Computer Aided Geometric Design 82 (2020) 101929

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Visibility-driven skeleton extraction from unstructured points ☆

Lifeng Zhu [a,*], Wen Xing [a], Aiguo Song [a], Yongjie Jessica Zhang [b]

[a] The State Key Laboratory of Bioelectronics, Jiangsu Key Lab of Remote Measurement and Control, School of Instrument Science and Engineering, Southeast University, China
[b] Department of Mechanical Engineering, Carnegie Mellon University, USA

ARTICLE INFO

Article history:
Received 30 April 2020
Received in revised form 20 August 2020
Accepted 27 August 2020
Available online 12 September 2020

Keywords:
Curve skeleton
Point cloud
Visibility

ABSTRACT

We present an algorithm for curve skeleton extraction from unstructured points using visibility as a guide. We introduce visible cells inside the point samples. By locating a viewpoint in the space, visible points from the viewpoint are connected to indicate the visible region, which locally captures the interior structure of the points and is called *a visible cell*. We then analyze and clean the visible cells for noisy and incomplete points. By connecting the cleaned visible cells with a region growing method, we obtain the skeleton along which the visibility of the unstructured points is balanced. No normal computation or ellipse fitting is required by using the proposed method. We show that the proposed skeleton well represents the shape. We experiment with our algorithm on imperfect points and show that comparative curved skeletons are extracted with relative lower computational cost.

1. Introduction

A curved skeleton exhibits the lower-dimensional structure of a shape. The simplicity of the structure allows efficient control of the shape, facilitating automatic and interactive shape manipulation. In geometric processing, applications such as shape editing Jacobson et al. (2014), compression Thiery et al. (2013), matching Hilaga et al. (2001), reconstruction Yin et al. (2014) and meshing Zhang et al. (2007), Liu et al. (2015), Usai et al. (2015) show the significance of a good skeleton abstraction of a shape and require an efficient skeleton extraction technique.

Skeleton extraction has been studied for years. Many algorithms were designed for extracting skeletons from structured meshes Tagliasacchi et al. (2016). As a raw data representation of shapes, unstructured points are commonly available for geometric applications with the development of sampling and data acquisition techniques. However, in comparison to skeletons from structured meshes, skeleton extraction from unorganized points is a challenging problem due to the missing connectivity. The orientation of the points and the topology of the shape are ambiguous, making the underlying skeletons hard to discover. The problem gets even more challenging when raw points are corrupted with noises and missing parts.

Recent efforts on skeleton extraction from raw points focus on locally finding cutting planes Tagliasacchi et al. (2009) or the mass centers Cao et al. (2010), Huang et al. (2013), Qin et al. (2019). Fitting, clustering, relaxation or contraction operators are iteratively applied to find the geometry of the skeleton. With these efforts, by carefully selecting parameters and

☆ Editor: Carlotta Giannelli is the Managing Guest Editor.

* Corresponding author.
E-mail addresses: lfzhulf@gmail.com (L. Zhu), jessicaz@andrew.cmu.edu (Y.J. Zhang).

https://doi.org/10.1016/j.cagd.2020.101929
0167-8396/

tuning the topological connection, curved skeletons with fairly good quality emerge after iterations. As the computational cost is heavy, the tuning process is relatively tedious in finding a good skeleton.

Instead of computing the local mass centers or planes, in this paper we use the visibility as a hint and propose a new method to find the skeletons from unstructured points. We compute the visible points and form a *visible cell* for each viewpoint inside the points. We show that the visibility can be used to locally connect the unorganized points and locate candidate skeleton points on the skeleton. By analyzing the visible cell under noisy and incomplete points, we define a feature to characterize the quality of the candidate points from visibility. A region growing method is then proposed to connect the visible cells, producing a visibility-driven skeleton from the points. We show that the visibility-driven skeleton well captures the structure of points with an efficient algorithm.

In the following, we first briefly summarize related work in Sec. 2. In Sec. 3, we introduce the concept of visible cells and analyze its behavior under noisy and incomplete points. Then we propose a method to extract skeletons from unstructured points based on the visibility information in Sec. 4. After the experiments and discussion in Sec. 5, we conclude the paper in Sec. 6.

2. Related work

Medial axis transform Blum (1967) has been recognized as an important lower-dimensional structure of shape for years and many skeletonization methods were developed to find the medial axis or its variants Kaleem and Stephen (2009). By its strict definition, the medial axis transform may produce two-dimensional (2D) structures from three-dimensional (3D) shapes. As an alternative choice, curved skeletons with only one-dimensional (1D) structures are introduced to geometric processing Cornea et al. (2007). In this work, we work on 1D skeletons from points. As there are many thorough surveys on the curved skeleton and medial axis Saha et al. (2016) Tagliasacchi et al. (2016), we will not cover all the previous methods and only briefly review closely related literatures.

Taking unstructured points as input, there are many ways to construct their curved skeleton. Because skeleton extraction techniques are well developed for structured data, such as curves or images in 2D and meshes or distance fields in 3D, a common solution is to first convert the point cloud to implicit representations using voxelization or explicit representations by applying shape reconstruction. We regard these methods as indirect methods. We do not take steps on indirect methods because the reconstruction problem itself is an unsolved problem.

Methods of directly working on points have been developed in the past years. The Voronoi diagram from boundary points was introduced to produce curved skeletons Ogniewicz and Ilg (1992) in early work. Medial axis transform has also been proposed to obtain the skeleton points Ma et al. (2012). However, these methods are sensitive to noises on the points by definition. In Sharf et al. (2007), a deformable blob grows inside the points and a curved skeleton is defined by tracking the front of the growing region. The growing operation requires the Euclidean distance field of the points as guidance. However, the missing data may lead to a corrupted distance field, making it hard to set the parameters to stop the growing operation around the missing parts. In Tagliasacchi et al. (2009), the authors assume the underlying shapes are locally generalized cylinders and fit cutting planes to the points. The fitting operation relies on a well-oriented point cloud, which requires to filter the points with low quality. In Cao et al. (2010), the Laplacian contraction operations Au et al. (2008) on closed polygonal meshes are extended to raw points. Iteratively applying the Laplacian operator essentially moves each point to the center of its neighbors, finally converging to a low-dimensional structure. However, it is still challenging to obtain a good Laplacian from raw points. In Huang et al. (2013), the authors design l_1-medial filters on the raw points, moving each sample point to the center of its neighboring region robustly. Curved skeletons can be obtained from raw points by gradually adjusting the neighborhood size. An online-published work by Qin et al. (2019) further improves the way to locate the mass center locally by using optimal mass transport, showing its advantages in restoring the topology when the input points are highly corrupted. The previous study shows that these methods based on computing the local mass center are effective to produce curve skeletons, however, at a high computation cost. In this work, other than finding the mass center, we propose a lightweight method for skeleton extraction using visibility as a guide. Recently, deep learning for point primitives has developed. Learning-based methods also show their effects in skeleton extraction and medial axis transform from points Yin et al. (2018) Yang et al. (2020). As the quality of the results depends on well-prepared training data, a lightweight tool for extracting skeletons will be a good choice to prepare the data.

The visibility of a point set with respect to a viewpoint is related to the spatial distance between the points and the viewpoint. As shown in Katz et al. (2007), it is possible to efficiently determine a subset of visible points without reconstructing the surface from points. Taking this elegant solution as an insight, in this paper, we aim to use the visibility to directly extract skeletons from unstructured points without surface reconstruction. The hidden point removal (HPR) operator in the direct visibility of points has been further generalized in Katz and Tal (2015). In Mehra et al. (2010), the authors improve the HPR operator in the presence of noises and show its application in surface reconstruction. In parallel to those applications, the HPR operator has also been applied to orient the point normals Cao et al. (2011), render the point set surfaces Machado e Silva et al. (2014) or even enhance the visibility of documents Kligler et al. (2018). The visibility conditions are related to the skeleton in the work from Peters et al. (2015). In this study, the authors use the medial axis transform of the points to assist the computation for visibility. Differently, we use visibility as a guide to generate a good skeleton from unstructured points.

