Signal Processing 112 (2015) 4–16

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Junction-aware shape descriptor for 3D articulated models using local shape-radius variation

Jinlong Feng [a], Yu-Shen Liu [a,b,c,*], Lianjie Gong [a]

[a] School of Software, Tsinghua University, Beijing 100084, China
[b] Key Laboratory for Information System Security, Ministry of Education of China, China
[c] Tsinghua National Laboratory for Information Science and Technology, China

ARTICLE INFO

Article history:
Received 16 March 2014
Received in revised form
7 May 2014
Accepted 20 May 2014
Available online 27 May 2014

Keywords:
3D articulated shapes
Junction
Shape descriptor
Shape-radius

ABSTRACT

An articulated model is composed of a set of rigid parts connected by some flexible junctions. The junction, as a critical local feature, provides valuable information for many 3D semantic analysis applications such as feature recognition, semantic segmentation, shape matching, motion tracking and functional prediction. However, efficient description and detection of junctions still remain a research challenge due to high complexity of 3D articulated deformation. This paper presents a new junction-aware shape descriptor for a 3D articulated model defined by a closed mesh surface. The core idea is to exploit the local shape-radius variation for encoding junction information on the shape boundary surface, where the shape-radius at each point on the surface is the radius of corresponding medial balls within the shape. The presented descriptor is typically computed using a center-surround filter operator, which calculates the Gaussian-weighted average of shape-radius in the neighborhood of each point on the surface. Our descriptor is robust to articulation and can reflect the junction feature well without any explicit shape decomposition or prior skeleton extraction procedure. The experimental results and several potential applications are proposed for demonstrating the effectiveness of our method.

1. Introduction

Non-rigid shape analysis [1,2] has been receiving growing attention in many applications in computer vision, computer graphics, pattern recognition, etc. Non-rigid shapes are ubiquitous in the world and, due to their physical properties, can undergo variant deformations [1]. One simplified strategy to non-rigid shape analysis is based on *articulated models* [2–5], which assumes that the non-rigid shape is composed of a set of rigid parts connected by some flexible junctions (or named *joints*/*hinges* in some literature [2,6]). Each of the rigid parts has a certain degree of freedom to move, and junctions are relatively small compared with parts connected.

The junction, as a critical local feature, provides valuable information for analyzing 3D articulated models. Many applications of 3D semantic analysis such as feature recognition, semantic segmentation, shape matching, motion tracking and functional prediction, can benefit from automatic detection of junctions. For instance, detection of junctions and parts can improve the performance of shape segmentation of 3D articulated models [7,8]. In medical applications, some visual tasks such as vessel

* Corresponding author at: School of Software, Tsinghua University, Beijing 100084, China. Tel.: +86 10 6279 5455, +86 159 1083 1178 (Mobile); fax: +86 10 6279 5460.
E-mail addresses: therealoneisneo@gmail.com (J. Feng), liuyushen@tsinghua.edu.cn (Y.-S. Liu), lianjiegong@gmail.com (L. Gong).
URL: http://cgcad.thss.tsinghua.edu.cn/liuyushen/ (Y.-S. Liu).

http://dx.doi.org/10.1016/j.sigpro.2014.05.025
0165-1684/

tracking may rely on junction extraction results, which can facilitate diagnosis and understanding of pulmonary vascular diseases [9]. In molecular structure analysis, many flexible molecules can be regarded as 3D articulated objects with special hinge/junction sites [6], where identifying such hinge sites is a fundamental problem for protein functional prediction and structure comparison. However, efficient description and detection of junctions still remain a research challenge due to the high complexity of 3D articulated deformation.

To address this issue, this paper presents a new junction-aware shape descriptor for a 3D articulated model defined by a closed mesh surface. To encode junction information on the shape boundary surface, the core idea is to exploit the local *shape-radius* variation, where the shape-radius [10,11] at each point on the boundary surface is the radius of its corresponding medial balls within the shape. Our algorithm consists of the following steps. First, the local shape-radius at each vertex of mesh surface is approximated using the classical ray-shooting technique, which associates volumetric information of the shape to the mesh surface. Then we setup a neighboring range for each vertex on the mesh and calculate the difference of local shape-radius within this range to measure the mount of their variance. Finally, a center-surround filter operator, which calculates the Gaussian-weighted average of shape-radius variance, assigns descriptive value to each vertex as its junction-aware descriptor value. One advantage of the presented descriptor is that it is insensitive to articulation deformation and can reflect the junction feature well without any explicit shape decomposition or prior skeleton extraction procedure. The experimental results and several potential applications are proposed for demonstrating the effectiveness of our method.

The main contributions of our work can be summarized as follows.

- A junction-aware shape descriptor is presented for 3D articulated models, which is defined as local shape-radius variation for encoding junction information at each point on the boundary surface.
- A center-surround filter operator is designed for quantifying local shape-radius variation, which calculates the Gaussian-weighted average of shape-radius in the neighborhood of each point on the surface.
- Several potential applications, such as initial junction extraction and handle loop approximation, are proposed for demonstrating the capability and effectiveness of our method.

2. Related work

There are several possible approaches to solve junction detection/extraction of 3D models. One type of approaches is based on shape structure analysis [6,12], which is often used in protein structure analysis. If given a pair of proteins, their junction positions can be found through structure alignment of pairwise proteins [6]. If given a single protein without comparing with others, its junction positions can be predicted with the help of additional chemical information [12]. However, protein structures and additional non-geometric information are not explicitly available for general 3D models in computer vision and computer graphics.

Another type of approaches is based on skeleton extraction (e.g. [13,14]). Those approaches first extract the topological skeleton of a 3D model and then identify its junctions and branches based on further skeleton analysis. However, skeleton extraction is often sensitive to perturbation and noise on the object's boundary, while robust skeleton calculation is not trivial. In addition, skeleton extraction only induces the rough junction positions, while indication of final junction regions still needs to be relocated on the boundary surface. Several robust skeleton extraction methods [11,13] may provide a prior reference for the purpose of junction detection, but this remains a separate research topic. In contrast, our method directly highlights the junction regions on the boundary surface without requiring any prior skeleton extraction procedure.

The third type of approaches is based on shape segmentation. After dividing a 3D model into meaningful parts, the intersection curves where two or more parts meet (referred as “cuts” in some literature) can assist to find the candidate junctions. However, most of segmentation approaches have to face the same question of how to define parts or part boundaries [7,11,15]. The classical part-type segmentation techniques are to analyze the geometric structure of an individual shape in order to detect its parts or part boundaries (e.g. [7,8,11]). In essence, there is a strong connection between part-partitioning and skeletonizing [8,15]. When using part-type segmentation to assist junction identification, it will be a type of “chicken-or-egg” dilemma, since the definition of a part implies the identification of clear-cut junction. In contrast, our method produces a junction-aware shape descriptor without any explicit shape segmentation procedure.

The fourth type of approaches is based on shape descriptor. Our method also falls into this type. A shape descriptor is a concise representation of the shape that expresses some specific properties [4,10]. A simple realization of junction detection can be based on measuring the local surface properties of shape boundary, such as curvature [16] and gradient [17]. However, such measurements are limited by their local surface nature and are very sensitive to local surface noise. There have been some extra efforts to develop new shape descriptors for detecting important regions or features on boundary surfaces, such as distinctive regions [18,19] and salient regions [16,20]. However, most of the shape descriptors are surface-based measurements, which do not explicitly consider the volumetric information inside 3D shapes. The work most related to ours is the *shape diameter* function (SDF) [10,11,15] presented by Shamir et al. The SDF is a scalar function defined on the boundary surface of a 3D shape, but it provides a link between the local volume of the shape and its boundary through mapping volumetric information onto the surface boundary. In essence, the shape diameter is approximated instead of computing the actual medial axis. As a shape descriptor, the SDF is pose-invariant. Another existing volume-oriented surface

metric is inner distance [3], which measures the length of the shortest path between a pair of surface points within the object's volume. Our recent work extended inner distance [4] and derived some applications [21–23]. As pairwise point distances, inner distance is pose-invariant but not junction-aware.

Generally speaking, most existing shape descriptors are not specifically designed to characterize the junction features of 3D articulated models. Moreover, many of them relate to the measurements of local surface convexities or concavities, but such convex and concave features cannot completely reflect characteristics of volumetric junctions. A reasonable junction-aware descriptor should also capture the local volumetric context of 3D shapes. In contrast, this paper exploits the volume-oriented measurement (i.e. shape-radius) and quantifies its variation as a junction-aware shape descriptor.

