Au *et al*. (2008) apply the Laplace operator (Desbrun *et al*., 1999) on the mesh models to extract skeletons. Their method moves vertices along their approximate curvature normal direction and then converts the contracted mesh into a 1D curve skeleton through a connectivity surgery process. Inspired by the work of Au *et al.* (2008), Tagliasacchi *et al*. (2012) propose a method that uses a mean curvature flow for curve skeleton extraction. Yan *et al*. (2016) propose a global significance measure which can be used to guide the creation of topology-preserving and clean skeletons of 3D shapes. Li and Wang (2018) compute a medial surface from 3D models of enclosed surfaces using the distance field, and then extract a curve skeleton from the medial surfaces using a locally optimal projection operator (Lipman *et al*., 2007).

Compared with the mesh, extracting a skeleton from unorganized point clouds is more challenging due to unavailable information of vertices connectivity and the noise, as well as incomplete data in the point cloud. Verroust and Lazarus (2000) as well as Lazarus and Verroust (1999) propose a level-set based method (LS) to extract the skeleton of a point cloud model which has a tree-like structure. Sharf *et al*. (2007) devise an algorithm which captures the object's volumetric shape and compute its curve skeleton on-the-fly through a deformable model evolution process. Bucksch *et al*. (2010) build a neighborhood graph from the point cloud and then extract a curve skeleton using a graph-reduction (GR) method. Cao *et al*. (2010) extend the work of Au *et al*. (2008) and use Laplacian based contraction (LC) to extract skeletons from point cloud models. Livny *et al*. (2010) use the branch-structure graph (BSG) for the representation of the tree skeleton. They extract a minimum spanning tree as the initial BSG from the raw scan of the tree and then obtain the skeleton via a biological-priors-driven global optimization.

Currently, the works of Tagliasacchi *et al*. (2009) and Huang *et al*. (2013) represent the state of the art. Tagliasacchi *et al.* (2009) assume that the branches of shapes could be covered by generally cylindrical regions, and thus the curve skeleton can be thought of as a generalized rotational symmetry axis (ROSA) of such shapes. The ROSA method utilizes this shape prior to extracting curve skeletons from point cloud models with large missing data. Huang *et al*. (2013) propose an L_1-median (L_1) method which utilizes the statistical tool L_1-median and extracts a clean and well-connected skeleton from raw point scans. Their method does not place strong requirements on the shape or quality of point cloud models. The latest work is by Song *et al*. (2018). They utilize the distance field (DF) to generate an initial skeleton and use it to guide the L_1-median optimization. Though their method outperforms the L_1 approch on some models, it fails to extract a skeleton from point clouds



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

Fig. 1. We compare the KNN-median (denoted by ∘) and the L_1-median (denoted by ×) of a set of generated points: set of uniformly distributed points (a), remove a portion of the points in (a) (b), add more noise to the set of points in (b) (c), set of points with missing data, noise and varying density (d).

which have flat regions or a large amount of missing data. We compare and contrast these methods in six aspects; see Table 1.

Our method is similar to the L_1 one. Both need to iteratively contract sample points while gradually increasing the neighborhood size. However, our iterative contraction process is totally different from that of the L_1 method and runs faster than it. Additionally, we start to connect skeleton nodes after the end of the contraction process while in the L_1 method these two processes are performed simultaneously.

3. k-Nearest-neighbors based contraction

In this section, we introduce a contraction method that is inspired by the optimization process of the sum of k-nearest-neighbors distance. We will call this method KNN contraction.

Given a set of sample points $X = \{x_i\}_{i\in I} \subset \mathbb{R}^3$, consider the following optimization:

$$\arg\min_X \sum_{i\in X} \sum_{j\in N_{i,k}} ||x_j - x_i||^2, \tag{1}$$

where $N_{i,k}$ indicates the initial k nearest neighbors of x_i in X. A gradient descent based optimization makes all points accumulate into a local center, which we call the KNN-median. To further understand its properties, we generate four sets of points and compare their KNN-median and L_1-median. We select the L_1-median for comparison because it is known to be robust to noise and outliers (Weber and Friedrich, 1929), and widely utilized in the L_1 method. The number of neighbors k is set to one-forth of the size of the points set. From Fig. 1, we can see that the KNN-median is robust to the missing data and noise. Figure 1(d) shows that the KNN-median is more sensitive to the density of points than the L_1-median, but its robustness to noise is sufficient to drive us to investigate the possibility of utilizing k nearest neighbors for skeleton extraction.