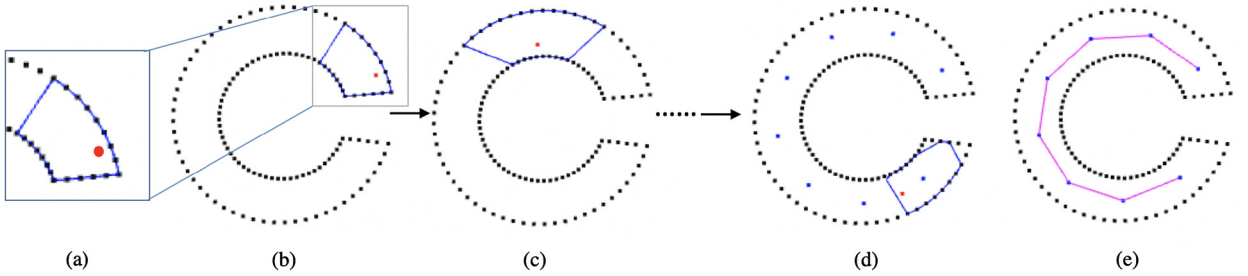

Fig. 1. The pipeline of our algorithm. The zoom-in view (a) of the visible cell from the black points with respect to the red viewpoint in (b) which locally connects the discrete points. By marching the visible points (b, c, d), a curved skeleton inside the unstructured points is extracted (e). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

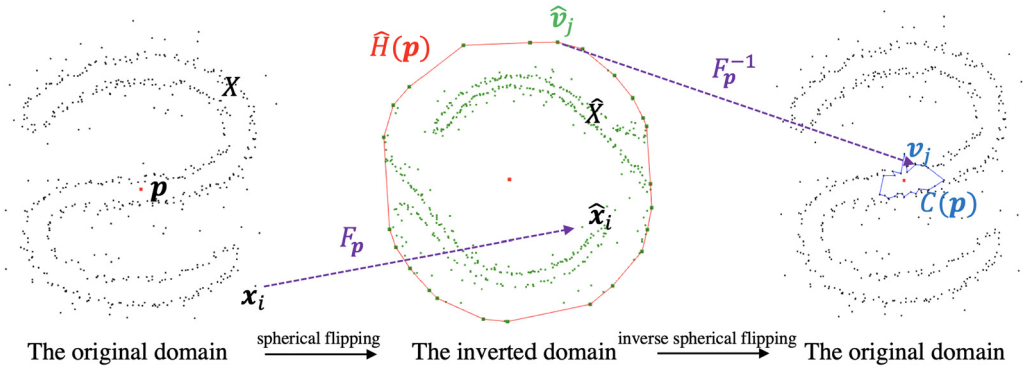

Fig. 2. The definition of the visible cell. The points (black dots) in the original domain are mapped onto the inverted domain by a spherical flipping. Then a convex hull (red lines) of the points in the inverted domain (green dots) is constructed. The convex hull is mapped onto the original domain, forming the visible cell (blue lines) from the viewpoint (the red dot).

3. Visible cells from points

Our goal is to directly extract the curved skeleton from unstructured points. The challenging part is mostly from the missing connectivity information of points. Therefore, we hope to locally build the topological connections between points, which will be helpful in locally finding the center points on the skeleton, as illustrated in Fig. 1(a). However, it is tedious to select and order the raw points for producing a simple polygonal or polyhedral region. For example, a greedy strategy by searching and connecting the points with the smallest Euclidean distances may lead to a general graph rather than a closed region. Instead of using Euclidean distance as the metric, we propose to use visibility as the key criterion to locally determine the connection.

In this section, we first introduce the visible cells which will be used to define the points on the curved skeleton. Then we test the HPR operator to generate visible cells inside raw points and improve its robustness in the presence of noises and missing data. At the same time, we define a feature to characterize the quality of the skeleton points based on the visible cell. In the following, we will first illustrate the visible cells in 2D and then discuss its 3D counterpart.

3.1. Visible cells based on the HPR operator

Suppose we are extracting the curved skeleton from a point set with n points $X = \{\mathbf{x}_i\}_{i=1,\dots,n}$. We define a visible cell $C(\mathbf{p})$ with respect to the viewpoint $\mathbf{p}$ to be the polygon connecting a subset of the points $V(\mathbf{p}) = \{\mathbf{v}_j \in X\}_{j=1,2\dots,m}$, where each connection between $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ in $V(\mathbf{p})$ is not occluded by any point in X if it is viewed from $\mathbf{p}$. From the definition shown in Fig. 1(a), we can observe that given a viewpoint inside the unstructured points, it is possible to locally connect a subset of the points to form a simple polygon. A visible cell contains the visible region from the viewpoint. Suppose we are looking around at the viewpoint to see the shape from inside, the visible region is close and continuous. In other words, the visible cell has a closed boundary touching the points sampled on the contour of the shape.

Because the points cannot occlude each other, it is not trivial to compute visible cells if the input is a set of points. An HPR operator Katz et al. (2007) was proposed to select the visible points without surface reconstruction or normal estimation. We use this method to compute the visible cells. The operator includes two steps. Given a viewpoint $\mathbf{p}$, each point $\mathbf{x}_i$ is first mapped by a spherical flipping $F_{\mathbf{p}} : X \to \hat{X}$ with

$$\hat{\mathbf{x}}_i = F_{\mathbf{p}}(\mathbf{x}_i) = \mathbf{x}_i + 2(r - ||\mathbf{x}_i - \mathbf{p}||)\frac{\mathbf{x}_i - \mathbf{p}}{||\mathbf{x}_i - \mathbf{p}||}, \tag{1}$$

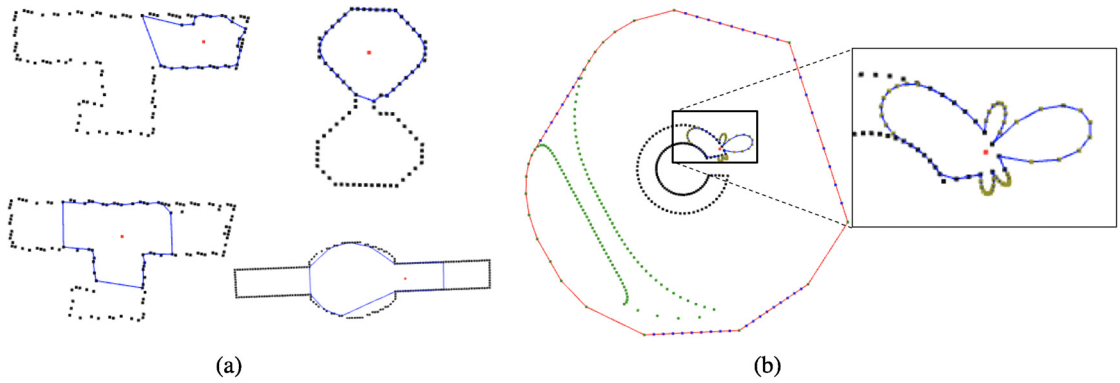

Fig. 3. (a) Examples of visible cells. (b) An undesired structure is observed if we interpolate the convex hull in the inverted domain.

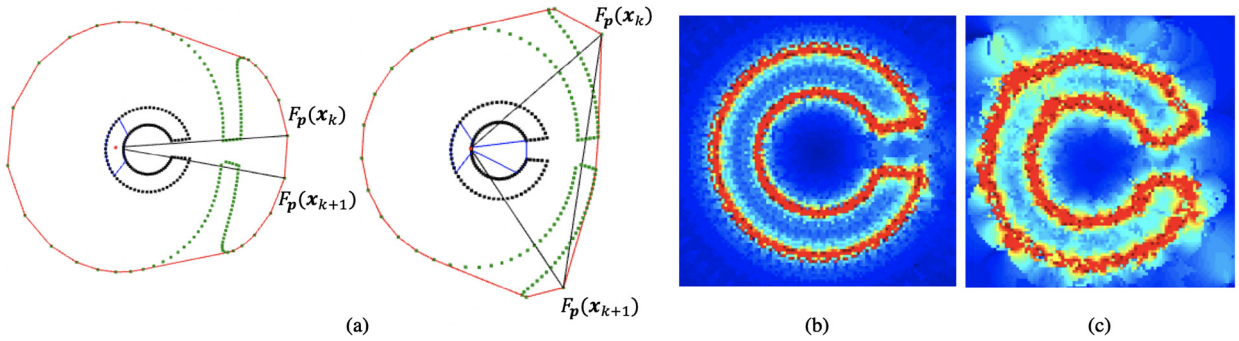

Fig. 4. (a) The edge $F_{\mathbf{p}}(\mathbf{x}_k)F_{\mathbf{p}}(\mathbf{x}_{k+1})$ in the inverted domain is longer if the viewpoint is moving near $\mathbf{x}_k\mathbf{x}_{k+1}$. (b) The color-coded map of the visibility score, where low visibility score is mapped to red and high visibility score is mapped to blue. (c) The map of the visibility score when the points are perturbed by noise.

where the radius r is set as the maximum distance from the input points to the viewpoint. We call the mapped domain $\hat{X}$ *an inverted domain*. Then the convex hull $\hat{H}(\mathbf{p})$ of the points $\{\mathbf{x}_i\} \cup \mathbf{p}$ is calculated and the points locating on the convex hull are selected as the visible points. In Katz et al. (2007), the points whose images on the inverted domain are not on the convex hull $\hat{H}(\mathbf{p})$ are tagged as hidden points and are directly removed. In this work, in order to get a visible cell with additional connection information, we do not only keep the indices of visible points. We further add a step to map the convex hull back to the original domain, because the convex hull also encodes the connectivity of the points on it. In this way, we obtain the visible cell $C(\mathbf{p}) = F_{\mathbf{p}}^{-1}(\hat{H}(\mathbf{p}))$. We illustrate the steps in Fig. 2.