3. Computing junction-aware shape descriptor

This section first introduces the definition of articulated shapes and links it with the local shape-radius. Then we present an algorithm for computing the junction-aware shape descriptor of 3D articulated models.

3.1. Articulated shapes and shape-radius

In this paper, the input model is dealt with as a 3D *articulated shape* defined by a closed surface of triangular mesh. Based on several previous bodies of literature [2,3,5], the articulated shape can be roughly defined as follows.

Definition 1 (*Articulated shape*). A shape A consisting of K disjoint rigid parts $R_1,\ldots,R_K$ and L flexible junctions $J_1,\ldots J_L$, such that

$$A=(R_1\cup\cdots\cup R_K)\cup(J_1\cup\cdots\cup J_L), \tag{1}$$

is called an articulated shape. Intuitively, the shape A should satisfy the following conditions:

(1) A can be decomposed into several rigid parts connected by junctions.
(2) Each junction connected by rigid parts is relatively small when compared to its connected parts.
(3) Let Φ be a transformation that changes the pose of A. Φ is roughly considered as an articulated transformation if the transformation of any part of A is rigid (rotation and translation only) and the transformation of junctions can be non-rigid or flexible.
(4) The new shape A' achieved from articulation of A is again an articulated object and can articulate back to A. This preserves the topology between the articulated shapes after articulated transformation.

Fig. 1 illustrates an articulated shape and its articulated transformation.

The idea of junction-aware shape descriptor presented in this paper is to create a type of low pass filtering to the local shape-radius measurement, which relates to the medial axis transform (MAT). We refer to [10,11] for a short review as follows. In general, the MAT represents an object by the distance of each point to the medial axis that can be intuitively thought of as the skeleton of the object. The distance from an arbitrary point on the boundary to the medial axis is the radius of the maximal ball that is completely contained within the object. This ball is also called the medial-ball, and its radius can be considered as the local shape-radius that provides an effective link from the object's volume to its boundary surface.

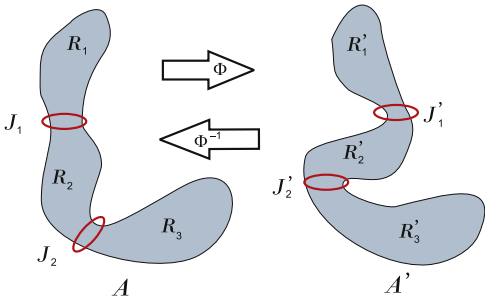

Fig. 1. Illustration of an articulated shape and its articulated transformation. An articulated shape $A=(R_1\cup R_2\cup R_3)\cup(J_1\cup J_2)$ consists of three rigid parts (R_1, R_2 and R_3) and two junctions (J_1 and J_2). The shape A can change its appearance into $A'=(R_1'\cup R_2'\cup R_3')\cup(J_1'\cup J_2')$ through an articulated transformation Φ, where Φ^{-1} denotes the reverse mapping of Φ.

Our work is mainly inspired by Definition 1, especially for that the junctions between parts are relatively small compared to the parts they connect. It implies that the shape-radius often changes a lot when an observing boundary point shifts across the junction portion. This leads to the realization that junctions can be captured by considering local shape-radius variation. Fig. 2 illustrates such observation by an example, where there is a large change of local shape-radius as the observing point crosses junction portion.

3.2. The algorithm implementation

Starting with a closed mesh model as an input, the main procedure of our algorithm consists of two steps: (1) approximating the local shape-radius and (2) computing the shape-radius variation. The details of each step are given as follows.

3.2.1. Approximating the local shape-radius

Computing the exact medial axis of a mesh surface is a complex and expensive task [10,11], and the medial axis itself is sensitive to perturbation and noise on the object's boundary. Therefore, we adopt the classical ray-shooting technique of shape diameter function (SDF) [10,11] to approximate the local shape-radius instead of computing the exact medial axis.

The first step of the algorithm can be conducted as follows. Given a mesh surface A and an arbitrary vertex $\mathbf{p}\in A$, a cone centered around $\mathbf{p}$'s inward-normal direction (the opposite direction of its normal) is created. Then some rays are shot inside this cone to the opposite side of the surface (see the green rays in Fig. 3(a)). Next, the intersection points between the rays and the boundary surface are collected, and the median or weighted average length of all intersection rays is defined as the shape

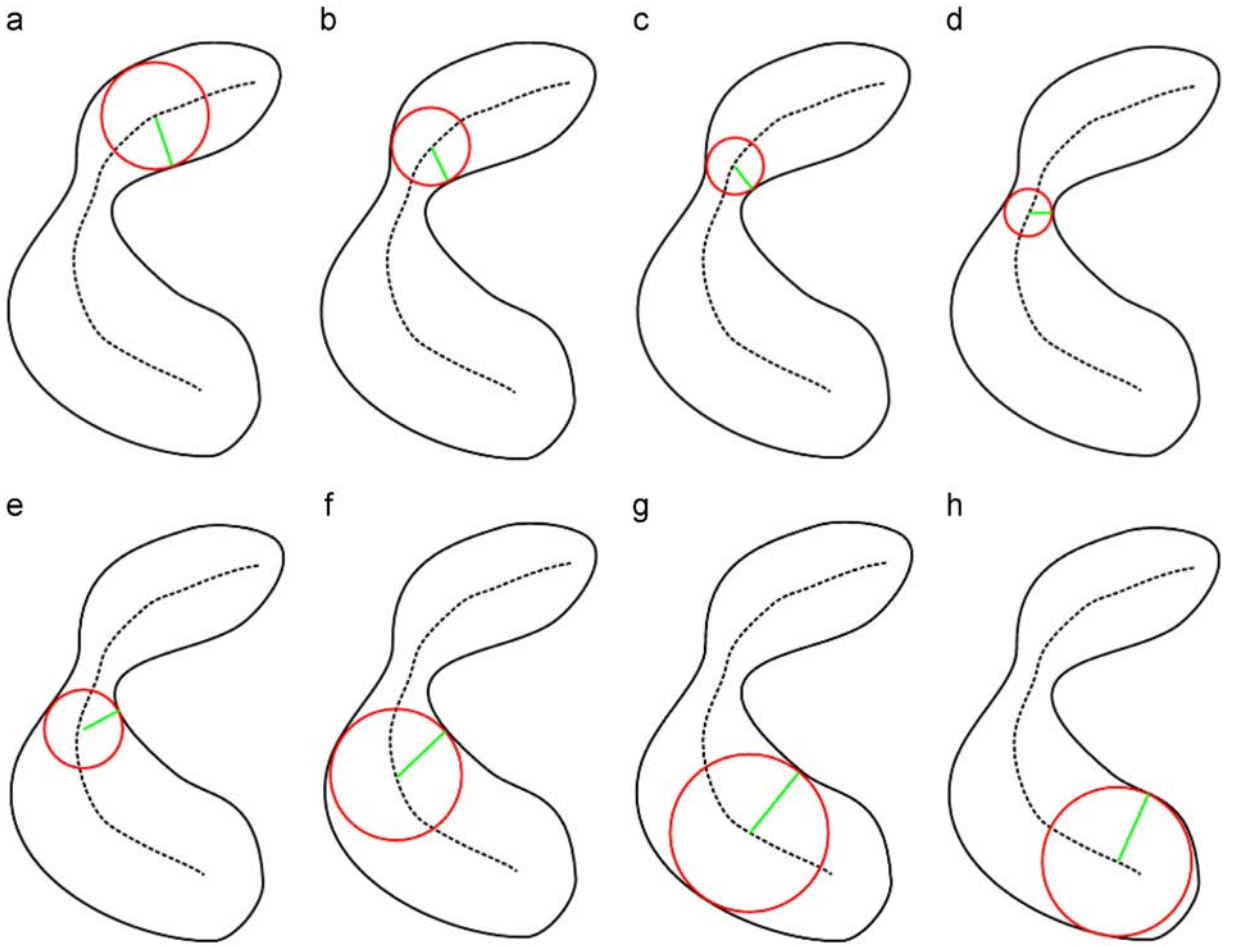

Fig. 2. Illustration of the local shape-radius (green) as observed from a boundary point travelling along the articulated shape (from (a) to (h)). Note that there is a large change in local shape-radius as the observing point crosses the junction portion. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

diameter of $\mathbf{p}$ [10,11]. Finally, the length of local shape-radius of $\mathbf{p}$, denoted by $f(\mathbf{p})$, is equal to half of its shape diameter (see the red line segment in Fig. 3(a)).