3.1. Contraction with fixed k.

Supposing that the k nearest neighbors of a point x_i are natural neighbors (Zhu *et al.*, 2016), and computing the gradient of the energy given in Eqn. (1), we have the following displacement vector $\mathbf{u}_i$ with which the point x_i in X moves in each iteration:

$$\mathbf{u}_i = \frac{1}{k_i} \sum_{j\in N_{i,k}} (x_j - x_i). \tag{2}$$

Each time before we move all points, we perform the principal component analysis (PCA) on the neighborhood of each point. There are two reasons for this. First, the result of the PCA can help determine when to terminate the contraction. Second, the principal direction can help reserve radial contraction and avoid axial contraction. We decompose the displacement vector $\mathbf{u}_i$ into two vectors; one parallel to the local principal direction and the other perpendicular to the local principal direction. The former will lead all points accumulating into a local center, and we choose to scale this vector in each iteration.

When we perform the PCA, the neighborhood G of point x_i is defined as its neighboring points within the distance $3d_{3nn}$, where d_{3nn} is the average length of edges in the three-nearest-neighbors graph built on initial sample points; then we get a symmetric 3×3 positive semi-definite matrix:

$$\mathbf{M} = \sum_{x_j\in G(x_i)} (x_j - x_i) \otimes (x_j - x_i),$$

where $\otimes$ denotes the outer product vector operator. Let $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$ denote the eigenvalues of matrix $\mathbf{M}$ associated with unit eigenvectors $\hat{\mathbf{v}}_i^1, \hat{\mathbf{v}}_i^2, \hat{\mathbf{v}}_i^3$, respectively. We compute $\sigma_i = \lambda_i^1/(\lambda_i^1 + \lambda_i^2 + \lambda_i^3)$ to measure the linearity of the neighborhood G. The principal direction $\mathbf{v}_i^p$ of point x_i is $\mathbf{v}_i^1$ or $-\mathbf{v}_i^1$ if $\cos\langle \mathbf{v}_i^1, \mathbf{u}_i\rangle < 0$. We set the scaling factor to $(1 - \sigma_i)$, and get the following displacement vector:

$$\begin{aligned}\Delta x_i =& |\mathbf{u}_i| \cos\langle \mathbf{v}_i^p, \mathbf{u}_i\rangle (1 - \sigma_i)\mathbf{v}_i^p \\ &+ (\mathbf{u}_i - |\mathbf{u}_i| \cos\langle \mathbf{v}_i^p, \mathbf{u}_i\rangle \mathbf{v}_i^p).\end{aligned} \tag{3}$$

The average of all linearity measurements, denoted as $\overline{\sigma}$, indicates how many sample points have formed a skeletal structure. As this contraction process iterates, $\overline{\sigma}$ will be close to 1. We set a threshold η (η is 0.995 by default) to control when the contraction process stops. The contraction process is elaborated in Algorithm 1.

As shown in Fig. 2, when fixing k_i throughout the contraction, this contraction process can work on some simple objects like cylinders, rings and boxes. However, the objects in a natural world usually have complex structures which make it difficult to set a proper k. We are inspired by the strategy for updating the neighborhood size in the L_1 method (Huang *et al.*, 2013), and therefore devise a algorithm for updating parameter k.

3.2. Updating the number of nearest neighbors. The symbol k_i^t denotes the number of nearest neighbors of sample point x_i at the t-th iteration. When contraction begins, $\forall i \in I, k_i^0 = \lfloor d_{bb}/(\sqrt[3]{|X|}d_{3nn})\rfloor$, where d_{bb} is the diagonal length of the bounding box of initial sample points.

The parameter $\{k_i\}_{i\in I}$ is fixed most of the time and updated every c iterations, where c is empirically set to 12. Note that, when updating the parameter $\{k_i\}_{i\in I}$, we only increase the number of nearest neighbors of points where σ_i is smaller than a threshold τ ($\tau = 0.99$ by default). First, we define a growth rate a ($a = 1.0$ by default) for parameter k_i; then the anticipated increment Δk_i^t for k_i is calculated as follows:

$$\Delta k_i^t = k_i^0(a^2(2(t/c)-1)+2a) \times \exp\left(-\frac{(\sigma_i-\sigma_{\min})^2}{(0.95-\sigma_{\min})^2}\right).$$

We do not expect a sudden change in the value of k^{t+1} around the points where σ_i is close to the threshold τ, hence a Gaussian weighting and PCA weighting are incorporated to determine k_i^{t+1}:

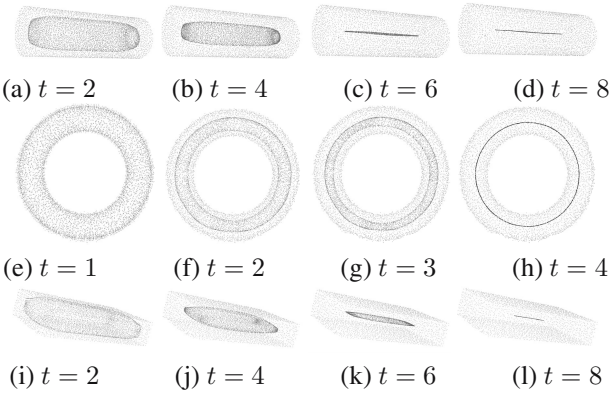
$$k_i^{t+1} = \frac{\sum_{j\in N_{j,k}}(k_j^t+\Delta k_j^t)\theta(||x_i-x_j||)}{\sum_{j\in N_{j,k}}\theta(||x_i-x_j||)}$$

where $\theta(t)$ is a Gaussian weighting function $\theta(t) = e^{-t^2/(r_0/2)^2}$. The support radius $r_0 = d_{bb}/\sqrt[3]{|X|}$.