3.2. Visibility score

We show different visible cells from point samples on a curve in Fig. 3(a). From the test, we find that a visible cell well captures the local structure of the shape. We also test to map the interpolated points on the convex hull $\hat{H}(\mathbf{p})$ back to the original domain, as shown in Fig. 3(b). Although the interpolation leads to a finer visible cell, we find unnecessary detailed structures from a zoom-in view. Therefore, we do not interpolate the convex hull for a finer cell.

As shown in Fig. 3, the visible cell locally connects the points along the boundary. It is possible to measure whether a viewpoint is close to the boundary of the shape. Suppose a camera is located at the viewpoint and is looking around to see all visible points on the boundary of its visible cell, the distance between the viewpoint and each point on the visible cell measures how well each visible point is observed. Because the curved skeleton is placed near the middle of the shape, we locate points on the skeleton with a balanced view to each visible point. Therefore, we define a feature measuring whether a viewpoint has a balanced view of its visible cell, which is called *the visibility score* $S(\mathbf{p})$.

As we locally get a closed visible cell for every viewpoint, a straightforward way to compute the visibility score $S(\mathbf{p})$ is to measure the distribution of the distance between the viewpoint $\mathbf{p}$ and its visible point on the cell $C(\mathbf{p})$. However, due to the variation of the size and anisotropy of the visible cell, it is not hard to normalize the distance measurement. Looking into the spherical flipping $F_{\mathbf{p}}$, we propose a different way to quantify the visibility score $S(\mathbf{p})$. Note that if a viewpoint is close to the boundary of the shape, it is also close to one edge of its visible cell. If $\mathbf{x}_k\mathbf{x}_{k+1}$ is the closest edge on $C(\mathbf{p})$, we also know its image on the inverted domain $\hat{\mathbf{x}}_k\hat{\mathbf{x}}_{k+1} = F_{\mathbf{p}}(\mathbf{x}_k\mathbf{x}_{k+1})$. When a viewpoint is moving toward the edge $\mathbf{x}_k\mathbf{x}_{k+1}$, the spherical flipping will drastically enlarge the image of the edge on the inverted domain, as shown in Fig. 4(a). Based on

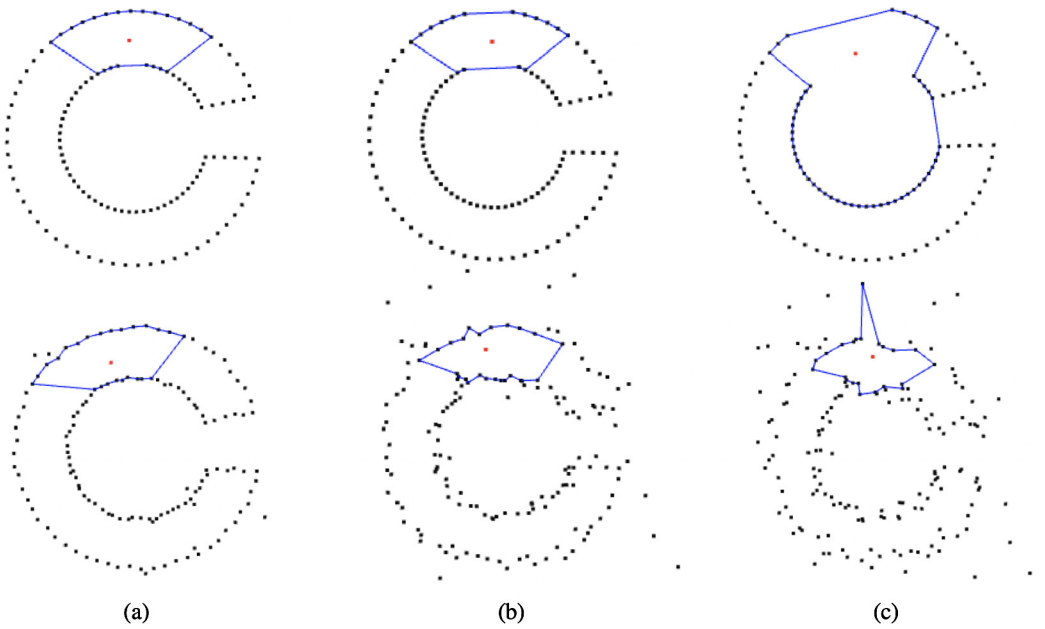

Fig. 5. (a, b, c) The visible cells with increasing missing data (upper row) and noisy data (lower row).

this observation, we define the visibility score as the ratio between the edge length in the original domain and the inverted domain

$$S(\mathbf{p}) = \frac{||\mathbf{x}_k - \mathbf{x}_{k+1}||}{||F_{\mathbf{p}}(\mathbf{x}_k) - F_{\mathbf{p}}(\mathbf{x}_{k+1})||}, \tag{2}$$

where $\mathbf{x}_k\mathbf{x}_{k+1}$ is the closest edge on the visible cell $C(\mathbf{p})$ to the viewpoint $\mathbf{p}$ and $F_{\mathbf{p}}$ is the spherical flipping defined in Eqn. (1). In Fig. 4(b), given a set of 2D points, we show the map of the visibility score on the plane $S(\mathbf{p}), \mathbf{p} \in R^2$. The visibility score is color-coded in the figure, where the low score is mapped to red and the high score is mapped to blue. From the figure, we can observe that the proposed visibility score is lower near the boundary of the shape and is higher near the middle of the shape.

3.3. Visible cells for imperfect data

By assuming that the points are sampled from a shape, we have defined the visible cell and the visibility score of each viewpoint. We further study the visible cells when the input points are with noisy and missing data. In Fig. 5, we gradually increase the missing points and noise in the original samples and then compute the corresponding visible cells. From the results in the upper row of Fig. 5, we find that with a moderate amount of missing data, the visible cell successfully fills the missing part. When the missing part is too large, the visible cell will faithfully regard it as an opening port. If no point can be seen from the viewpoint, the visibility cell still well connects the points to close the gap. Otherwise, it may grow to connect other points that are visible from the gap. Therefore, the visible cell is capable of filling the missing data or closing the gap, depending on different cases. This property is used to extract the curved skeleton in the presence of missing data. We will discuss our strategy in Sec. 4.

For noisy data, the visible cells are not stable as shown in the lower row of Fig. 5. By the strict definition of visibility, if a hidden point is moving toward the viewpoint because of the added noise, the selection of the visible points may be updated and the structure of the visible cell may have a discrete change. In case the introduced noise produces a gap, the visible cell may get spiky. The discrete change of the visible cell may bring problems in the curved skeleton extraction. We, therefore, propose two strategies to alleviate the problem, as shown in Fig. 6.