To accelerate the intersection computation, a spatial octree is built around the mesh. Moreover, the outliers removal strategy is also used for removing the rays that have no intersection with the mesh [10,11], which makes the shape-radius calculation robust to cracks and holes in non-watertight meshes.

3.2.2. *Computing shape-radius variation*

After approximating the local shape-radius of each vertex on the boundary, the next step is to develop a junction-aware shape descriptor for a 3D articulated model based on local shape-radius variation. As discussed in Section 3.1, the primary problem lies in how to measure the changes of local shape-radius between a point and its neighborhood. To address this issue, this section typically uses a center-surround filter operator to quantify the changes. Our idea is inspired by Lee et al.'s method [16], in which mesh saliency, as a measurement of regional importance, is computed using a center-surround operator on Gaussian-weighted mean curvatures. In this section, we also takes advantage of a center surround mechanism for computing local shape-radius variation, which is similar to [16]. The main reason that we use the center-surround mechanism is that it has the intuitive appeal of being able to identify these regions which are distinctive from their surrounding context. The details of the algorithm are given as follows.

For each vertex $\mathbf{p} \in A$, we first find its k-nearest neighborhood using the ANN library.[1] One can consider to define the neighborhood based on different distance metrics, such as the Euclidean distance or geodesic distance. We have tried both and found that the Euclidean distance yields similar results to geodesic distance yet faster to compute. Let $N_k(\mathbf{p})$ denote the set of k-nearest neighborhood vertices of $\mathbf{p}$.

Then we define the junction-aware shape descriptor at each vertex $\mathbf{p}$ as the Gaussian-weighted average of local shape-radius variation between $\mathbf{p}$ and its neighboring vertices:

$$\mathcal{F}(\mathbf{p}) = \frac{\sum_{\mathbf{x} \in N_k(\mathbf{p})} |f(\mathbf{x}) - f(\mathbf{p})| \exp[-\|\mathbf{x}-\mathbf{p}\|^2/(2\sigma^2)]}{\sum_{\mathbf{x} \in N_k(\mathbf{p})} \exp[-\|\mathbf{x}-\mathbf{p}\|^2/(2\sigma^2)]}, \tag{2}$$

[1] http://www.cs.umd.edu/~mount/ANN/

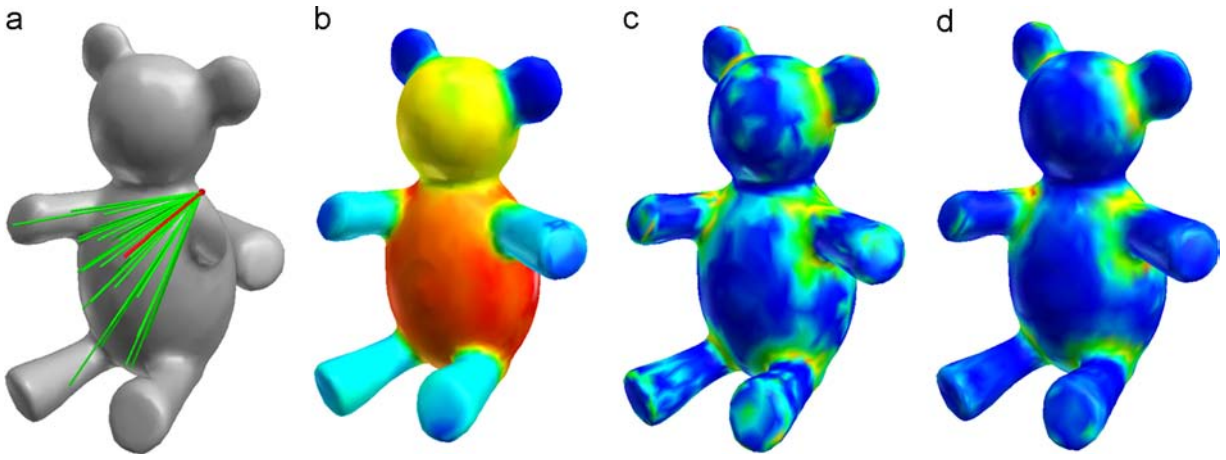

Fig. 3. Illustration of local shape-radius, shape-radius variation and junction-aware shape descriptor. (a) Teddy model with a demonstration of ray shooting from a sample vertex on the boundary, where the red line is the shape-radius is shown. (b) The shape-radius values on the model are visualized. (c) The shape-radius variation is displayed. (d) The junction-aware shape descriptor, where junctions are highlighted by the descriptor is visualized. The high values are displayed by warmer colors (red and yellow), while the low values are displayed by cooler colors (green and blue). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

where $\mathcal{F}(\mathbf{p})$ denotes the shape descriptor of $\mathbf{p}$, $f(\mathbf{p})$ is the shape-radius of $\mathbf{p}$ computed in Section 3.2.1, $|f(\mathbf{x})-f(\mathbf{p})|$ denotes the local shape-radius variation between $\mathbf{p}$ and its neighboring vertex $\mathbf{x}$, and σ is the standard deviation of the Gaussian filter. In our implementation, σ for each vertex $\mathbf{p}$ is adaptively computed as the average distance between $\mathbf{p}$ and its neighboring vertices. $\mathcal{F}(\mathbf{p})$ is insensitive to shape deformation that does not alter the volumetric shape locally, which includes articulated deformation or skeleton based movements or piecewise rigid transformation.

The following Algorithm 1, named **VertexDescriptor**, gives the pseudo-code of applying Eq. (2) to a single vertex $\mathbf{p}$, where the algorithm returns the value of junction-aware descriptor. Fig. 3 illustrates our method by a Teddy model, in which Fig. 3(b) visualizes the shape-radius values on the mesh. Fig. 3(c) shows the shape-radius variation, which is computed by the average difference of the shape-radius between each vertex and its neighborhood vertices. Fig. 3 (d) displays the values of junction-aware shape descriptor by applying the Gaussian filter to the shape-radius variation. Observe that how the junction portions are highlighted by our descriptor.

Algorithm 1. VertexDescriptor (Vertex $\mathbf{p}$).

```
Find k-nearest neighborhood vertices {q_i} of p;
k = |{q_i}|;
Compute σ as the average of Euclidean distances between {q_i}
  and p;
normalizer = 0;
sum = 0;
for i := 1 to k do
    d = ||p − q_i||;
    W = exp(−d^2/(2σ^2));
    variation = |f(p) − f(q_i)|;
    sum += W · variation;
    normalizer += W;
end for
return descriptor = sum/normalizer;
```

3.2.3. *Parameter analysis*

This section analyzes the parameters used in our junction-aware shape method. We do not try to adaptively optimize the parameters, instead we fixed them at specific values that we observe to produce fairly good results.

The implementation of Algorithm 1 includes two important parameters. The first one is the number k of neighborhood vertices. The descriptor with the small neighborhood number k highlights the *thin* junction features, while the descriptor with the large k identifies the *thick* junction features. Fig. 4 shows the results of junction indication with different k selections. In Fig. 4(a), the small k sharpens the junction features, but this may result a discontinuous feature (see the joint of teapot). In contrast, in Fig. 4(d), the large k over-smooths the junction features, but this may introduce some redundant features. We typically set $k=20$ for large meshes in our experiments.

The second important parameter is the standard deviation σ of the Gaussian filter. In Eq. (2), we are assuming a cut-off for the Gaussian filter at a distance σ that penalizes the neighborhood vertices far from $\mathbf{p}$. In practice, one may choose a large σ, such as the maximal distances between $\mathbf{p}$ and its neighboring vertices, but this often over-smooths the junction features.

4. Experimental results and applications

In this section, we apply the junction-aware shape descriptor to several geometry processing applications to illustrate its capability and effectiveness. Note that in each application, the weights, k and σ used in Algorithm 1, of the junction-aware descriptor are fixed. Also, our goal herein is not to design new algorithms, but to evaluate the capability and effectiveness induced by our junction-aware descriptor to existing applications and algorithms.

