

## Technical Section

Feature preserving noise removal for binary voxel volumes using 3D surface skeletons<sup>☆</sup>Herman R. Schubert<sup>a</sup>, Andrei C. Jalba<sup>b</sup>, Alexandru C. Telea<sup>c,\*</sup><sup>a</sup> Bernoulli Institute, University of Groningen, Nijenborgh 9, Groningen 9747AG, the Netherlands<sup>b</sup> Department of Mathematics and Computer Science, Eindhoven University of Technology, Den Dolech 2, Eindhoven 5600MB, the Netherlands<sup>c</sup> Department of Information and Computing Sciences Utrecht University, Princetonplein 5, CCUtrecht 3584, the Netherlands

## ARTICLE INFO

## Article history:

Received 15 July 2019

Revised 18 December 2019

Accepted 20 December 2019

Available online 22 January 2020

## Keywords:

Skeletonization

3D shape smoothing

Shape processing

Denoising

## ABSTRACT

Skeletons are well-known descriptors that capture the geometry and topology of 2D and 3D shapes. We leverage these properties by using surface skeletons to remove noise from 3D shapes. For this, we extend an existing method that removes noise, but keeps important (salient) corners for 2D shapes. Our method detects and removes large-scale, complex, and dense multiscale noise patterns that contaminate virtually the entire surface of a given 3D shape, while recovering its main (salient) edges and corners. Our method can treat any (voxelized) 3D shapes and surface-noise types, is computationally scalable, and has one easy-to-set parameter. We demonstrate the added-value of our approach by comparing our results with several known 3D shape denoising methods.

© 2020 Elsevier Ltd. All rights reserved.

## 1. Introduction

High-resolution 3D models are widely used in a variety of applications such as prototyping, computer-aided industrial designs, games, and virtual reality systems. Such models inevitably have measurement noises from various sources, such as scanning, discretization, or quantization [1,2]. Similarly, 3D shapes extracted from volume data (e.g. MRI or CT scans) often contain significant noise, be it topological [3] or geometric [4], that needs to be removed before further usage.

Denoising a 3D model while preserving its salient geometric details (features) – also called regularization – is hard. Side effects can occur, such as shape distortion and feature blurring, which reduces the quality of the model. Additive noises from various sources, present at different scales, complicate the situation, as some noise may be wrongly considered a feature during regularization. An effective regularization method should remove noise, maintain features, and avoid side-effects.

Many feature-preserving denoising methods have been proposed [1,5–12]. Such methods work well for small- and single-scale noise. They usually separate noise (to be removed) from features (to be kept) *locally*. More advanced methods can remove noise occurring at different spatial scales (multiscale noise). While

multiscale denoising methods exist, as discussed in Section 2.1, they are more complex and delicate to set up. A second issue is that most feature-preserving denoising methods work on mesh or point-cloud representations. Far fewer methods exist for *binary voxel volumes*.

*Surface skeletons* capture the topology and geometry of shapes at different scales [13]. They are effective in many contexts – shape segmentation [14], registration [15], retrieval [16], and animation [17]. They have also been used for feature-preserving denoising of 2D binary shapes [18,19]. Although surface skeletons of 3D binary voxel shapes can be easily and efficiently computed [20,21], they have not been used for feature-preserving denoising of voxel shapes.

Given (1) the scarcity of feature-preserving denoising methods for voxel shapes, and (2) the demonstrated effectiveness of 3D surface skeletons to represent shapes in a multiscale way, we focus on how we can address task (1) with techniques from (2) for voxel shapes. To this end, we study the skeleton-based denoising of 2D binary shapes proposed in [19], identify its limitations when applied to 3D binary shapes, and show how to overcome these. We demonstrate our proposal on real-world binary 3D shapes corrupted by large amounts of multiscale, dense, and high-amplitude surface noise. Results show that our method can recover very well the underlying features (corners and edges) of the original 3D binary shapes. Our method can be applied to any binary 3D shape corrupted by surface noise, is computationally scalable, and has only one simple-to-set parameter. Our method is useful in cases where one wants to fully work in a voxel setting when

<sup>☆</sup> This article was recommended for publication by Dr D. Bommes.