The parameter k_i is restricted to the range $[20, 0.25|X|]$. Armed with such a parameter update policy, the iterative point contraction process extracts a topology-preserving skeletal structure from more complex point cloud models, as shown in Fig. 3.

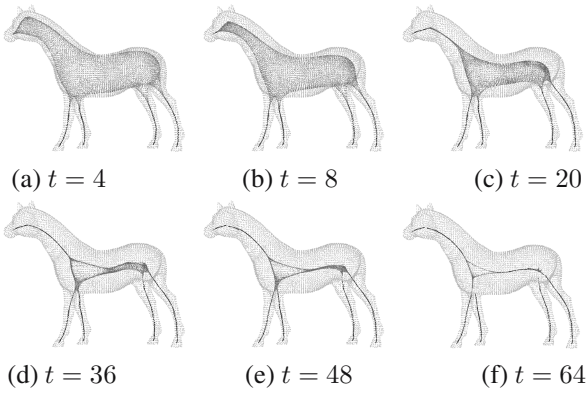
Algorithm 1. KNN contraction.

Require: Sample points $X = \{x_i\}_{i\in I}$
iteration number $t = 0$
compute $\{\sigma_i\}_{i\in I}, \{\mathbf{v}_i^1\}_{i\in I}, \{k_i^0\}_{i\in I}, \{N_{i,k}\}_{i\in I}$,
repeat
  $\forall i \in I$, compute the displacement vector Δx_i
  $\forall i \in I, x_i = x_i + \Delta x_i$
  $\forall i \in I$, compute σ_i and $\mathbf{v}_i^1$
  if $t\%c == 0$ and $t > 0$ **then**
    $\forall i \in I$, update k^{t+1}
    $\forall i \in I$, search k_i^{t+1} nearest neighbors for x_i
  else
    $\forall i \in I, k_i^{t+1} = k_i^t$
  end if
  $t = t + 1$
until $\overline{\sigma} < \eta$
return X



(a) $t = 2$ (b) $t = 4$ (c) $t = 6$ (d) $t = 8$
(e) $t = 1$ (f) $t = 2$ (g) $t = 3$ (h) $t = 4$
(i) $t = 2$ (j) $t = 4$ (k) $t = 6$ (l) $t = 8$

Fig. 2. KNN contraction on simple objects: cylinder, 5821 points, $k = 300$ (a)–(d), ring, 2485 points, $k = 130$, (i)–(l), box, 4064 points, $k = 500$ (e)–(h). The light gray points are a point cloud, the gray points are moving points, the black points are moving points whose σ is larger than η.



(a) $t = 4$ (b) $t = 8$ (c) $t = 20$
(d) $t = 36$ (e) $t = 48$ (f) $t = 64$

Fig. 3. KNN contraction with gradually increasing k on the horse model: points belonging to the horse's limbs firstly complete the contraction(a)–(b), as the number t of iterations increases, points belonging to the larger branches of horse models are contracted to a skeletal structure (c)–(f).

4. Skeletonization algorithm

In this section, we present a skeletonization algorithm consisting of two stages. First, we re-sample the raw point cloud and perform the KNN contraction operation. Second, we extract a graph based representation of the curve skeleton from the skeletal point cloud. Figure 4 shows an overview of our algorithm.

4.1. Constructing a skeletal point cloud. The aforementioned KNN contraction process can extract the skeletal structure from uniformly distributed samples. However, the sample points of raw scans are usually non-uniformly distributed, and a non-uniform distribution may make the contraction process fail to work on the raw scans. Therefore, to make use of the KNN contraction

to extract a curve skeleton from raw scans, we acquire a set of uniformly distributed samples via a re-sampling process at the very start.

First, we make use of a 3D grid to uniformly down-sample the raw point cloud. The size of the voxel in the grid is d_{3nn} and its barycenter is taken as the new sample point. Only a 3D grid based re-sample does not satisfy the demands for the uniformity of the points distribution. Next, we apply the WLOP (Huang *et al.*, 2009) algorithm to the previous output, which can denoise and uniformly re-sample the point cloud. We randomly pick 7000 sample points by default, and set the default number of iterations to 10. Other parameters for the WLOP are set as default in the original paper.