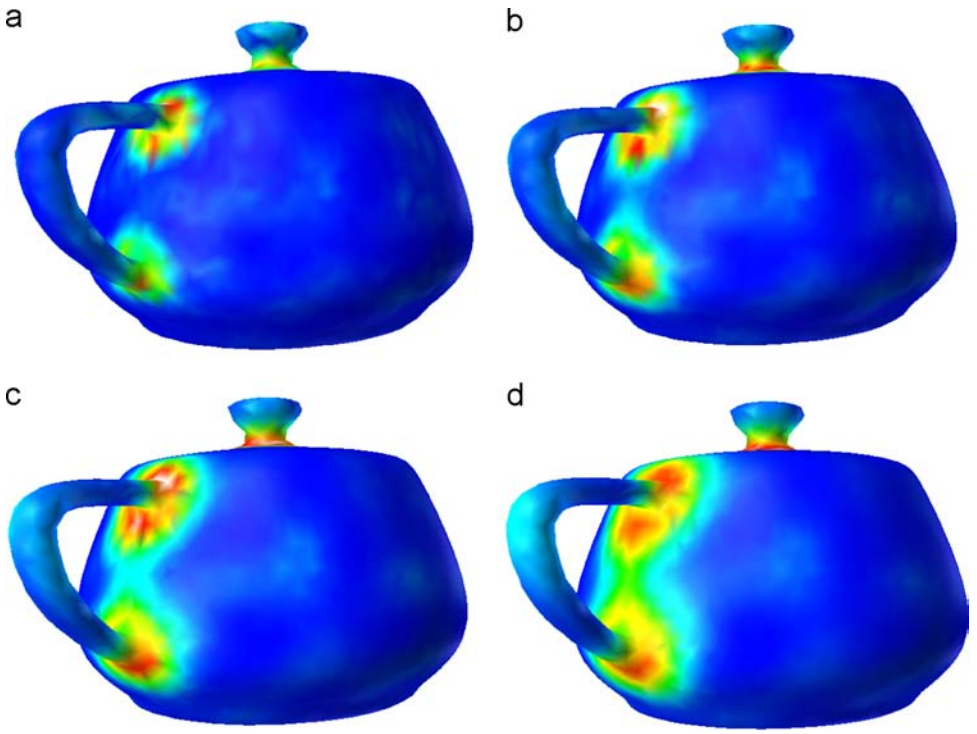

Fig. 4. The junction-aware shape descriptor is relative to the number k of neighborhood vertices. (a) $k=10$. (b) $k=20$. (c) $k=40$. (d) $k=70$. The small k sharpens the junction features, while the large k over-smooths the junction features.

4.1. Experimental setting

Our algorithm is implemented in C++ and experimented with adequate 3D articulated models. All the experiments are run on a PC with an Intel 3.3 GHz processor and 6 GB memory. The tested models are selected from two well-known 3D articulated shape databases, including Princeton Segmentation Benchmark (PSB) [24] and ISDB [10].

- *PSB* [24] is a benchmark for 3D mesh segmentation, where some object categories like humans and animals are selected for our junction identification purpose.
- *ISDB* [10] is a database of different articulated models of animals and humans containing about 104 models.

Table 1
Computational time of some models used in our experiments.

Models	# Vertices	$T\,1$[a] (s)	$T\,2$[b] (s)
Teapot	4k	31	38.7
Deer	4k	21	44.2
Ant	6.5k	19.6	116.5
Octopus	8k	50.9	99
Camel	10k	54.7	168.4
Hand	12k	82.3	193.6
Teddy	13k	94.6	166.1
Horse	15k	99.8	222.8

[a] "T 1" is the time for octree construction and cell classification during the preprocessing step.
[b] "T 2" is the time for shape-radius approximation using the ray-shooting technique.

The overall performance of our method is largely related with the efficiency of ray-shooting process at the step of approximating the local shape-radius. In the preprocessing step, we have used an axis-aligned octree at depth 5 built around the mesh to assist in intersection finding. In general, this octree construction takes little time, and consequently, it only takes a few minutes to compute the shape-radius even on large meshes. As an example for a mesh with 4k vertices, it takes around 30 s to calculate the descriptor excluding the time of loading the model, where 50 rays are shot for ray-shooting at each vertex. Table 1 lists the computational time of some examples used in our experiments, where "# Vertices" is the number of vertices of each mesh model, "T 1" is the time for octree construction and cell classification during the preprocessing step, and "T 2" is the time for shape-radius approximation using the ray-shooting technique.

4.2. Testing the robustness of our method

To verify the capability of our method, several articulated models are tested and demonstrated in Fig. 5. The left column is the original models, the middle column shows their SDF values and the right column displays the result of junction-aware descriptor. In these cases, we can see that the large values of junction-aware descriptor, highlighted by red and yellow color, appear consistently at the desired junction regions. For example, the positions

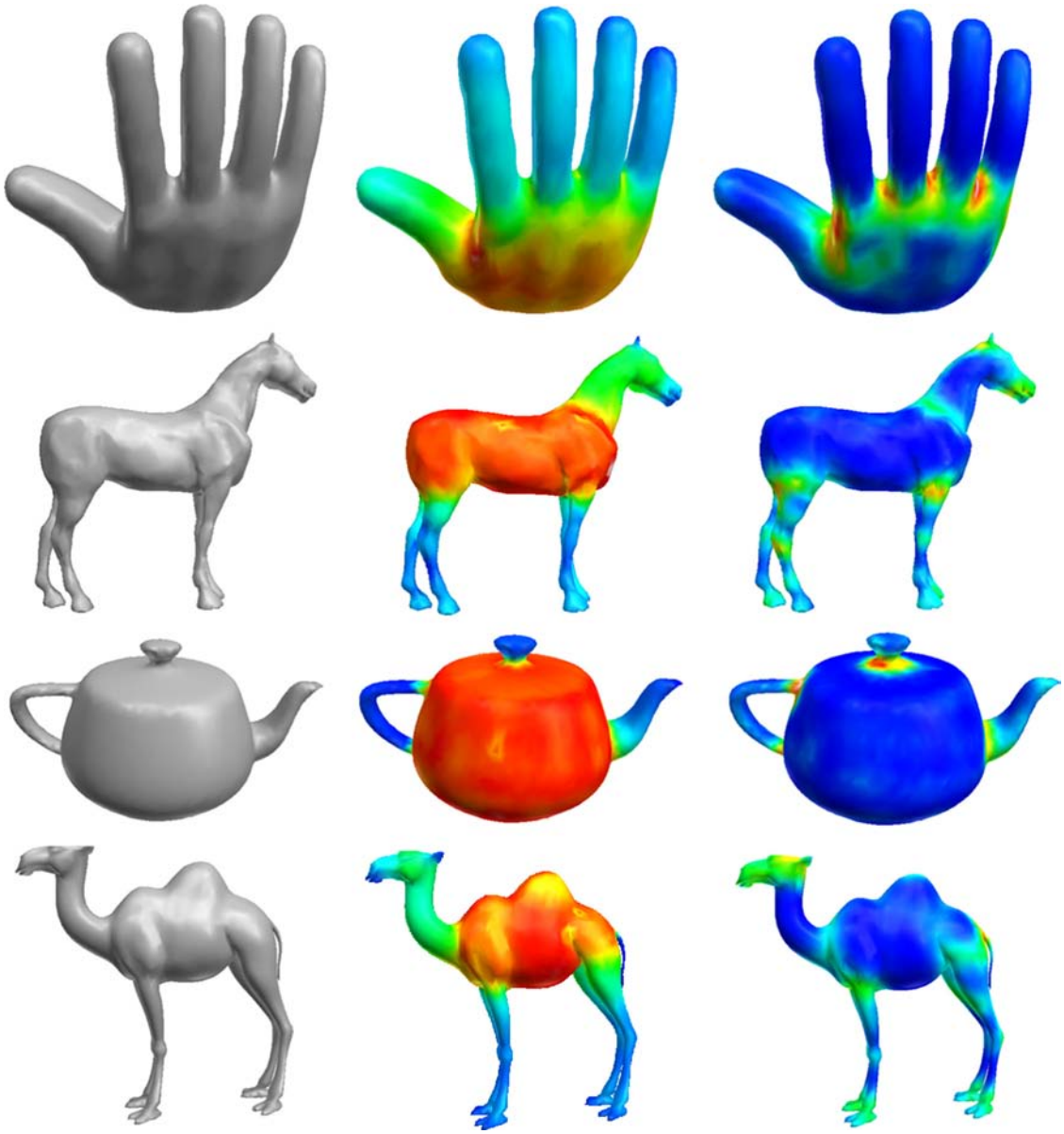

Fig. 5. Visualizing the SDF value and our junction-aware shape descriptor. Rows are hand, horse, teapot and camel. The leftmost column shows the original models, the middle one visualizes the SDF value, and the rightmost one displays our junction-aware shape descriptor. The high values are displayed by warmer colors (red and yellow), while the low values are displayed by cooler colors (green and blue). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

of fingers connected with the palm are highlighted, and the joints of handle and the body of a teapot are clearly different. This result shows that the proposed descriptor can make these junction regions distinct from other places.