\* Corresponding author.

E-mail addresses: [h.robert.schubert@gmail.com](mailto:h.robert.schubert@gmail.com) (H.R. Schubert), [a.c.jalba@tue.nl](mailto:a.c.jalba@tue.nl) (A.C. Jalba), [a.c.telea@uu.nl](mailto:a.c.telea@uu.nl) (A.C. Telea).

representing and denoising shapes, rather than working with a boundary representation (meshes or point clouds). Finally, we show that 3D *surface skeletons* are an effective tool for feature-preserving shape denoising – a task for which these skeletons have not been used so far. Summarizing, our contributions are as follows:

1. We show (to our knowledge, for the first time) how *surface skeletons* can do feature-preserving shape denoising;
2. We treat feature-preserving denoising of shapes entirely in a *voxel* setting;
3. We show how we can effectively handle the case of *multiscale* noise (of widely varying frequency and high spatial amplitude).

This paper is structured as follows. [Section 2](#) overviews related work in shape denoising and skeletonization. [Section 3](#) details our method. [Section 4](#) gives implementation details. [Section 5](#) shows results on real-world shapes and compares them with existing feature-preserving denoising methods. [Section 6](#) discusses our method. [Section 7](#) concludes the paper.

## 2. Background

### 2.1. Feature-preserving shape denoising

Many methods for feature-preserving denoising (fairing) of 3D shapes exist. Early methods use anisotropic geometric diffusion [22–25], building on earlier image-processing methods [26] to 3D shapes. Many such methods use a saliency map of the shape surface that assigns high values to important features such as strong edges. The saliency map weighs a shape deformation process to remove noise but keep features. For example, Lange et al. [12] use directional and principal curvatures as well as the Weingarten map for the saliency map, and use anisotropic geometric mean-curvature flow for their optimization. Other saliency-based mapping methods exist [9,27]. Diffusion-based methods [25,28–30] preserve salient geometric features of the denoised surface and can be computed efficiently [31]. However, such methods use first, second, or even fourth-order derivatives [32] (moments, curvature, and its second-order derivatives), which are local and can become unstable when a large amount of noise is present.

Other denoising methods use robust statistics or bilateral filtering. They rely on a similarity measure that changes how the optimization process depends on the value of the points, thus being more robust to outliers. Hence, feature preserving smoothing can be seen as estimating the surface in presence of outliers. An important early work here is [5]. They use bilateral filtering, where tangent planes based on filtered normals drive the similarity measure. Oztireli et al. [11] use robust statistics (M-estimators) to drive an implicit least-squares procedure that has good edge-preserving qualities. This method performs well with a low number of samples. Many other methods use robust statistics or bilateral filters [1,10,33–36]. Yet, such methods rely on *local* point neighborhoods, so they cannot differentiate globally important edges from local (possibly noisy) geometric details. Also, such methods need a given finite-kernel size of to estimate curvature. If the size is too small, one gets noisy curvature estimates; if the size is too large, curvature estimates are stable, but are localized in the filtered version of the input shape (rather than the shape itself), which results in poor localization of shape features. To address this issue, Hildebrandt and Polthier [37] formulate surface fairing as an optimization where a surface fairness measure is minimized subject to constraints, e.g., maximum distance to the input mesh.

Another class of fairing methods first smooth face normals and then reconstruct the denoised surface [38–40]. Yagou et al. use mean, median [39], and alpha-trimming [40] filters to smooth the normal field.

Tools from mathematical morphology can also perform surface fairing. For an implicit surface, level-set methods perform mean curvature-flow smoothing [41]. This PDE-based method is fast and has an automatic criterion to stop smoothing and keep the denoised surface close to the input. Additionally, an iterated median filter of the embedding (implicit) function is equivalent to the mean curvature flow of the level sets [42,43]. A morphological opening-closing filter smooths a (binary) signal similar to a median filter [44].