After doing these, we apply the nearest neighbor contraction to the samples and get a skeletal point cloud from them.

4.2. Extract curve skeleton. The curve skeleton is represented as an undirected graph $G(V, E)$, while the skeletal point cloud remains unorganized. We take two steps to extract a curve skeleton from the skeletal point cloud. First, we down-sample the skeletal points, obtain the skeleton nodes, and connect them. Next, we connect the disconnected branches in the skeleton and clean the outliers in the skeleton. To obtain the skeleton nodes, we first filter the points whose corresponding σ is smaller than 0.99. From the points that have not been visited, every time we pick out the point x_i which has the largest σ, and search the farthest five points along two directions $\mathbf{v}_i^1$ and $-\mathbf{v_i^1}$ within a ball whose radius is $d_{bb}/20$. The median point of these five points is the next skeleton node, and this node is the next ball's center. We repeat this process until all skeletal points have been visited. More details about this down-sample process are described in the moving least-squares (MLS) method (Lee, 2000).

After the down-sample step, influenced by the incomplete data or improper parameter setting, the curve skeleton may consist of several disconnected branches; see Fig. 4(e). For the endpoints v_e of all branches, we search all skeleton nodes and compute the corresponding confidence estimate γ. If a skeleton node v_s has the largest confidence γ and the confidence is larger than a preset threshold λ ($\lambda = 0.6$ by default), we connect these two nodes. The computation of the confidence is summarized in Algorithm.2.

Algorithm 2. Confidence computation for branch connection.

Require: Skeleton node v_s, endpoint v_e

1: $\gamma = 1.0$
2: find the previous node v_{e-1}
3: compute angle θ between vectors $\overrightarrow{v_{e-1}v_e}$ and $\overrightarrow{v_ev_s}$
4: $\gamma = \cos(\theta)\gamma$
5: compute the length s of the branch where v_e is located
6: $\gamma = \gamma - (|\overrightarrow{v_ev_s}|/s)$
7: **return** confidence γ

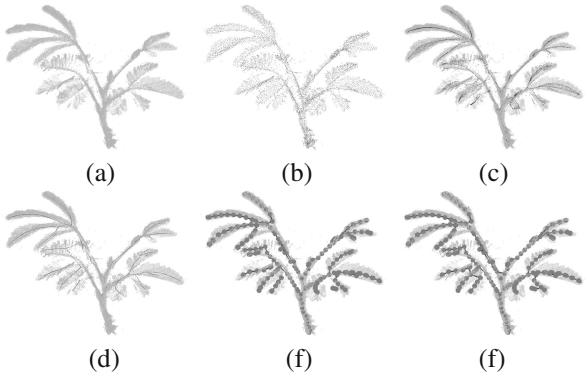


Fig. 4. Overview of our proposed skeletonization algorithm: raw point cloud (a), sample points (b), process of KNN contraction (c), raw point cloud and a skeletal point cloud (d), downsample the skeletal point cloud and connect the nodes (e), connect the disconnected branches and remove outliers (f).

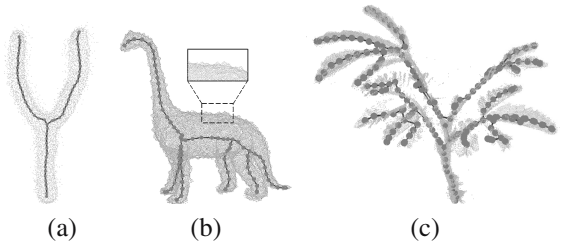
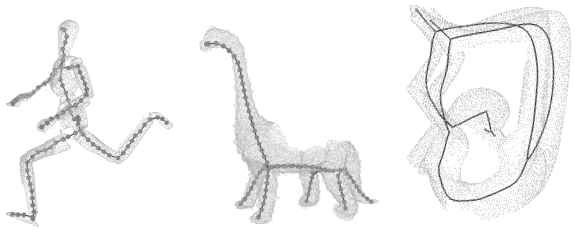


Fig. 5. Extracted curve skeletons for point clouds with noise; default parameters are applied to them: noisy-Y model (a), noisy dinosaur model (b), raw scan of a mimosa (c).

5. Results and discussions

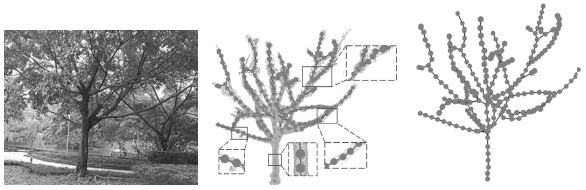
To validate the effectiveness of our skeletonization algorithm, we pick out several representative point cloud models and run the algorithm on them. The results show that our skeletonization can extract a correct skeleton from various point clouds and is robust to noise, incomplete data and non-uniform point distributions. We also discuss the parameter setting strategy and the main limitation of our skeletonization algorithm in this section.