The robustness of our method can be recognized by its insensitivity to the surface noise which may exist in many scanned models. To test the capability of our method to handle the noise, we add the uniformly distributed random noise along the vertex normals with increasing variances of the diagonal of the bounding box. Fig. 6 shows the result of a vase model, where a series of noisy models are produced by randomly adding Gaussian noise to each vertex along the positive normal directions with increasing variances. The result shows that the values of our junction-aware shape descriptor in different level noise are almost consistent with each other.

4.3. Comparison with several relevant methods

We also compare our work with several methods of surface-based measurements including mean curvature and mesh saliency [16]. Fig. 7 shows the result of various examples using mean curvature, mesh saliency and our junction-aware shape descriptor. The result suggests that mean curvature only reflects the convexities or concavities of local changes on boundary surface, and it is very sensitive to local perturbation

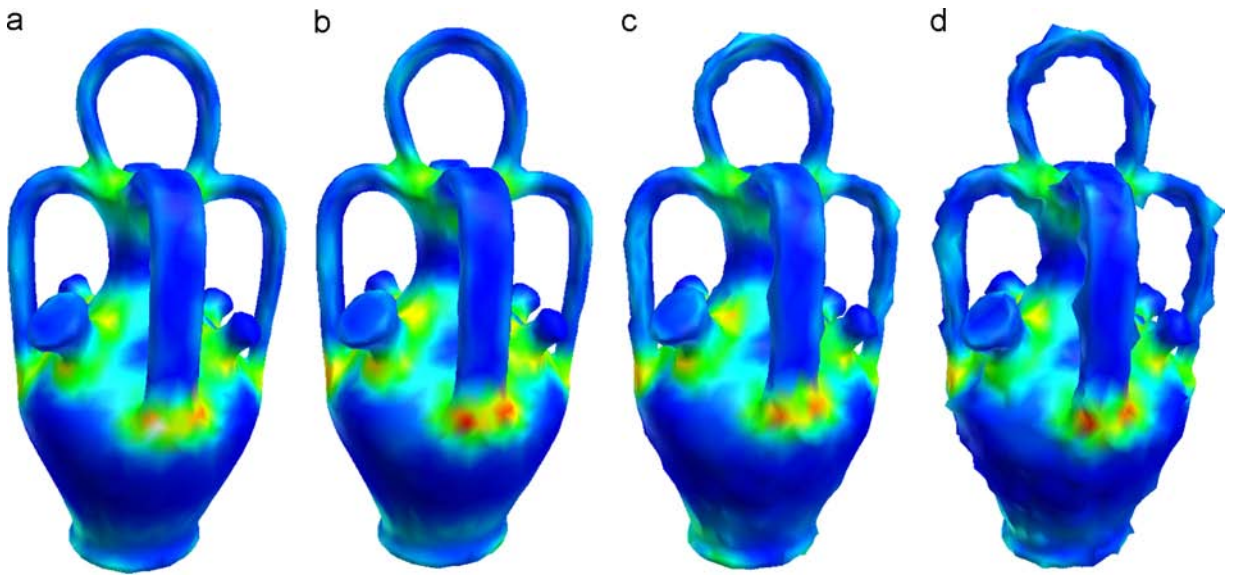

Fig. 6. Testing the robustness of our junction-aware shape descriptor for the same model but with different level noise. (a) Our descriptor for the original vase model without adding any noise is shown. (b)–(d) Our descriptor for the noisy models added the noise along the vertex normals with 1%, 5% and 10% variances, respectively, is shown.

and noise on surface. Mesh saliency [16,20] is a measurement of regional importance for 3D meshes using a center-surround operator on Gaussian-weighted mean curvatures. It is able to identify the regions that are different from their surrounding context. Although mesh saliency measurement is superior to mesh curvature, it does not capture the volumetric context and structure changes inside 3D shapes. In addition, mesh saliency has the same drawback with mean curvature during identifying the junction features, i.e. that the convex and concave features on local surfaces do not completely reflect the characteristics of junctions. For example, the nose of deer and the tips of octopus's tentacles are convex portions (see the middle column of Fig. 7) which are highlighted by both mean curvature and mesh saliency, but they are not junctions. In contrast, our junction-aware descriptor ignores such convex features and distinguishes the junction features well.

4.4. Applications

The junction-aware shape function presented in this paper is a new powerful geometric measurement, which encodes junction information on the shape boundary surface through explicitly considering the volumetric information inside 3D shapes. It offers a new perspective for understanding 3D articulated shapes, which can assist many other geometry processing applications. Some potential applications in computer vision and computer graphics may benefit from our descriptor. This section explores two potential applications of our descriptor in junction region extraction and handle loop approximation.

4.4.1. Initial junction region extraction

A direct application of our shape descriptor is for initial junction region extraction, which can assist on locating the targets to be deformed during articulated shape deformation. This can be simply done by using a predefined threshold λ to divide the descriptor values of mesh vertices into two types of regions, i.e. junction regions and rigid parts. Here we mark these vertices with their descriptor values larger than the threshold λ as the candidate vertices of junction regions, and the remaining vertices are marked as rigid parts. Fig. 8 shows an example of this application for several models, where the threshold λ is typically selected as the median of descriptor values of all vertices on the mesh.

However, when using such a predefined threshold for classifying the descriptor values of vertices into junction portions and rigid parts, the result may be unsatisfactory somewhere. In practice, because of the influence of local noise and perturbation, the distribution of computed shape descriptors may not always be smooth and continuous on boundary surfaces. This results in that some mesh vertices located in a presumptive junction region are not included in this junction, or that some distant vertices with high descriptor values are misclassified as the junction portion, leading to a discontinue junction region. As an example, some vertices on the ant's tail in Fig. 11(c) are wrongly identified as junction when using a threshold of descriptor median (see the colored regions on the ant's tail). This also happens in the bottom of the palm in Fig. 11(a).

Therefore, a more desired junction refinement needs further processing after extracting the initial junction region. In addition, a clear cut of curve along the refined junction region may provide many supports to improve the performance of articulated shape segmentation, so how to compute the cut along the junction region is also an interesting problem. We will introduce an algorithm to solve the above two issues in the following section.

4.4.2. Junction refinement and handle loop approximation

This section includes two purposes, i.e. how to obtain a more desired junction region and how to compute a cut of curve along the junction region. To achieve them, we