### 2.2. Denoising voxel vs polygon or point-cloud shapes

As [Sec. 2.1](#) shows, most feature-preserving denoising methods use a *boundary representation* (b-rep) of shape, typically a polygon mesh or point cloud. Only very few denoising methods treat *binary voxel* representations (v-reps) of 3D shapes. This is explainable since many applications use b-reps, which are more compact to store and can represent fine details more efficiently than v-reps. Also, b-reps offer a higher freedom for denoising, as (1) points can be placed anywhere in  $\mathbb{R}^3$ , and (2) points can be added and/or deleted to enforce local shape properties. In contrast, v-reps densely sample  $\mathbb{R}^3$  on a fixed-resolution grid, so have far less freedom to represent small-scale details (and thus also when denoising). This makes the creation of high-quality denoising methods for v-reps (our goal in this paper) more challenging. Yet, binary v-reps have several advantages: They have a simpler *implementation*; do not suffer from the problem of *missing or inconsistent* data, such as holes in a point cloud or meshes with inconsistently oriented, self-intersecting, or degenerated, triangles; and can trivially handle *outlier* samples, which appear as small-scale voxel groups disconnected from the main shape, by using largest connected-component filtering or morphological opening.

### 2.3. Skeletonization

**Notations** Let  $\Omega \subset \mathbb{R}^d$  be a compact shape, with boundary  $\partial\Omega$ , embedded in 2D ( $d = 2$ ) or 3D ( $d = 3$ ). Its distance transform  $DT_\Omega : \mathbb{R}^d \rightarrow \mathbb{R}^+$  is given by

$$DT_\Omega(\mathbf{x} \in \mathbb{R}^d) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|. \quad (1)$$

Using  $DT_\Omega$ , we can define the skeleton  $S_\Omega \subset \mathbb{R}^d$  as

$$S_\Omega = \{\mathbf{x} \in \Omega \mid \exists (\mathbf{y}_1, \mathbf{y}_2) \in \partial\Omega \times \partial\Omega, \mathbf{y}_1 \neq \mathbf{y}_2, \|\mathbf{y}_1 - \mathbf{x}\| = \|\mathbf{y}_2 - \mathbf{x}\| = DT_\Omega(\mathbf{x})\} \quad (2)$$

Simply put, [Eq. \(2\)](#) says that  $S_\Omega$  is the set of points inside the shape  $\Omega$  which are at a distance equal to their distance-transform from at least two different points  $\mathbf{y}_1$  and  $\mathbf{y}_2$  on the shape's boundary  $\partial\Omega$ . Even simpler put,  $S_\Omega$  is the locus of centers of maximally-inscribed balls in  $\Omega$  [45]. The points  $\mathbf{y}_i$  are called the *feature* points of skeleton point  $\mathbf{x}$  [46], and are the contact points with  $\partial\Omega$  of the maximally-inscribed ball in  $\Omega$  (of radius  $DT_\Omega(\mathbf{x})$ ) of center  $\mathbf{x}$ . The set  $FT_\Omega(\mathbf{x})$  of all feature points of a skeleton point  $\mathbf{x}$  is called the *feature transform* [47] of  $\mathbf{x}$ . The pair  $(S_\Omega, DT_\Omega)$ , called the *medial axis transform* (MAT) of  $\Omega$ , is a dual representation of the shape, i.e., allows one to exactly reconstruct  $\Omega$  as the union of balls centered at the skeletal points  $\mathbf{x} \in S_\Omega$  and having radii  $DT_\Omega(\mathbf{x})$ .

**Computing skeletons** Many methods exist for computing approximations of  $S_\Omega$ . Key to all such methods is a *regularization* process that removes from  $S_\Omega$  so-called spurious branches, caused by small-scale perturbations of the surface  $\partial\Omega$ , due to sampling and/or acquisition noise. Such methods define an importance  $\rho : S_\Omega \rightarrow \mathbb{R}^+$  and next define the regularized skeleton as  $\bar{S}_\Omega = \{\mathbf{x} \in S_\Omega \mid \rho(\mathbf{x}) \geq \tau\}$  for a user-selected regularization parameter  $\tau \geq 0$ . For 2D shapes  $\Omega \subset \mathbb{R}^2$ ,  $\rho$  is commonly set to the longest shortest-path distance  $\delta$  along  $\partial\Omega$  between all feature-point pairs, i.e.,

$\rho(\mathbf{x}) = \max_{(\mathbf{y}_1, \mathbf{y}_2) \in FT_{\Omega}(\mathbf{x})} \delta(\mathbf{y}_1, \mathbf{y}_2)$  [48–50]. This keeps in  $\bar{S}_{\Omega}$  only skeletal points corresponding, via the feature transform, to boundary details longer than  $\tau$  units.