5.1. Results. We implement the skeletonization algorithm under the environment of Java 8 and run it on a laptop which has 8 GB memory and an Intel Core i7-8550U processor running at 1.80 GHz. The operating system for the laptop is Windows 10. We validate the performance of our algorithm on point clouds which



(a) (b) (c)

Fig. 6. Extracted curve skeletons for point clouds with missing data: running man model (a), noisy dinosaur with missing data (b), lady model (c).



(a) (b) (c)

Fig. 7. Point cloud and an extracted curve skeleton of complex objects: ficus virens (a), point cloud of the tree (b), curve skeleton of the tree (c).

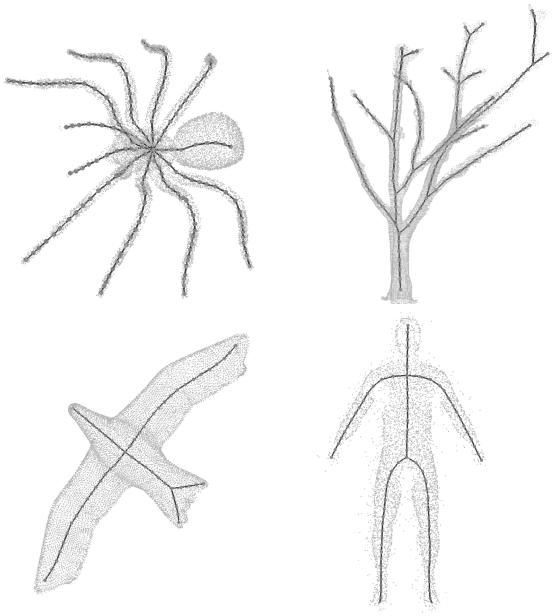


Fig. 8. Ground truth skeletons that we manually create.

contain noise, incomplete data and complex structures; then we download the software provided by Huang *et al*. (2013) and the code of Tagliasacchi *et al*. (2009) from the authors' repository on Github and compare their curve skeletons with ours both from the visual and quantitative aspects. We normalize the point cloud and follow the authors' instructions before running their algorithms. For every presented result of the L_1 and ROSA methods, we have tried at least five groups of parameters to pick out the best skeleton for comparisons.

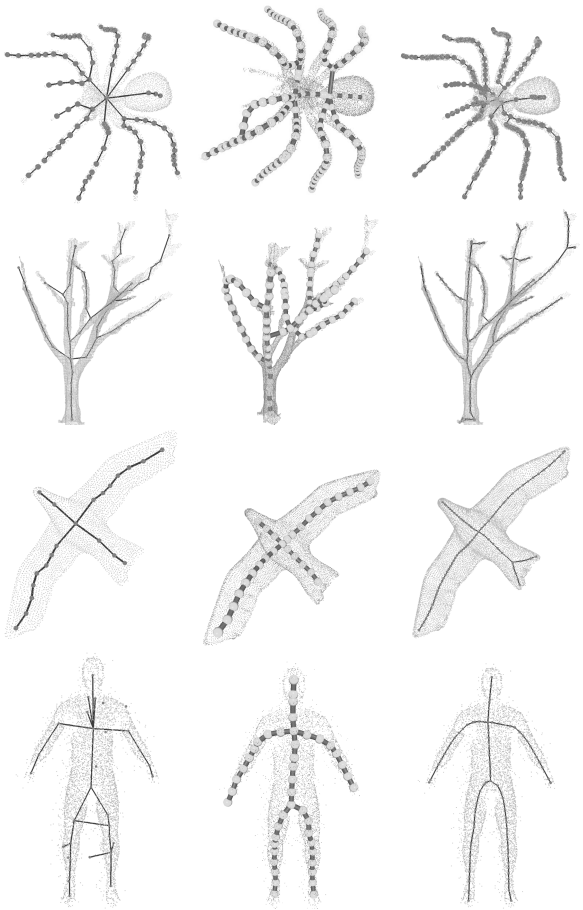
We test the robustness of our algorithm to the noise of the point cloud first. We run the algorithm on three point clouds that have obvious noise around the boundary; see Fig. 5. Among these three point clouds, the noise on the first one and the second one is generated manually, and the third one is from a raw scan, which has obvious noise and a non-uniform points distribution. The results show that the noise and outliers do not affect the correctness of our skeleton. The quality of the extracted skeleton of the third point cloud is not as satisfactory as that of the previous ones, but the topology is correct and no skeleton nodes are misconnected. We also test the performance of our algorithm on the point clouds with large amounts of missing data; see Fig. 6. We use two raw scans and a noisy dinosaur model as test cases. We manually remove some points from the dinosaur model to mock the incomplete data. The results show that our algorithm can handle the point clouds with incomplete data.