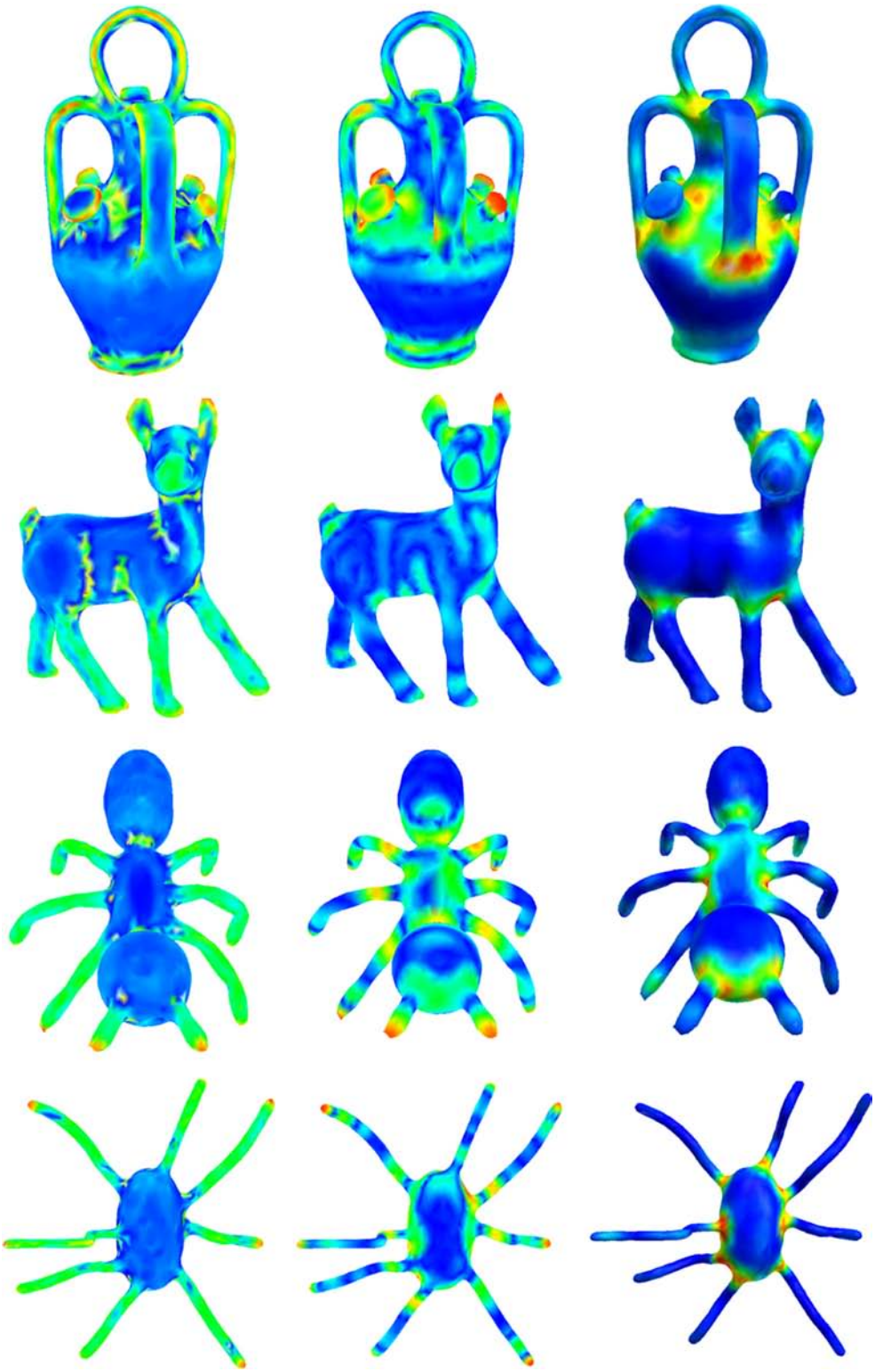

Fig. 7. Comparing our junction-aware shape descriptor with mean curvature and mesh saliency. Rows are vase, deer, ant and octopus. The leftmost column shows the values of mean curvature, the middle one visualizes the values of mesh saliency, and the rightmost one displays our junction-aware shape descriptor.

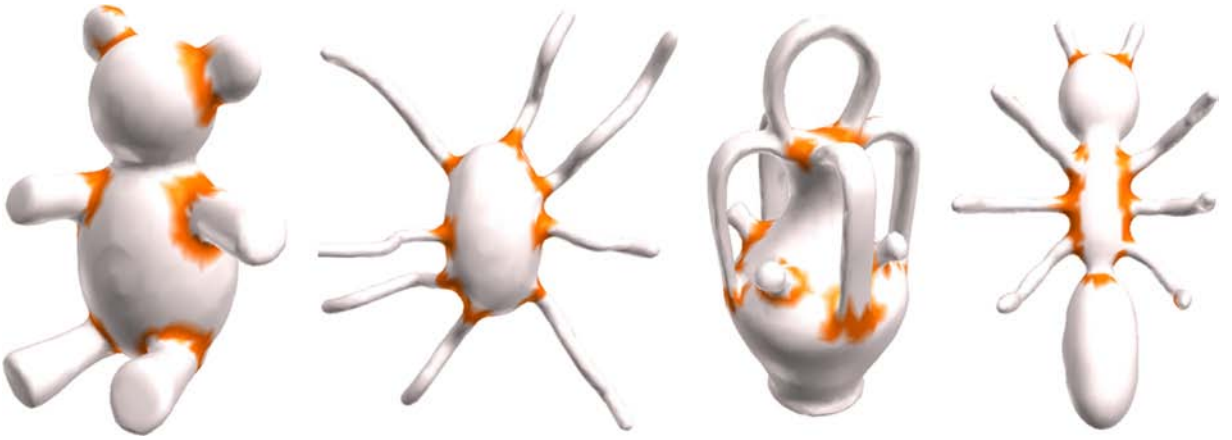

Fig. 8. Application to initial junction region extraction by using a predefined threshold to divide the descriptor values into two regions, where the colored regions are identified as the initial junction regions and the remaining ones denote the rigid parts. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

introduce a cluster-based algorithm for refining the initial junctions and approximating a *handle loop* as the cut along each junction. Our algorithm consists of three steps as follows.

(1) Firstly, we group all the vertices on initial junction regions into different clusters, one of which denotes a junction region to be further refined.
(2) Then, we refine the vertices on each clustering junction through removing some redundant vertices and adding some missing vertices, which leads to a desired junction region.
(3) Finally, we compute a handle loop along each junction region as the cut of shape segmentation.

Fig. 9 illustrates the procedure of our algorithm. The following gives the details of each step of the algorithm.

In the first step, all vertices of initial junction regions are first extracted using a predefined threshold, as shown in Section 4.4.1. Then we group these extracted vertices into different clusters, one of which denotes a junction region to be further refined. Here the clustering technique we use is the K-means++ algorithm [25] that is an extension version of K-means algorithm. Compared with the traditional K-means algorithm that heavily depends on the initial seeds, the K-means++ algorithm overcomes this disadvantage by initializing seed points iteratively. In Fig. 9(b), our method obtains three clusters on the horse's leg.

In the second step, we project all vertices of boundary surface in an inward normal direction to a distance which is shape-radius value [11]. This creates a set of reference points inside the shape that lies near its medial axis, as shown in Fig. 10. For each clustering junction, we find its *junction center*, which is one reference point with the maximum density computed by connecting the vertices of the cluster. Next, the boundary vertices, whose corresponding reference points are close to the junction center, are collected as the new vertices and added into the refined junction region. Meanwhile, some vertices of initial clusters, whose corresponding reference points are far from the junction center, are removed from the refined junction region. In Fig. 9(c), the colored vertices denote the refined junction regions. In contrast with Fig. 9(b), some new vertices are added and some old ones are removed at the refined junction regions on the horse's leg. We can find that the refined junction regions are smoother and more continuous than the old ones.

Alternatively, the last step further computes the *handle loop* [26,27] which is a critical geometric feature. The identification of handle loops can benefit a broad range of applications such as shape segmentation, topology simplification, surface parameterization and shape recognition [26,27]. Here we first use robust principal component analysis [28] for these vertices on each refined junction region to compute a plane. Then the intersection curve between the plane and the junction region forms the handle loop. In Fig. 9(d), the green loops denote the handle loops. In Fig. 11, the handle loops are displayed as the blue curves surround junction regions of each model.

4.4.3. Method evaluation

Besides visual inspection of the results of our method as shown in Fig. 11, we also present a brief statistical evaluation for our descriptor's performance on some articulated models selected from PSB [24,29]. PSB is built for testing 3D mesh segmentation algorithms, which covers a broad set of object categories (e.g. human, animals, furniture, tools, etc.). For our application purpose, we only choose some representative models of object categories, which most resemble articulated models. In addition, PSB provides segment boundaries (cuts) made manually by different people for each model, which are treated as the cuts of the "ground truth" segmentation. Also, we have discussed about the relationship of part boundary and junction in Section 2. Therefore, we set the cuts of meaningful parts of each selected model as the ground truth of our method evaluation, in which some non-joint cuts (e.g. nose contour and hairline on the face) in PSB are discarded since they are not relative to junctions.

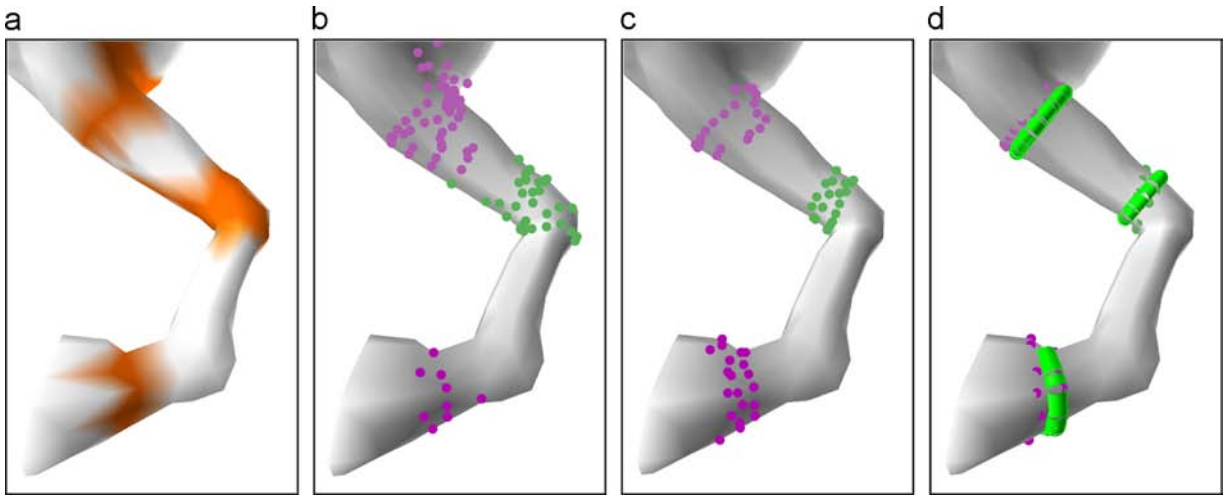

Fig. 9. Application to handle loop approximation for obtaining a desired cut along each junction portion. (a) The initial junction portions extracted with a predefined threshold of descriptor median. (b) We first group all the candidate vertices on initial junction portions into different clusters (highlighted by different colors). (c) Then we refine the vertices on each junction portions. (d) The final handle loops computed. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