3D skeletons computed via Eq. (2) or equivalents, also called medial surfaces, have a considerably more complex structure than their 2D counterparts, so regularization is imperiously needed to provide simple enough representations for practical applications. Medial surfaces can be computed for both boundary-representation (meshed) models and volumetric (voxel) models by Voronoi and bisector methods, shrinking ball methods, topological thinning, and distance-field methods, see recent surveys thereof [20,21]. Surface skeletons can be regularized by generalizing the distance-between-feature points metric for 2D skeletons to the shortest-path (geodesic) distance along the surface  $\partial\Omega$  between feature points [50–53]. A similar metric  $\rho$  can be computed by the advection of uniformly-spread mass from  $\partial\Omega$  to  $S_{\Omega}$  and then along the manifolds of  $S_{\Omega}$  towards its center [54]. Other regularization metrics include divergence and moments of the distance transform's gradient  $\nabla DT_{\Omega}$  [55–57] and discarding skeletal points whose feature points have a circumradius larger than a given threshold [58].

*Skeleton-based shape denoising* Regularized skeletons open new ways for shape denoising. If regularization removes the endpoints of skeletal terminal manifolds in  $S_{\Omega}$ , then the MAT ( $\bar{S}_{\Omega}, DT_{\Omega}$ ) of the regularized skeleton  $\bar{S}_{\Omega}$  allows reconstructing a *simplified* version  $\bar{\Omega}$  of  $\Omega$  where all surface zones corresponding to removed skeletal points are replaced by circle arcs (in 2D), respectively cylindrical and spherical caps (in 3D). Hence, regularizing the skeleton by removing points corresponding to noise-scale shape details directly eliminates such noise.

Not all regularization metrics perform equally well for the above task. *Local* metrics (divergence, moment, or circumradius) cannot distinguish between locally similar but globally different shape configurations and may disconnect the skeleton during denoising [55–58]. *Global* metrics like the geodesic distance or mass-advection do not have this problem as they monotonically increase from the skeleton boundary towards its center [50,52–54]. Yet, they have the separate problem that they also remove *salient* shape features, such as edges and/or corners, together with same-scale noise, much like classical isotropic Laplacian filtering [59].

*Salience* metrics alleviate this problem by adding information on important (salient) shape features into the regularization. For 2D shapes, Tek and Kimia iteratively remove skeleton branches using a saliency metric equal to the area-difference between the smoothed shape  $\bar{\Omega}$  and the original shape  $\Omega$  divided by the skeleton branch length [60]. Bai et al. [61] prune the skeleton by partitioning the boundary  $\partial\Omega$  into segments by discrete-curve evolution and removing branches corresponding to less important segments. Liu et al. [18,62] remove skeleton branches based on a saliency metric including the reconstruction contribution (area difference between  $\Omega$  and  $\bar{\Omega}$ ) and the length of the skeleton branch-part that is not contained in the maximal sphere centered at the adjacent branch point. Yet, such branch-pruning methods require a careful topological analysis of the skeleton (detecting branches and their junctions) which is very hard to generalize to the complex structure of 3D medial surfaces, so are not readily applicable to 3D shapes.

A different salience metric [19] uses the ratio

$$\sigma(\mathbf{x}) = \frac{\rho(\mathbf{x})}{DT_{\Omega}(\mathbf{x})} \quad (3)$$

where  $\rho$  is the shortest-path importance metric in [49]. Consider a noisy rectangle (Fig. 1a). For this shape,  $\sigma$  gradually decreases along skeleton branches caused by noise details on  $\partial\Omega$  but stays roughly constant along branches caused by salient (important) corners (Fig. 1d and inset). Hence, we can disconnect the noise branches from the *skeleton core* – that is, the skeleton excluding