The first strategy is to make visibility conditions more strict. In other words, we propose to increase the threshold in the HPR operator and only select strongly visible points to form a visible cell. As discussed in Katz et al. (2007), the convexity of the polygons in the inverted domain infers the visibility of the selected points. A marginal case is that, if a vertex $\hat{\mathbf{x}}_k$ in the inverted domain lies on the line connecting $\hat{\mathbf{x}}_{k-1}$ and $\hat{\mathbf{x}}_{k+1}$, we regard $\mathbf{x}_k$ just not occluded by $\mathbf{x}_{k-1}$ and $\mathbf{x}_{k+1}$. If the $\mathbf{x}_k$ is perturbed so that its image $\hat{\mathbf{x}}_k$ moves inside the convex hull containing $\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_{k+1}$, it will be removed from the set of the visible points. Therefore, if $\hat{\mathbf{x}}_k$ on the convex hull forms a small angle with its neighboring vertices $\hat{\mathbf{x}}_{k-1}$ and $\hat{\mathbf{x}}_{k+1}$, preferably much smaller than 180 degrees, it will be less probable to leave the convex hull under perturbation. Therefore, we propose to set a threshold θ of the angles on the convex hull, where $\theta \leq 180^o$. With the threshold, we greedily remove vertices on the convex hull $\hat{H}(\mathbf{p})$ with the largest angle if it is greater than θ and update the simplified convex hull. We

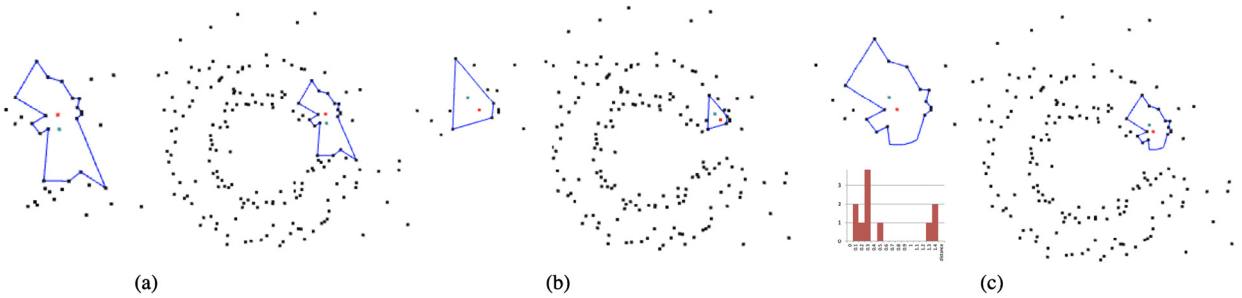

Fig. 6. For noisy points, we improve the visible cell (a) by using a strong visible cell (b) or a truncated visible cell (c). The distance distribution of the original visible cell shown in the subfigure (c) is used to set the truncated threshold.

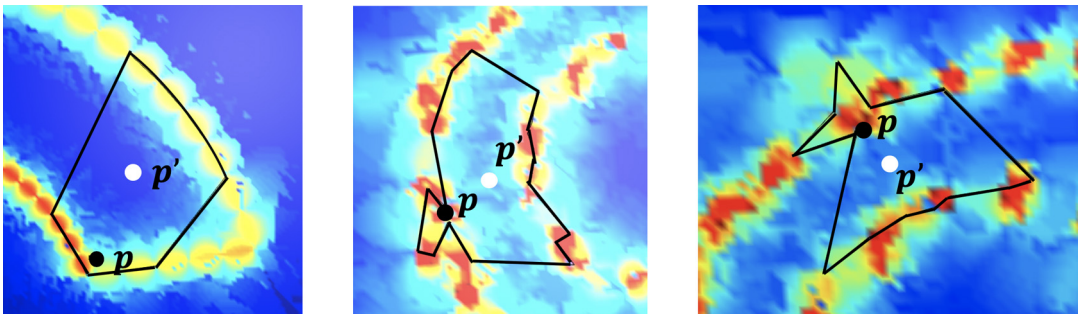

Fig. 7. The center $\mathbf{p}'$ (white dots) of the visible cell $C(\mathbf{p})$ (black lines) has higher visibility score.

iteratively remove vertices until no vertex on $\hat{H}(\mathbf{p})$ has an angle greater than θ. By mapping this improved convex hull to the original domain, we obtain a simplified visible cell, which is called *a strong visible cell*. We set the threshold $\theta = 160^o$ by default. One example is shown in Fig. 6(b). It is stable with the noise.

The other strategy is to shorten the sight from the viewpoint. As shown in Fig. 6(c), if we restrict the point in the original domain whose distance to the viewpoint is greater than a threshold d to be invisible, we can truncate the visible cell to get a stable visible region. Specifically, if the visible point satisfies $||\mathbf{x}_k - \mathbf{p}|| > d$, we project the visible point to $\mathbf{p} + d \cdot (\mathbf{x}_k - \mathbf{p})/||\mathbf{x}_k - \mathbf{p}||$. In this case, if the noisy points open a gap to select a faraway point as a visible point, the spike will be truncated, prevent the visible cell with large updates under perturbation. We call this variant *a truncated visible cell*. In order to decide the threshold d, we compute the distances from the visible points to the viewpoint in the original domain and form a histogram of the distances. In practice, we use the 75%-quantile of the distance as the threshold d and one example is shown in Fig. 6(c). From Fig. 6, we find that both the strong visible cell and the truncated visible cell alleviate the problem due to the noisy input. It is relatively costly to compute for the strong visible cell due to the topological update of the cell, especially in 3D cases. We, therefore, suggest using the truncated visible cell for the skeleton extraction.

3D cases The construction of the visible cells for 3D points is similar to 2D cases. The only difference is that a 3D convex hull algorithm produces a convex mesh instead of a convex polygon in 2D cases. The visible cell in the original domain is also represented as a mesh. The computation of the visibility score in 3D is adapted by using the area of the closest triangle instead of the edge length in Eqn. (2).

4. Visibility-driven skeleton extraction

4.1. Geometrical embedding

With the computed visible cell, we are ready to design a method to extract skeletons from unstructured points. As shown in Fig. 1, the key idea is to move the visible cell inside the points and track its trajectory. We recover the topology of the curved skeleton by checking the neighborhood relationship between different visible cells. The geometric information of the skeleton point is computed by finding a viewpoint with a high visibility score inside the visible cell.

Let us first talk about the geometrical embedding of the skeleton points. Because the visibility score well measures the visibility if the viewpoint is in the middle of the shape, the geometry of the point on the skeleton can be determined by locating the points at the place with a high visibility score. Specifically, we are solving this following subproblem: given a viewpoint $\mathbf{p}$, we optimize its location $\mathbf{p}'$ for a higher visibility score $S(\mathbf{p}')$. Because the topological connection of the visible cell $C(\mathbf{p})$ will be updated when $\mathbf{p}$ is optimized to a new place, the visibility score $S(\mathbf{p}')$ is not continuous with respect to the coordinate of $\mathbf{p}$. We need to rely on a local search or a random search method to locally optimize the viewpoint. In

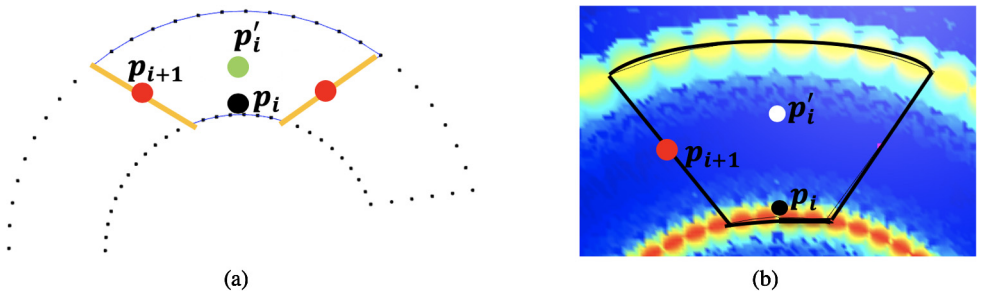

Fig. 8. (a) The next viewpoint (purple dots) is on the cap (orange lines) of the visible cell viewing from $\mathbf{p}_i$ (the red dot). The center of the visible cell $\mathbf{p}_i'$ is set as the candidate point on the skeleton. (b) The center of the cap has a higher visibility score.

practice, we find that the geometric center of the visible cell commonly has a higher visibility score, as shown in Fig. 7. Therefore, we suggest to directly use the center of the visibility cell to improve the placement of the viewpoint. Specifically, we solve the subproblem with two steps. First, we compute the visible cell $C(\mathbf{p})$ from the viewpoint $\mathbf{p}$. Then, we update the viewpoint to the center of the visible cell as a better location of the skeleton point $\mathbf{p}'$. Suppose the visible cell $C(\mathbf{p})$ consists of m ordered visible points $\{\mathbf{v}_i\}_{i=1,\dots,m}$, we use the length-weighted average of the middle point on each edge of the visible cell as its mass center

$$\mathbf{p}' = \frac{\sum_{i=1}^{m} ||\mathbf{v}_i - \mathbf{v}_{i+1}||(\mathbf{v}_i + \mathbf{v}_{i+1})/2}{\sum_{i=1}^{m} ||\mathbf{v}_i - \mathbf{v}_{i+1}||}. \tag{3}$$

We could iteratively take these two steps, or add a local search for a location with higher visibility score from the center $\mathbf{p}'$ for a better quality of the skeleton point. In practice, we find that one iteration of the geometric update with a local search is sufficient. There are cases that the visible cells are concave. However, in practice, we find the mass center $\mathbf{p}'$ rarely locates outside the cell. Especially when we are using the truncated visible cell to suppress the noisy visible cell, the mass center $\mathbf{p}'$ well represents the local center of points.