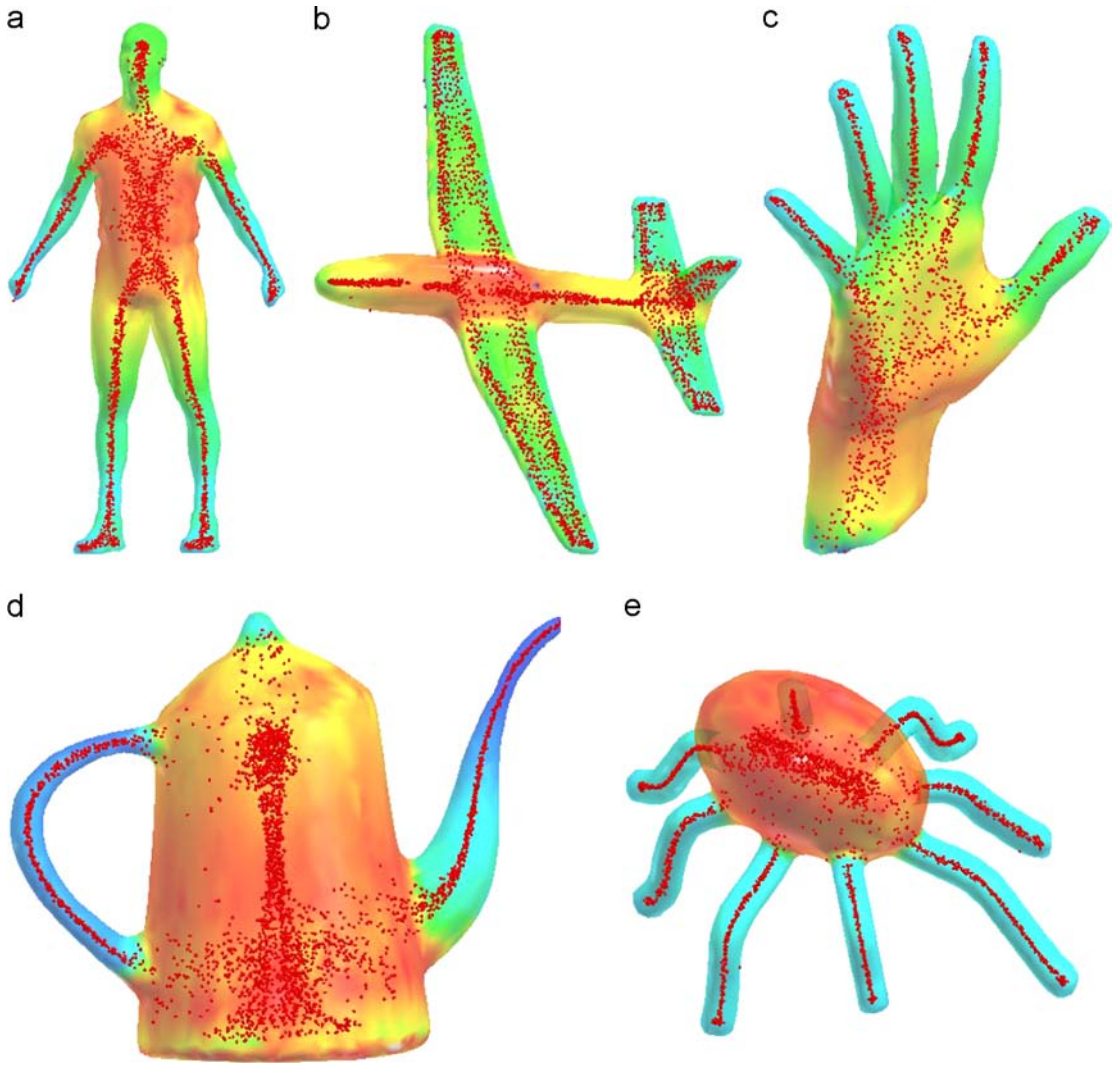

Fig. 10. Reference points (red dots) are computed by projecting all vertices of boundary surface inside the shape. Here the boundary surfaces are colored by their shape-radius values. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

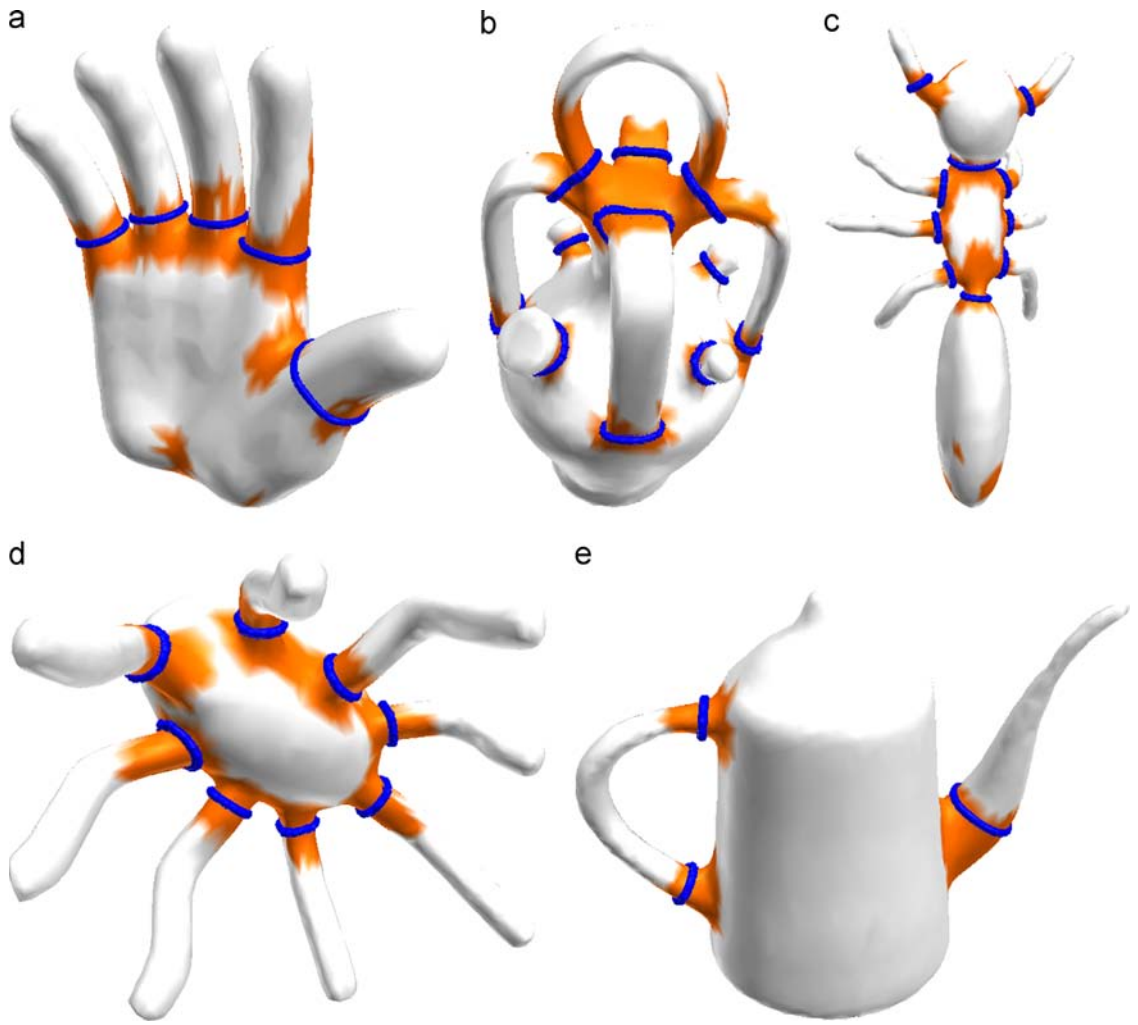

Fig. 11. Approximating the handle loops (blue curves). (a) Hand. (b) Vase. (c) Ant. (d) Octopus. (e) Teapot. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Table 2
Evaluation of our method on some models selected from PSB.