We also test the algorithm on the point cloud yielded by the multi-view stereo technique (Seitz *et al*., 2006; Belter *et al*., 2016). We take 30 photographs of a ficus virens and synthesize a 3D point cloud using the open source software MVE (Fuhrmann *et al*., 2014). The point cloud contains much noise, non-uniformly distributed points and complex structures, which are marked in the box of Fig. 7. Despite these challenges, our algorithm extracts a well-centered and topology-preserving tree skeleton.

We compare our resulting skeletons with the ROSA and L_1 methods; see Fig. 9. The results show that our approach achieves better performance than current state-of-the-art works on some challenging models. From the visual comparison, our method outperforms the L_1 one on the spider model and the tree model, and it outperforms the ROSA method on the human model. For quantitative comparison, we manually create ground truth skeletons of these four point clouds using modeling software, as shown in Fig. 8. We count the number of incorrect or missing branches in the generated skeleton and denote it as N_b. We call the skeleton node that connects more than three skeleton branches as the

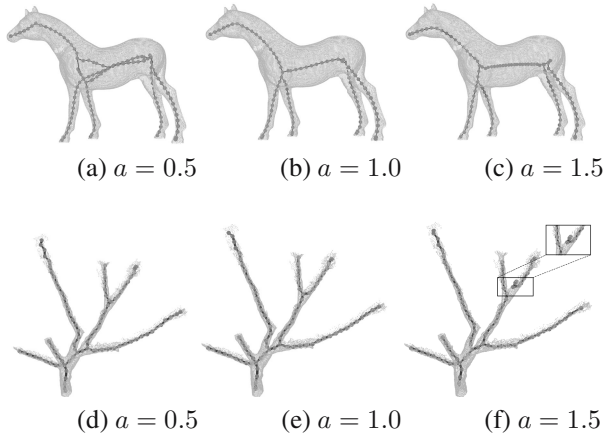
Table 2. Number of incorrect skeleton branches and connection nodes in the skeleton generated by different methods.

Model	ROSA		L_1		Ours	
	N_b	N_c	N_b	N_c	N_b	N_c
Spider	3	4	1	6	0	4
Tree	5	3	8	3	1	2
Bird	2	0	2	0	0	0
Human	7	1	0	0	0	0



(a) ROSA (b) L_1 (c) ours

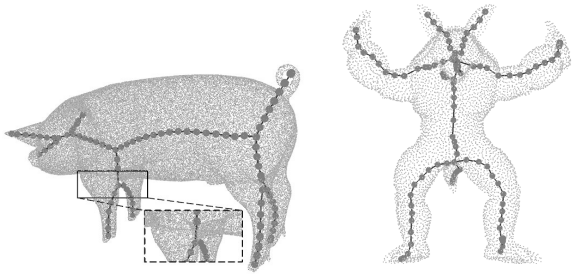
Fig. 9. Comparisons with the previous skeletonization algorithms: results of the ROSA method (a), results of the L_1 method (b), results of our method (c).



(a) $a = 0.5$ (b) $a = 1.0$ (c) $a = 1.5$

(d) $a = 0.5$ (e) $a = 1.0$ (f) $a = 1.5$

Fig. 10. Extracted curve skeleton under different growing speed a; other parameters are kept fixed for both models: horse model (a)–(c), part of the tree model (d)–(f).



(a) (b)

Fig. 11. Centeredness of our extracted curve skeletons is not good on some models; we render down-sampled points of the armadillo model instead of raw points for better presentation of the skeleton.

connection node. Then we count the number of the incorrect connection nodes and denote it with N_c. The quantitative comparison of these two indicators is shown in Table 2. It can be seen that the skeleton generated by our method has fewer problem skeleton branches and incorrect connection nodes than is the case with the other two methods.

We also test the running time of our skeletonization algorithm and compare it with the previous work. Because the ROSA method (Tagliasacchi *et al*., 2009) is implemented in MATLAB, we compare the running time with the L_1 method (Huang *et al*., 2013), which is implemented in C++, whereas we write the program in Java. We take 7000 sample points from the point cloud when running our method. When we run the L_1 approach, we take 1000 sample points and use the default parameters. From Table 3, we can see that, though Java is slower than C++ because of the mechanism differences, our skeletonization method takes fewer iterations for extracting a curve skeleton and runs faster on most point cloud models. On the Armadillo model with over 172 K points, the running time of our skeletonization algorithm is about 94 seconds, nearly one-fourth of that of the L_1 method. Additionally, most of the operations in the contraction process can be parallelized, and thus the running time of the algorithm can be further accelerated.