4.2. Topological operations

In order to connect the centers of visible cells, we propose to extract skeletons based on a region growing method, which consists of seeding, region growing and topological connection with termination conditions.

Seeding To start the marching of the visible cells, we need to first initialize a viewpoint as the seed. Because the curved skeleton is inside the shape, we suppose to keep the visible cells inside the points. Therefore, the seed is manually specified to be inside the shape which is approximated by its sample points. As we only need to specify the seed once, it is affordable to interactively find the seed by using a simple user interface. Note that the seed does not need to be perfectly in the middle of the shape. As long as it is inside the shape, our algorithm can start and optimize the candidate points on the skeleton.

Region growing Once a seed $\mathbf{p}_0$ inside the shape is specified, we start the region growing. We move the visible cells along the cavity formed by the unstructured points and grow the skeleton as the visible cells move. Specifically, the following subproblem is solved in this step: suppose a viewpoint $\mathbf{p}_i$ is being visited, we need to find a candidate skeleton point $\mathbf{p}_i'$ from its visible cell and sample the next viewpoint $\mathbf{p}_{i+1}$ to keep looking at different parts of the unstructured points.

This subproblem is solved in two phases. In the first phase, we compute the visible cell $C(\mathbf{p}_i)$ from the viewpoint $\mathbf{p}_i$. Then we update the candidate point $\mathbf{p}_i'$ on the skeleton to be the center of the visible cell $C(\mathbf{p}_i)$ by using Eqn. (3). In the second phase, we aim to find the next viewpoint $\mathbf{p}_{i+1}$. By observing the visible cell, we find that the edges of the visible cell have two parts. One part connects sampled points along the boundary of the shape and the other part links points that are not neighbors on the shape, which are called *the caps* of the visible cell, as shown in Fig. 8(a). We set the next viewpoint $\mathbf{p}_{i+1}$ to be at the center of the cap to start the next round of marching. The remaining task is to identify the caps on the visible cell. We take the visibility score as the key feature to characterize the cap edge, because the cap edge is not on the boundary of the shape and its center has a high visibility score, as illustrated in Fig. 8(b). Another feature of the cap edge is its length. Because cap edges do not connect neighboring points on the shape, their length is relatively longer than other edges. Therefore, we first set a threshold of the edge length to filter short edges on the visible cell. In this case, we do not need to compute the visibility score for each edge to reduce the computational cost. After short edges are filtered, we query for the visibility score by Eqn. (2). If the visibility score of a long edge is high, we regard it as a cap and set the next viewpoint $\mathbf{p}_{i+1}$ at the center of the cap. Starting from the seed viewpoint $\mathbf{p}_0$, the points $\mathbf{p}_i'$ on the skeleton are iteratively computed until no new visible cell can be found.

Topological connection In order to build a curved skeleton, we need to connect the candidate points $\mathbf{p}_i'$ computed from the region growing step. Although we are clear about how to find the candidate points $\mathbf{p}_i'$, two problems about the topological

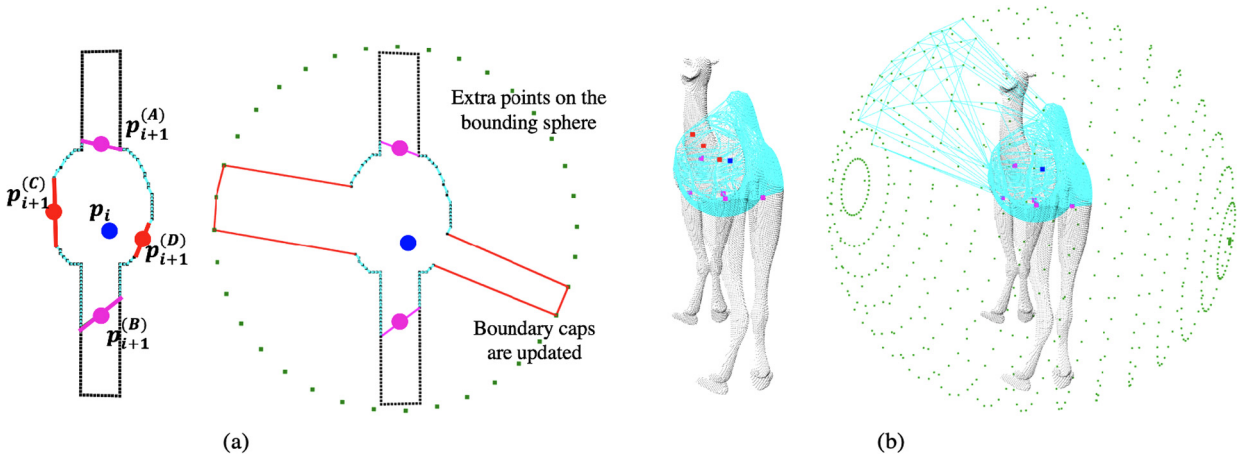

Fig. 9. (a) With incomplete points, the visible cell $C(\mathbf{p}_i)$ (shown in blue lines) has four caps corresponding to the next candidate viewpoints $\mathbf{p}_{i+1}^{(A)}, \mathbf{p}_{i+1}^{(B)}, \mathbf{p}_{i+1}^{(C)}$ and $\mathbf{p}_{i+1}^{(D)}$. We test the visible cell with extra points (shown in green) on the bounding sphere of the input points. The caps which touch the boundary (shown in red lines) update due to extra points and are regarded open to the outside. We only grow the visible cells according the inside caps (shown in purple lines) at $\mathbf{p}_{i+1}^{(A)}$ and $\mathbf{p}_{i+1}^{(B)}$. (b) An example to illustrate the 3D cases. The centers of candidate caps on the back of the camel model (shown in red) will not trigger new visible cells, because their corresponding caps update due to the extra points on the bounding sphere. After filtering these red cap centers, we only grow the visible cells from the candidate cap centers (shown in purple) inside the neck and four limbs.

connection are still left in order to define the traversal order and clean the skeleton. We first discuss the traversal order. If a visible cell $C(\mathbf{p}_i)$ has multiple caps, we may have more than one branch from the currently visited skeleton point $\mathbf{p}_i'$. In this case, both a broad-first or a depth-first traversal will work. In the implementation, we take a broad-first traversal in growing the skeleton. One problem with the traversal is that, for example, a visible cell $C(\mathbf{p}_i)$ is generated from the previous cell $C(\mathbf{p}_{i-1})$, but the detected cap does not share the historical information of $C(\mathbf{p}_{i-1})$. In this case, we need to distinguish the next visible cell $C(\mathbf{p}_{i+1})$ from its precede $C(\mathbf{p}_{i-1})$. Our solution is to check if the next viewpoint $\mathbf{p}_{i+1}$ is inside the preceding visible cell $C(\mathbf{p}_{i-1})$. Only the next viewpoints outside its preceding visible cell will be regarded to see the unvisited points. They will be used to grow new visible cells to cover the unvisited region inside the points.

Termination conditions Having defined the traversal order, we connect the center of the visible cells with the traversal order. In the beginning, there is no visited cap of the preceding cell. We track all the new viewpoints from the caps of the seed cell. As we are using a breadth-first traversal, we keep storing the visible cell corresponding to a growing cap at each marching front. Each time one visible cell at the front is grown, we check whether it is intersected with other visible cells at the marching front. If the new visible cell $C(\mathbf{p}_j)$ is intersected with one visible cell $C(\mathbf{p}_k)$ tracked at the marching front, we connect the center of two cells and the new edge $\mathbf{p}_j'\mathbf{p}_k'$ closes a loop on the skeleton. After a loop is detected and closed, we stop finding the next viewpoints for these cells if they do not have other caps. If a growing skeleton creates a visible cell at the end of one branch, the new cell only has one cap and the cap is detected as a visited cap. In this case, we also stop adding a new viewpoint at this branch. We terminate the growing of the skeleton when no new visible cell can be found. Note that in terms of the termination conditions, we do not count to check if all the points are seen by the collection of the moving visible cells, because for a noisy point cloud, not all the points are covered by the visible cells. It is not trivial to use a threshold of the number of visited points to determine the termination. After the skeleton is extracted, we could interpolate on the skeleton and optimize the interpolated points to the centers of their corresponding visible cells in case we prefer a dense sampling on the skeleton.