Category	# Models[a]	# Cuts[b]	# Loops[c]	# Matched loops[d]	Matching rate[e](%)
Teddy	13	91	95	89	97.8
Ant	20	210	197	192	91.4
Octopus	20	155	166	147	94.8
Vase	17	107	93	90	84.1

[a] "# Models" is the number of models in each object category.
[b] "# Cuts" is the number of cuts in the ground truth of manual segmentation [29].
[c] "# Loops" is the number of handle loops approximated by our method.
[d] "# Matched loops" is the number of handle loops matched with the cuts in ground truth.
[e] "Matching rate" is the ratio of the number of matched loops to the number of cuts in ground truth.

In our test, the database of junction evaluation contains 70 models from four categories (i.e. Teddy, Ant, Octopus and Vase) in PSB, the evaluation process examines how many of our approximated handle loops match with the ground truth cuts of each model. Table 2 shows the result of method evaluation, where "# Models" is the number of models in each object category, "# Cuts" is the number of cuts in the ground truth of manual segmentation, "# Loops" is the number of handle loops approximated by our method and "# Matched loops" is the number of handle loops matched with the cuts in the ground truth. The evaluation result is typically indicated by "Matching rate", which is the ratio of the number of matched loops to the number of cuts in ground truth. The matching rate is generally very high (the average is 92%), with a maximum of 97.8% in the "Teddy" category and a minimum of 84.1% in the "Vase" category. This suggests that our descriptor can locate most of the junctions in articulated models of PSB.

5. Conclusion

This paper considers the problem of junction description and detection for 3D articulated shapes, which only

resorts to geometric representation of the given individual shape without any other reference model. The main challenge lies in how to encode junction information on the boundary surface of the shape. By revealing the relationship of junction locations and shape-radius, we presented a new method for computing the junction-aware shape descriptor, which measures the local shape-radius variation as the Gaussian-weighted average of shape-radius in the neighborhood of each point on the surface. Being different with local surface-based measurement, the new descriptor takes account of the volumetric information within 3D shapes to capture the junction features. The effectiveness of our method is testified by a number of examples selected from several well-known 3D articulated shape databases. Furthermore, we explored several potential applications including junction region extraction and handle loop approximation, which could benefit from our descriptor. In addition, method evaluation is also introduced to show our descriptor's performance on some articulated models selected from PSB.

The type of junctions studied in this paper is restricted to non-rigid shape articulations, and the junction-aware feature we focused on is the shape-radius (local volume) variation. For this reason, this method does not involve an explicit structure of a 3D shape, and its effectiveness may be ineffective when applied on some artificial shapes with consistent shape-radius. For instance, as for a bending cylinder-like shape with nearly consistent shape-radius, our method will obtain almost identical values of descriptors, leading to indistinctive results. One possible way to overcome this drawback is to combine more volumetric information besides considering the shape-radius variation. On the other hand, the ray-shooting technique relies heavily on the determination of normal direction of a vertex, which has significant influence on the final result, so a more robust method for normal direction generation (or the normal-free method) is needed to cope with mesh surface noises such as evident fluctuations and holes.

Acknowledgements

This research is supported by the National Science Foundation of China (61272229, 61003095), the National Technological Support Program for the 12th-Five-Year Plan of China (2012BAJ03B07). The second author is also supported by Chinese 863 Program (2012AA040902) and Chinese 973 Program (2010CB328001).

References

[1] A. Bronstein, M. Bronstein, R. Kimmel, Numerical Geometry of Non-Rigid Shapes, Springer, Berlin, Germany, 2007.
[2] A. Bronstein, M. Bronstein, A. Bruckstein, R. Kimmel, Analysis of two-dimensional non-rigid shapes, Int. J. Comput. Vis. 78 (1) (2008) 67–88.
[3] H. Ling, D. Jacobs, Shape classification using the inner-distance, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 286–299.
[4] Y.-S. Liu, K. Ramani, M. Liu, Computing the inner distances of volumetric models for articulated shape description with a visibility graph, IEEE Trans. Pattern Anal. Mach. Intell. 23 (12) (2011) 2538–2544.
[5] C. Wang, Y.-S. Liu, M. Liu, J.-H. Yong, J.-C. Paul, Robust shape normalization of 3D articulated volumetric models, Comput.-Aided Des. 44 (12) (2012) 1253–1268.
[6] M. Shatsky, R. Nussinov, H.J. Wolfson, Flexible protein alignment and hinge detection, Proteins 48 (2) (2002) 242–256.
[7] R. Liu, H. Zhang, A. Shamir, D. Cohen-Or, A part-aware surface metric for shape analysis, Comput. Graphics Forum 28 (2) (2009) 397–406.
[8] D. Reniers, A. Telea, Part-type segmentation of articulated voxel-shapes using the junction rule, Comput. Graphics Forum 27 (7) (2008) 1845–1852.
[9] R. Su, C. Sun, T.D. Pham, Junction detection for linear structures based on Hessian, correlation and shape information, Pattern Recognit. 45 (10) (2012) 3695–3706.
[10] R. Gal, A. Shamir, D. Cohen-Or, Pose-oblivious shape signature, IEEE Trans. Vis. Comput. Graphics 13 (2) (2007) 261–271.
[11] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonization using the shape diameter function, Vis. Comput. 24 (4) (2008) 249–259.
[12] U. Emekil, D. Schneidman-Duhovny, H. Wolfson, R. Nussinov, T. Haliloglu, Hingeprot: automated prediction of hinges in protein structures, Proteins 70 (4) (2008) 1219–1227.
[13] O.K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction, ACM Trans. Graphics 27 (3) (2008). (Article 44).
[14] T. Ju, M. Baker, W. Chiu, Computing a family of skeletons of volumetric models for shape description, Comput.-Aided Des. 39 (5) (2007) 352–360.
[15] L. Shapira, S. Shalom, A. Shamir, R.H. Zhang, D. Cohen-Or, Contextual part analogies in 3D objects, Int. J. Comput. Vis. 89 (2–3) (2010) 309–326.
[16] C.-H. Lee, A. Varshney, D. Jacobs, Mesh saliency, ACM Trans. Graphics 24 (3) (2005) 659–666.
[17] O.K.-C. Au, Y. Zheng, M. Chen, P. Xu, C.-L. Tai, Mesh segmentation with concavity-aware fields, IEEE Trans. Vis. Comput. Graphics 18 (7) (2012) 1125–1134.
[18] P. Shilane, T. Funkhouser, Distinctive regions of 3D surfaces, ACM Trans. Graphics 26 (2) (2007). (Article 7).
[19] T. Funkhouser, P. Shilane, Partial matching of 3D shapes with priority-driven search, in: Symposium on Geometry Processing (SGP'06), 2006, pp. 131–142.
[20] Y.-S. Liu, M. Liu, D. Kihara, K. Ramani, Salient critical points for meshes, in: ACM Symposium on Solid and Physical Modeling (SPM'07), 2007, pp. 277–282.
[21] Y.-S. Liu, Y. Fang, K. Ramani, IDSS: deformation invariant signatures for molecular shape comparison, BMC Bioinf. 10 (157) (2009) 1–14.
[22] Y.-S. Liu, M. Wang, J.-C. Paul, K. Ramani, 3DMolNavi: a web-based retrieval and navigation tool for flexible molecular shape comparison, BMC Bioinf. 13 (95) (2012) 1–7.
[23] Y.-S. Liu, Q. Li, G.-Q. Zheng, K. Ramani, W. Benjamin, Using diffusion distances for flexible molecular shape comparison, BMC Bioinf. 11 (480) (2010) 1–15.
[24] X. Chen, A. Golovinskiy, T. Funkhouser, A benchmark for 3D mesh segmentation, ACM Trans. Graphics 28 (3) (2009). (Article 73).
[25] K–means++ Clustering, Available from: http://rosettacode.org/wiki/K-means++_clustering.
[26] T.K. Dey, K. Li, J. Sun, D. Cohen-Steiner, Computing geometry-aware handle and tunnel loops in 3D models, ACM Trans. Graphics 27 (3) (2008). (Article 45).
[27] T.K. Dey, F. Fan, Y. Wang, An efficient computation of handle and tunnel loops via reeb graphs, ACM Trans. Graphics 32 (4) (2013). (Article 32).
[28] Y.-S. Liu, K. Ramani, Robust principal axes determination for point-based shapes using least median of squares, Comput.-Aided Des. 41 (4) (2009) 293–305.
[29] A Benchmark for 3D Mesh Segmentation, Available from: ⟨http://segeval.cs.princeton.edu/⟩.