5.2. Discussions. There are two important parameters in our skeletonization algorithm; the initial number of neighbors k^0 and the neighborhood growth rate a. The former one controls the minimal size of the neighborhood where we need to extract a skeleton, for example, the radius of legs in the horse model. In our experiments, the default value of k^0 works very well for most point clouds. The latter one controls the growth rate of the number of neighbors. We notice that our algorithm is not so robust to the change of this parameter. We change the growth rate a and test the algorithm on two representative models, which is the horse model and the tree branch one, respectively; see Fig. 10. The results show that, once

Table 3. Running time.

Model	Points	L_1 method		Our method	
		Iterations	Running time	Iterations	Running time
Noisy-Y	7029	55	7.10 s	16	6.08 s
Horse	48485	264	65.88 s	51	41.87 s
Mimosa	40551	161	44.36 s	26	16.58 s
Armadillo	172974	169	362.33 s	89	93.78 s
Lady	11125	142	21.51 s	46	17.31 s
Noisy-Dinosaur	23255	192	34.19 s	62	50.3 s
Spider	18671	214	55.68 s	42	22.64 s
Running Man	11191	147	21.20 s	38	13.45 s
Ficus virens	68103	117	37.31 s	27	17.56 s
Tree	32400	74	20.46 s	31	18.55 s
Bird	11537	106	23.45 s	51	17.81 s
Human	6045	135	27.02 s	33	8.27 s

the rate is too small, the resulting skeleton will contain unexpected branches; see Fig. 10(a). Conversely, some branches may lose connectivity to other branches; see Fig. 10(f). Therefore, the growth rate a should be set more carefully according to the shape of the object. For objects consisting of branches of similar size, like trees, the growth rate a should be set at a smaller value. For objects consisting of branches with large differences in size, like horses, it should be set larger.

The main limitation of our skeletonization algorithm is that the extracted skeleton is not very well-centered in some point clouds, as shown in Fig. 11. Many previous works (Huang *et al*., 2013; Wang *et al*., 2012) use an ellipse-fitting step to improve the centeredness of the skeleton. However, we think such a re-centering step is not a good solution. The reasons are described below. First, the ellipse fitting based re-centering step requests the local cylindrical shape, which cannot be always satisfied in real-world objects, for example, the leaves of plants. Second, for one skeleton node, it is hard to give an automated algorithm to find the correct subset of point clouds for performing ellipse fitting. Once the outliers and near-by branches are included, the centeredness of a skeleton may even deteriorate. In our view, the centeredness should be best guaranteed during the points contraction. A better contraction algorithm is to be studied in the future.

6. Conclusion

In this paper, we proposed a novel skeletonization method for point clouds. The approach features (i) utilizing the nearest neighbors to contract sample points into a skeletal point cloud, (ii) connecting the skeleton branches via a confidence computation. Extensive case studies suggest that our method clearly solves the existing challenges in extracting curve skeleton from point clouds. It is robust against a noise, outliers, incomplete data and complex structures in the point cloud. It can handle arbitrary shapes of real-world objects, including flat regions, tree-like structures, loops, and so on. Employing this method, we can extract a clean and topology-preserving skeleton from raw scans. Also, this skeletonization algorithm can be further improved in two aspects. First, we need to improve the points contraction so that the centeredness of a curve skeleton can be guaranteed. Second, it can be improved by investigating automated strategies for determining the neighborhood growth rate a from the point cloud.

Acknowledgment

This work is supported by the Chongqing Research Program of Basic Research and Frontier Technology (grant no. cstc2019jcyj-msxmX0033), the National Natural Science Foundation of China (grant no. 61701051), and the Fundamental Research Funds for the Central Universities (grant no. 2019CDYGYB012).

References

Au, O.K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D. and Lee, T.-Y. (2008). Skeleton extraction by mesh contraction, *ACM Transactions on Graphics* **27**(3): 44.

Belter, D., Łabecki, P., Fankhauser, P. and Siegwart, R. (2016). RGB-D terrain perception and dense mapping for legged robots, *International Journal of Applied Mathematics and Computer Science* **26**(1): 81–97, DOI: 10.1515/amcs-2016-0006.

Bucksch, A., Lindenbergh, R. and Menenti, M. (2010). Skeltre: Robust skeleton extraction from imperfect point clouds, *The Visual Computer* **26**(10): 1283–1300.

Cao, J., Tagliasacchi, A., Olson, M. and Zhang, H. (2010). Point cloud skeletons via Laplacian based contraction, *Shape*

Modeling International (SMI 2010), Aix-en-Provence, France, pp. 187–197.

Cornea, N.D., Silver, D. and Min, P. (2007). Curve-skeleton properties, applications, and algorithms, *IEEE Transactions on Visualization and Computer Graphics* **13**(3): 530–548.

Desbrun, M., Meyer, M., Schroder, P. and Barr, A.H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow, *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA*, pp. 317–324.