For the points with missing data, the skeletons may grow out of the interior region of the shape from a large gap. Although it only happens when lots of points are missing, we also find a way to terminate the growing outside the points. When we are growing a visible cell for incomplete points, it may have candidate caps that near the large gap, as shown in Fig. 9. If the centers on these caps are activated to be the next viewpoints, the next cell centers may be outside the points. Therefore, we further filter the candidate caps to prevent growing cells outside. Specifically, we add the points on the bounding sphere of the input to test the current viewpoint $\mathbf{p}_i$. If the candidate caps are updated because of the extra points on the bounding sphere, we regard these caps open to the outside and do not grow new cells on these caps. In case there are holes inside the points, we suggest to add extra points to indicate the holes. We test this method with different datasets and find it works well for 3D points with large missing parts.

3D cases Similarly, in order to extract skeletons from 3D points, we replace the computation on closed polygons with its counterpart on closed meshes. For example, in the computation of the cell center, we replace the length-weighted average of the edge centers (Eqn. (3)) with an area-weighted average of the face centers. In the cap detection part, we group neighboring large triangles with similar face normal, find its center and use the visibility score of the center to check if

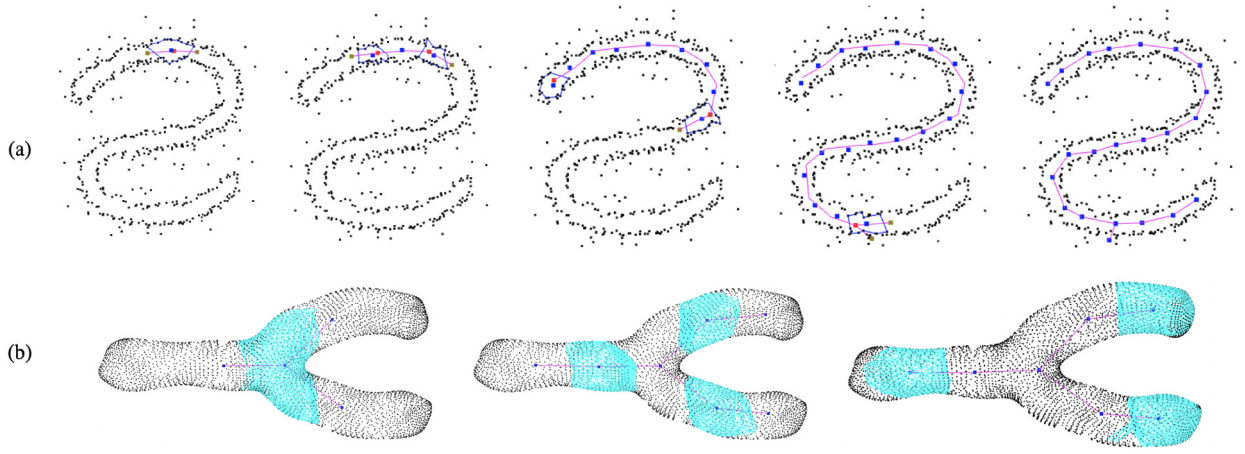

Fig. 10. From left to right: the growing process and the extracted skeleton from a 2D point cloud (a) and a 3D point cloud (b).

it is a cap. Because the visible cell in 3D is also a closed polyhedron, it is fast and reliable to compute the corresponding normals.

5. Results

We show the experimental results with 2D and 3D data in this section. In our experiment, the default parameters are set as follows. The distance threshold of the truncated visible cell is set as the 75%-quantile of the distance distribution from the visible points to the viewpoint. The edge length and visibility score threshold in the detection of the cell caps are $5h$ and 0.015, where h is the average gap between the points. We find three closest points for each point and average these three closest distances as the point gap h_i, then h is precomputed by averaging all point gaps $\{h_i\}$. For 3D cases, the cap size threshold is set to sh^2, where $s = 10$ by default.

In Fig. 10, we show the extracted skeleton from a 2D point set and a 3D point, as well as the growing process of the skeleton. We can observe that a skeleton is gradually grown from the seeding point that captures the structure of the shape. For the noisy input in Fig. 10(a), the noise may introduce a tiny branch to the skeleton, which can be removed after post-processing.

We test our method with noisy and incomplete data, in comparison with a well-sampled data. As shown in Fig. 11, the structure of the points is still well tracked by the visibility-driven skeleton. Note that in Fig. 11(c), we also have outliers in the unstructured points. Our method works well in the presence of outliers. In the lowest row, we also show an example with a closed loop.

For 3D points with large portion of missing data, we generally tune two parameters to find a good curved skeleton, including the threshold d to define the truncated visible cell and the cap size threshold s to detect the candidate caps. In Fig. 12, we experiment our method with different involving parameters. We find that the visible cells may not grow well if the visible cells are not truncated or over-truncated in the presence of strong noises. For the cap size threshold, small s may lead to many noisy skeletons in small size while large s may only capture the skeletons to the large trunk of the points. Users may find a good result by slightly tuning d and s around the recommended settings (d : 75%-quantile, $s = 10$).

We also compare our method with the l_1-medial skeleton Huang et al. (2013) and the Laplacian-based contraction method Cao et al. (2010), as shown in Fig. 13. We run our test with a single thread on a desktop with an Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz and its memory is 8G. We tune the parameters as suggested in Huang et al. (2013) and Cao et al. (2010) for a fair comparison. From the results, we see that the skeleton from our method successfully reproduces the structure of the data and its visual quality is comparable with previous methods. In addition, with unoptimized codes, our method has a better performance in terms of computational cost, because no fitting of primitives, iterative updating of the mass center or linear system solving is required in our method.

In Fig. 14, we experiment our method with the scanned dataset. Different levels of noises and portions of missing parts are included in the 3D points. We find our method works well for the real-world scans. Because the visible cells well represent the local cavity formed by the points, the skeleton points well reside in the concave regions. There are cases that the visible cell may contain points far from the local cavity because of the large missing portion of the points, which may distort the cell centers. For example, the skeleton points around the camel feet are slightly outside the points. However, because most points of the visible cell are in the same local region, the cell center only slightly deviates.

Discussion In this work, a visibility-aware skeleton is extracted by moving visible cells inside the points. The visible cell can be considered as a generalization of the circular or spherical primitive used in the medial axis transform. While growing a circular cell to touch the points requires an expensive search in the space of circles, the visibility cell in this work enables