branches created by small-scale, noise, perturbations on  $\partial\Omega$  [63] – by upper thresholding  $\sigma$  with a user-defined value  $\sigma_0$  (Fig. 1e). The low- $\sigma$  removed skeletal points are the so-called *skeleton ligatures* [13], along which  $\rho$  stays constant. Since  $\sigma$  stays high on branches caused by salient features (corners), these branches are not pruned. Next, one keeps the skeleton connected-component containing the largest  $\rho$  value after the pruning pass (Fig. 1f). From this, a denoised version  $\bar{\Omega}$  of the shape  $\Omega$ , with salient corners preserved, is computed as the union of balls of the core skeleton (Fig. 1f). However, the importance  $\rho$  is low towards the end of both noise and salient branches (Fig. 1b and inset). Hence, regularizing  $S_{\Omega}$  by upper thresholding  $\rho$  prunes both branch types, so reconstruction removes both noise and smooth corners (Fig. 1c).

### 3. Method

The skeleton-based method [19] described above effectively removes even large-amplitude and multiscale boundary noise from 2D shapes while keeping salient corners intact, is simple to implement, and computationally efficient (linear in the number of foreground pixels in  $\Omega$ ). Yet, generalizing this method to 3D shapes is not straightforward. We next identify three main problems for the 3D case (see also Fig. 2):

(A) *Noise near features* Consider the behavior of the saliency  $\sigma$  (Eq. (3)) along a noise-induced skeleton branch (Fig. 1d inset): The four noise bumps are successfully removed as they are far from the corner, so their ligature branches are long enough to allow  $DT_{\Omega}$  to increase sufficiently to make  $\sigma$  to drop below the user threshold  $\sigma_0$ . However, if such noise bumps were closer to the corner, their ligature branches would be too short to yield  $\sigma < \sigma_0$ . So, upper thresholding  $\sigma$  fails to remove noise close to salient corners. Decreasing  $\sigma_0$  does not solve this problem, as it also removes the tips of the skeleton branches caused by salient corners, which is unwanted. The same problem appears in 3D for noise close to shape edges or corners, which cannot be removed by thresholding  $\sigma$  (Fig. 2a, last column).

(B) *Noise crossing edges* Consider elongated noise that crosses (intersects) a salient edge of a 3D shape. Fig. 2b (column 1) shows such an example – a cube where a box-like bump was added across one edge. Fig. 2b (column 2) shows the skeleton  $S_{\Omega}$  colour-coded by the saliency  $\sigma$ . We see that the noise bump creates a skeletal sheet orthogonal to the cube's surface skeleton. Upper thresholding  $\sigma$  removes some, but not *all* of the voxels on this sheet (Fig. 2b, column 3). Unlike in the 2D case, thresholding does not *fully* disconnect the noise-sheet from the rump skeleton. Hence, we cannot remove such type of edge-crossing noise by the connected-components procedure outlined for the 2D case in Sec. 2.3.

(C) *Distance transform ripples* Fig. 2c shows a third and last problem of the method in [19]. The input cube shape was noised by adding a few simple Gaussian-like bumps (Fig. 2c, column 1). These create several skeletal sheets (Fig. 2c, column 2). Thresholding the saliency  $\sigma$  fully removes these sheets (Fig. 2c, column 3). However, close to the junction points of the removed sheets, the regularized skeleton exhibits several 'ripples', as its surface is bent so as to be centered in the input shape, following Eq. (2). Reconstructing the smoothed shape from this regularized skeleton next creates some subtle, but unwanted, undulations on the final surface (Fig. 2c, column 4).

We next propose several changes of 2D skeleton-based shape smoothing [19] that address the three above problems.

#### 3.1. Problem A: Removing noise close to shape features

As explained in Section 2.3, along a noise-induced skeleton branch, the importance  $\rho$  first increases, then plateaus over the