Fuhrmann, S., Langguth, F. and Goesele, M. (2014). MVE—a multi-view reconstruction environment, *in* R. Klein and P. Santos (Eds), *Eurographics Workshop on Graphics and Cultural Heritage*, Eurographics Association, Strasbourg, pp. 11–18.

Huang, H., Li, D., Zhang, H., Ascher, U. and Cohen-Or, D. (2009). Consolidation of unorganized point clouds for surface reconstruction, *ACM Transactions on Graphics (TOG)* **28**(5): 176.

Huang, H., Wu, S., Cohenor, D., Gong, M., Zhang, H., Li, G. and Chen, B. (2013). L1-medial skeleton of point cloud, *ACM Transactions on Graphics* **32**(4): 65.

Lazarus, F. and Verroust, A. (1999). Level set diagrams of polyhedral objects, *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications, Ann Arbor, MI, USA*, pp. 130–140.

Lee, I.-K. (2000). Curve reconstruction from unorganized points, *Computer Aided Geometric Design* **17**(2): 161–177.

Levet, F. and Granier, X. (2007). Improved skeleton extraction and surface generation for sketch-based modeling, *Graphics Interface 2007, New York, NY, USA*, pp. 27–33.

Li, L. and Wang, W. (2018). Improved use of LOP for curve skeleton extraction, *Computer Graphics Forum* **37**(7): 313–323, DOI: 10.1111/cgf.13570.

Lipman, Y., Cohen-Or, D., Levin, D. and Tal-Ezer, H. (2007). Parameterization-free projection for geometry reconstruction, *ACM Transactions on Graphics* **26**(3): 22.

Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H. and Elsana, J. (2010). Automatic reconstruction of tree skeletal structures from point clouds, *International Conference on Computer Graphics and Interactive Techniques, Seoul, South Korea*, p. 151.

Seitz, S.M., Curless, B., Diebel, J., Scharstein, D. and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms, *CVPR 2006, New York, NY, USA,* pp. 519–528.

Sharf, A., Lewiner, T., Shamir, A. and Kobbelt, L. (2007). On-the-fly curve-skeleton computation for 3D shapes, *Computer Graphics Forum* **26**(3): 323–328, DOI: 10.1111/j.1467-8659.2007.01054.x.

Song, C., Pang, Z., Jing, X. and Xiao, C. (2018). Distance field guided l_1-median skeleton extraction, *The Visual Computer* **34**(2): 243–255.

Tagliasacchi, A., Alhashim, I., Olson, M. and Zhang, H. (2012). Mean curvature skeletons, *Computer Graphics Forum* **31**(5): 1735–1744, DOI: 10.1111/j.1467-8659.2012.03178.x.

Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N. and Telea, A. (2016). 3D skeletons: A state-of-the-art report, *Computer Graphics Forum* **35**(2): 573–597, DOI: 10.1111/cgf.12865.

Tagliasacchi, A., Zhang, H. and Cohenor, D. (2009). Curve skeleton extraction from incomplete point cloud, *International Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA*, p. 71.

Verroust, A. and Lazarus, F. (2000). Extracting skeletal curves from 3D scattered data, *The Visual Computer* **16**(1): 15–25.

Wang, Y., Chang, X., Ning, X., Zhang, J., Shi, Z., Zhao, M. and Wang, Q. (2012). Tree branching reconstruction from unilateral point clouds, *in* Z. Pan *et al.* (Eds.), *Transactions on Edutainment VIII*, Springer, Berlin/Heidelberg, pp. 250–263.

Weber, A. and Friedrich, C.J. (1929). *Alfred Weber's Theory of the Location of Industries*, University of Chicago Press, Chicago, IL.

Yan, Y., Sykes, K., Chambers, E., Letscher, D. and Ju, T. (2016). Erosion thickness on medial axes of 3D shapes, *ACM Transactions on Graphics* **35**(4): 38.

Zhu, Q., Feng, J. and Huang, J. (2016). Natural neighbor: A self-adaptive neighborhood method without parameter k, *Pattern Recognition Letters* **80**: 30–36.

Jianling Zhou received his BS degree in computer science from Chongqing University, China, in 2017. He is currently pursuing his MS degree in computer science at Chongqing University. His research interests include point cloud processing, computer graphics and computer vision.

Ji Liu is on the faculty of the College of Computer Science at Chongqing University, China. He received his MS and PhD degrees in computer software and theory from Chongqing University in 2005 and 2009, respectively. He started working on tree modeling already during his PhD thesis. His research interests are in computer graphics, 3D reconstruction and tree modeling.

Min Zhang received her BS and PhD degrees in computer science from Chongqing University, China. Since 2003, she has been a lecturer in the School of Computer Science at Chongqing University. She is the author or a co-author of more than 20 research publications. Her research interests are in pattern recognition, machine learning algorithms, and applied intelligence.

Received: 13 May 2019
Revised: 8 September 2019
Accepted: 18 October 2019