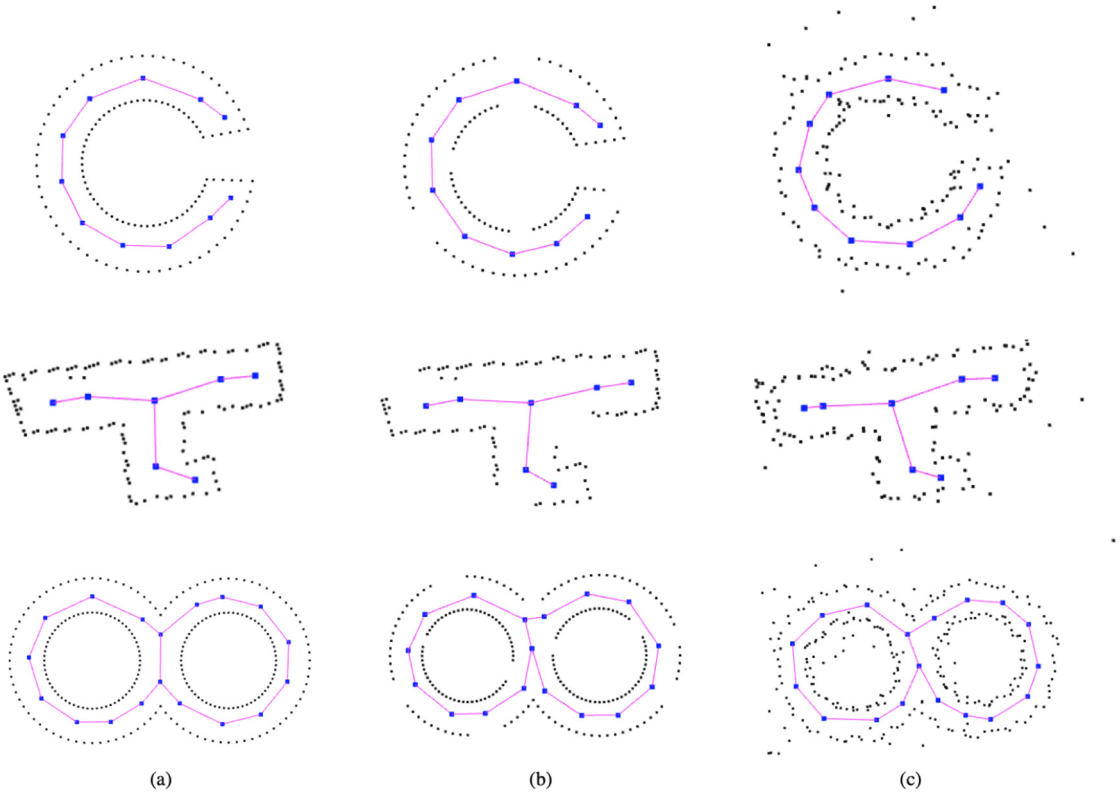

Fig. 11. The extracted skeleton from the points with missing data (b) and noise (c), in comparison with the one from a well-sampled point cloud (a).

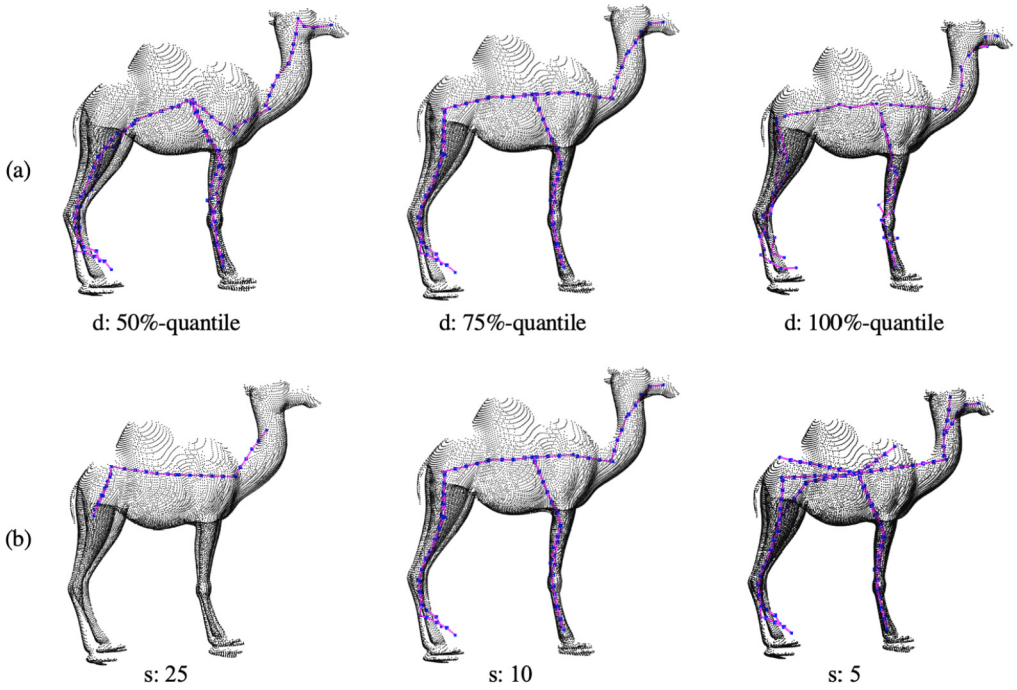

Fig. 12. Experiments with different threshold d to define the truncated visible cell (a) and cap size threshold s to detect the candidate caps (b).

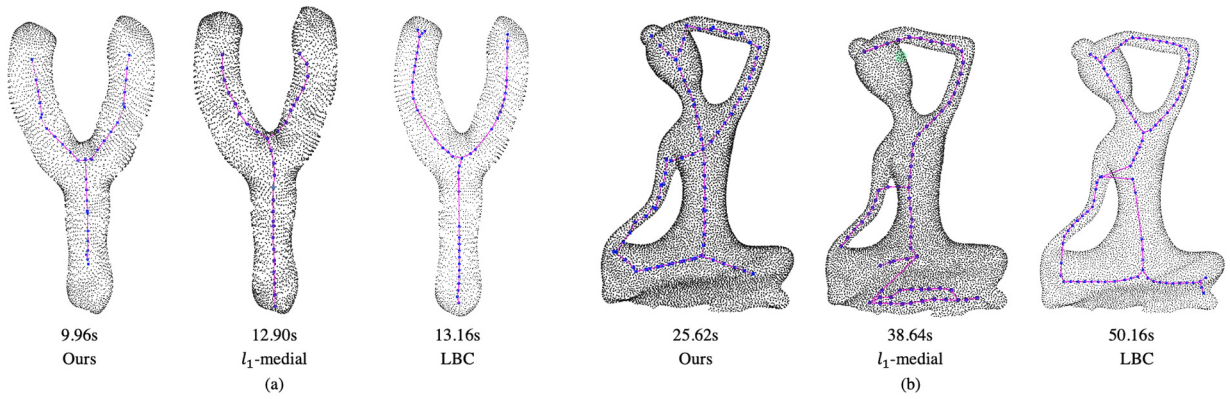

Fig. 13. The extracted skeletons from two 3D points (a,b) by using different methods. From left to right in each subfigure: the proposed visibility-driven method, the l_1-medial skeleton (l_1-medial) Huang et al. (2013) and the Laplacian-based contraction method (LBC) Cao et al. (2010). We also report the time cost in seconds of obtaining the skeleton.

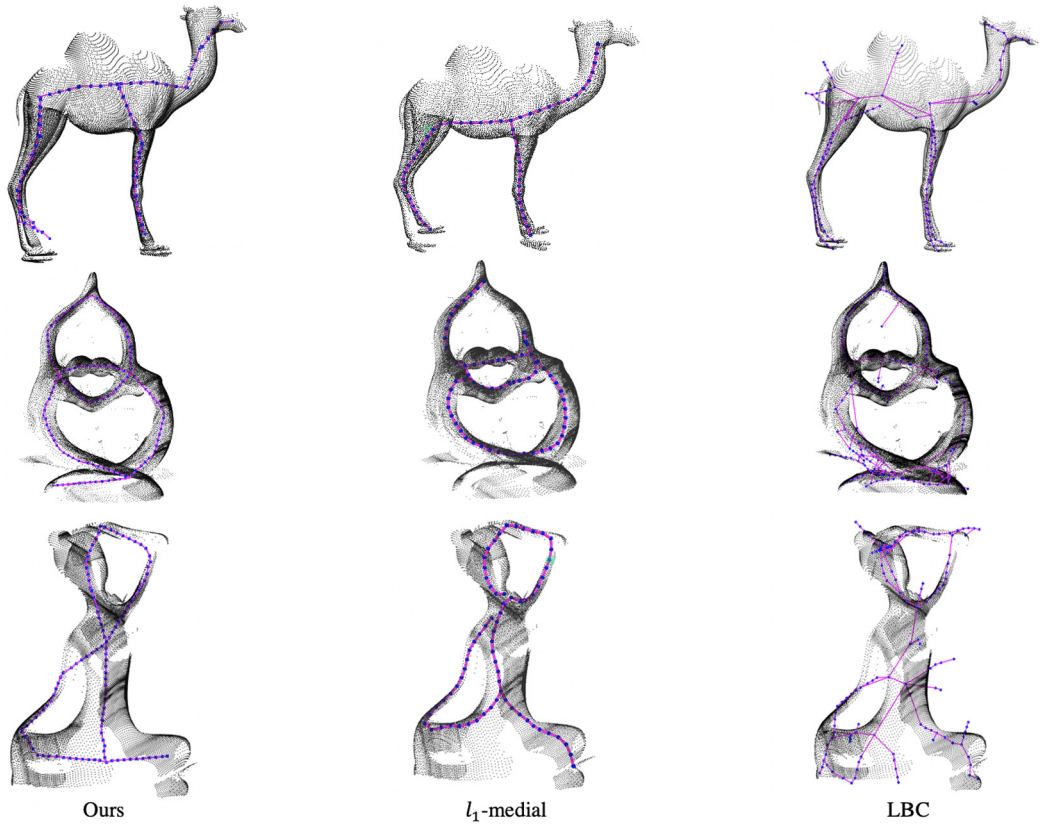

Fig. 14. The extracted skeletons from real-world scanned dataset including large noises and missing part. From left to right: the proposed visibility-driven method, the l_1-medial skeleton (l_1-medial) Huang et al. (2013) and the Laplacian-based contraction method (LBC) Cao et al. (2010).

a quick selection of the touching points using visibility as a guide. It also quickly finds the cell center which allows a fast refinement of the viewpoint location. In addition, the circular cell is sensitive to noise and missing data, while the proposed visibility cell with its center is relatively stable.