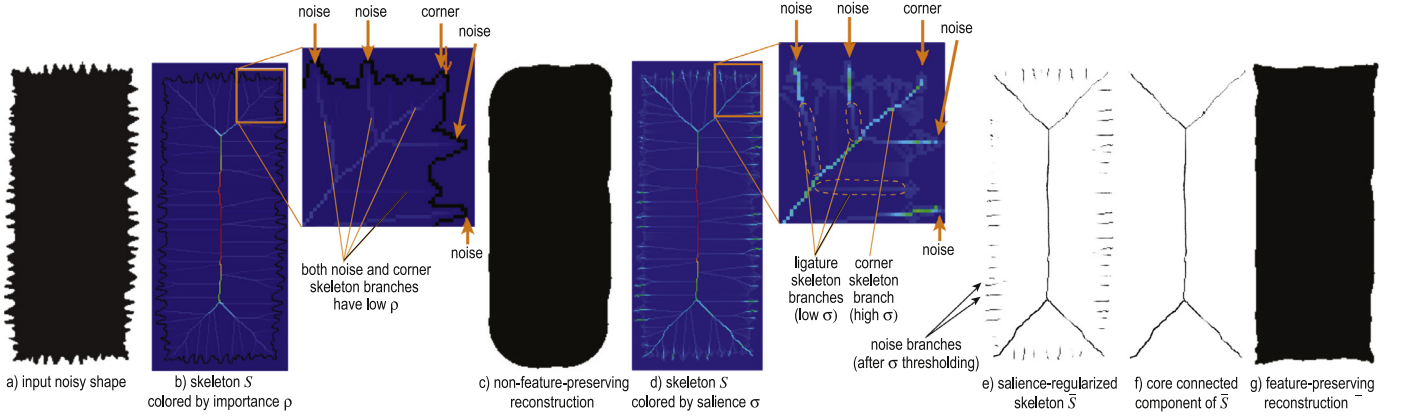


Fig. 1. Saliency skeleton smoothing method for 2D shapes [19].