Limitations The visibility-driven skeleton extraction method assumes an empty cavity inside the points. Only with the presence of the cavity, we can generate visible cells with a clear region. In other words, the proposed method will not work if the points are not only sampled along the boundary of the shape but also filled with the samples inside the shape, as illustrated in Fig. 15(a). The assumption of the boundary samples is not a strong restriction, because the points from most optical scanners are sampled on the shape boundary. This limitation also implies a weakness of the visibility-driven skeleton extraction in dealing with inliers. An inlier inside the points may stop the visibility cell to reach further, as shown

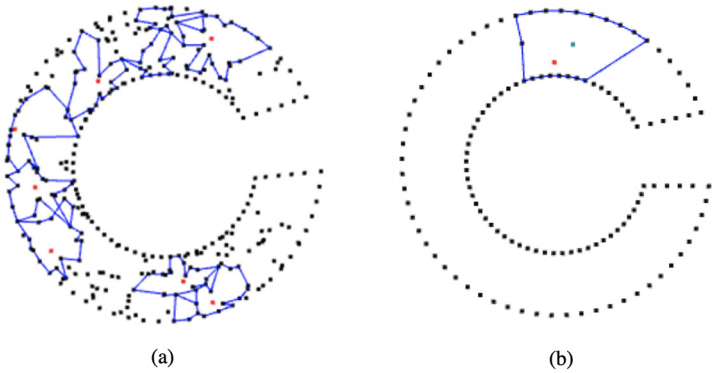

Fig. 15. (a) Our method cannot apply to points that solidly pack in a region. (b) The visible cell may also be sensitive to inliers.

in Fig. 15(b). In this case, we can group the cap edges with similar normals in the region growing phase to prevent the influence of inliers. Besides, the proposed method is not suitable for extracting skeletons for objects with degenerated structures because there is no clear visible region from the inside.

6. Conclusion

In this work, we study the visibility as a new condition for skeleton extraction with imperfect points as the input. We introduce visibility cells from the points and improve them in the presence of noisy and incomplete data. Based on the visibility information, we also design a feature that measures how well a viewpoint can see the points on the boundary and use the feature to locate points on the skeleton. A region growing method is finally developed to extract curved skeletons from unstructured points.

In the future, we will theoretically study more properties of the visible cell, in combination with the extension of the HPR operator in Katz and Tal (2015). In real-world engineering, the skeleton extraction from the raw points is an ill-posed problem because the signal-to-noise ratio is unknown. Especially for the cases with incomplete data, we do not know if it is an opening gate by design, or missing data due to occlusion. We hope to further study the visibility conditions in this case. Based on the visibility conditions, we also plan to design a user interface to further facilitate the users to address possible ambiguity.

CRediT authorship contribution statement

Lifeng Zhu: Conceptualization, Methodology, Software, Writing - original draft. **Wen Xing:** Methodology, Software, Validation, Visualization. **Aiguo Song:** Supervision. **Yongjie Jessica Zhang:** Visualization, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank anonymous reviewers for their constructive comments. This work has been supported by the National Key Technologies R&D Program under Grants No. 2019YFC0119303, the NSFC under Grants No. 61502096, 61773219, 61673114 and the Zhishan Youth Scholar Program of SEU.

References

Au, O.K.C., Tai, C.L., Chu, H.K., Cohen-Or, D., Lee, T.Y., 2008. Skeleton extraction by mesh contraction. ACM Trans. Graph. 27, 44:1–44:10.
Blum, H., 1967. A transformation for extracting new descriptors of shape. In: Models for the Perception of Speech and Visual Form. MIT Press, pp. 362–380.
Cao, J., He, Y., Li, Z., Liu, X., Su, Z., 2011. Orienting raw point sets by global contraction and visibility voting. Comput. Graph. 35, 733–740.
Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., Su, Z., 2010. Point cloud skeletons via Laplacian based contraction. In: 2010 Shape Modeling International Conference, pp. 187–197.
Cornea, N.D., Silver, D., Min, P., 2007. Curve-skeleton properties, applications, and algorithms. IEEE Trans. Vis. Comput. Graph. 13, 530–548.
Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.L., 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 203–212.
Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., Chen, B., 2013. L1-medial skeleton of point cloud. ACM Trans. Graph. 32, 65:1–65:8.
Jacobson, A., Deng, Z., Kavan, L., Lewis, J., 2014. Skinning: real-time shape deformation. In: ACM SIGGRAPH Courses.
Kaleem, S., Stephen, P., 2009. Medial Representations: Mathematics, Algorithms and Applications. Springer.

Katz, S., Tal, A., 2015. On the visibility of point clouds. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1350–1358.
Katz, S., Tal, A., Basri, R., 2007. Direct visibility of point sets. ACM Trans. Graph. 26, 24:1–24:11.
Kligler, N., Katz, S., Tal, A., 2018. Document enhancement using visibility detection. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), pp. 2374–2382.
Liu, L., Zhang, Y., Liu, Y., Wang, W., 2015. Feature-preserving T-mesh construction using skeleton-based polycubes. Comput. Aided Des. 58, 162–172.
Ma, J., Bae, S.W., Choi, S., 2012. 3D medial axis point approximation using nearest neighbors and the normal field. Vis. Comput. 28, 7–19.
Machado e Silva, R., Esperança, C., Marroquim, R., Oliveira, A.A.F., 2014. Image space rendering of point clouds using the HPR operator. Comput. Graph. Forum 33, 178–189.
Mehra, R., Tripathi, P., Sheffer, A., Mitra, N.J., 2010. Visibility of noisy point cloud data. Comput. Graph. 34, 219–230. Shape Modelling International (SMI) Conference.
Ogniewicz, R., Ilg, M., 1992. Voronoi skeletons: theory and applications. In: Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 63–69.
Peters, R., Ledoux, H., Biljecki, F., 2015. Visibility analysis in a point cloud based on the medial axis transform. In: Eurographics Workshop on Urban Data Modelling and Visualisation, pp. 7–12.
Qin, H., Han, J., Li, N., Huang, H., Chen, B., 2019. Mass-driven topology-aware curve skeleton extraction from incomplete point clouds. IEEE Trans. Vis. Comput. Graph.
Saha, P.K., Borgefors, G., di Baja, G.S., 2016. A survey on skeletonization algorithms and their applications. Pattern Recognit. Lett. 76, 3–12.
Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., 2007. On-the-fly curve-skeleton computation for 3D shapes. Comput. Graph. Forum 26, 323–328.
Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., Telea, A., 2016. 3D skeletons: a state-of-the-art report. Comput. Graph. Forum 35, 573–597.
Tagliasacchi, A., Zhang, H., Cohen-Or, D., 2009. Curve skeleton extraction from incomplete point cloud. ACM Trans. Graph. 28, 71:1–71:9.
Thiery, J.M., Guy, E., Boubekeur, T., 2013. Sphere-meshes: shape approximation using spherical quadric error metrics. ACM Trans. Graph. 32, 178:1–178:12.
Usai, F., Livesu, M., Puppo, E., Tarini, M., Scateni, R., 2015. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. ACM Trans. Graph. 35, 6:1–6:13.
Yang, B., Yao, J., Wang, B., Hu, J., Pan, Y., Pan, T., Wang, W., Guo, X., 2020. P2MAT-NET: learning medial axis transform from sparse point clouds. Comput. Aided Geom. Des. 80, 101874.
Yin, K., Huang, H., Cohen-Or, D., Zhang, H., 2018. P2P-NET: bidirectional point displacement net for shape transform. ACM Trans. Graph. 37.
Yin, K., Huang, H., Zhang, H., Gong, M., Cohen-Or, D., Chen, B., 2014. Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. ACM Trans. Graph. 33, 202:1–202:12.
Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C.L., Hughes, T.J., 2007. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Comput. Methods Appl. Mech. Eng. 196, 2943–2959.