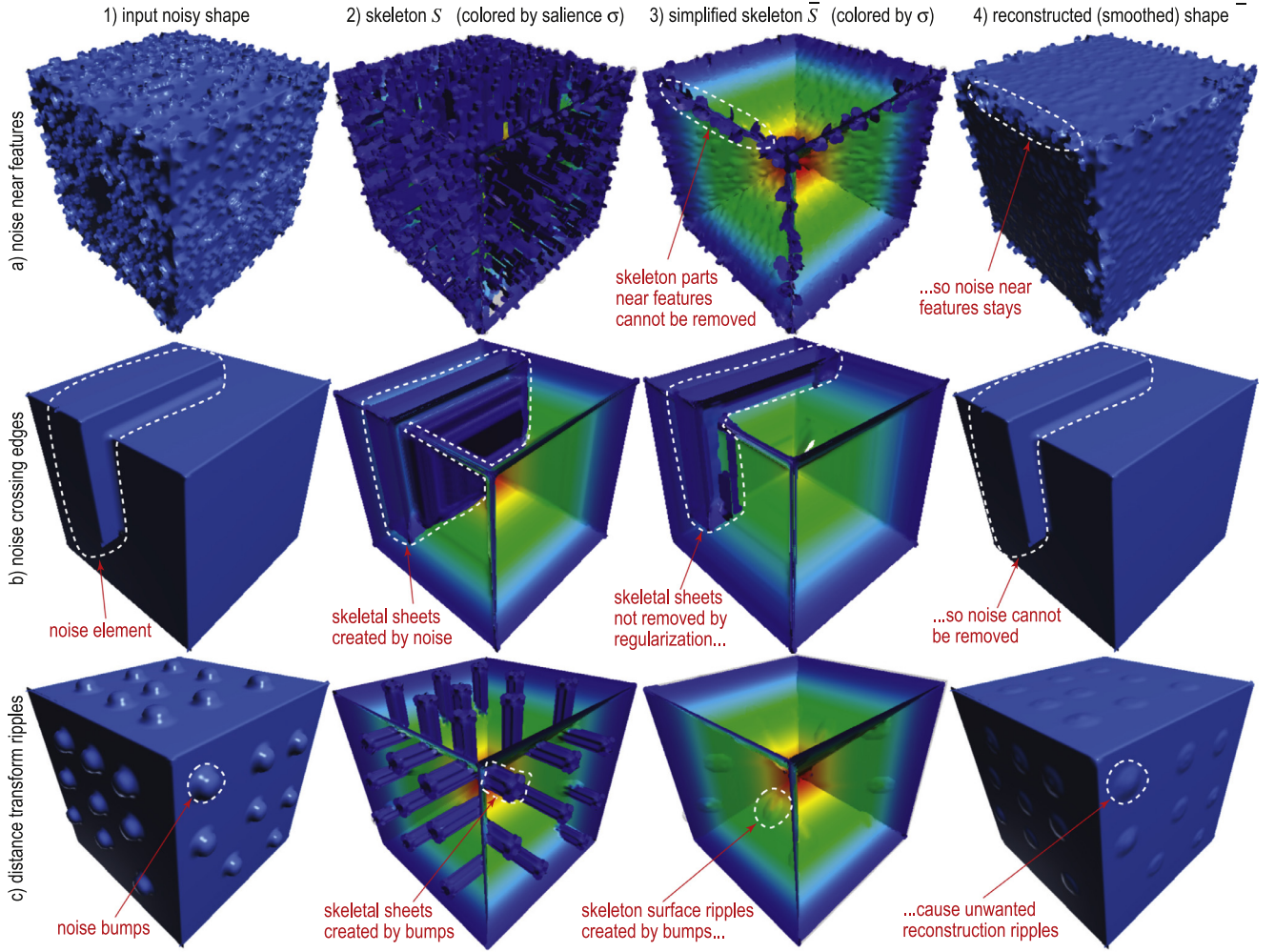


Fig. 2. Problems when applying the smoothing method [19] to 3D shapes. (a) Noise near corners or edges cannot be removed. (b) Noise crossing edges cannot be removed. (c) Distance transform ripples at the junctions of the removed skeletal sheets with the core skeleton create unwanted surface undulations.

range of ligature points. Hence, we propose to detect ligature points by using the directional derivative of  $\rho$  along a vector field  $\mathbf{v} : S_{\Omega} \rightarrow \mathbb{R}^3$  that is tangent to the skeleton and points towards the skeleton core, i.e., the quantity  $\nabla \rho \cdot \mathbf{v}$ . In ligature areas,  $\nabla \rho \cdot \mathbf{v}$ , approaches zero, as  $\rho$  is locally constant there. Indeed, by definition, ligature branches are sets of points that have the same feature points  $\mathbf{y}_i$  on  $\partial\Omega$  [13], hence the same shortest-paths between  $\mathbf{y}_i$ , hence the same  $\rho$  (shortest-path length). Separately, the distance transform always increases as we advance along a ligature branch

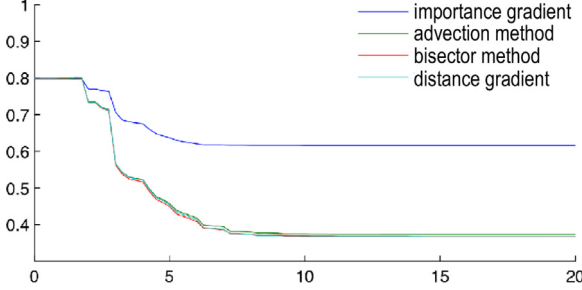
towards the skeleton core [13]. Hence, we propose to replace  $\sigma$  by

$$\bar{\sigma} = \frac{\nabla \rho \cdot \mathbf{v}}{\nabla DT_{\Omega} \cdot \mathbf{v}} \quad (4)$$

Just as  $\sigma$ ,  $\bar{\sigma}$  is low along ligatures. However,  $\bar{\sigma}$  depends only on the derivatives of  $\rho$  and  $DT_{\Omega}$ , and not their absolute values, so it is scale-invariant with respect to the noise size and the noise position vs salient shape features.

**Table 1**  
Errors of different vector fields used for the improved saliency metric  $\bar{\sigma}$  (Eq. (4)).

|         | Importance gradient             |                                      | Distance gradient               |                                      | Bisector method                 |                                      | Advection model                 |                                      |
|---------|---------------------------------|--------------------------------------|---------------------------------|--------------------------------------|---------------------------------|--------------------------------------|---------------------------------|--------------------------------------|
|         | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ |
| cube    | 0.2506                          | 0.2655                               | 0.1565                          | 0.1726                               | <b>0.1523</b>                   | <b>0.1686</b>                        | 0.1556                          | 0.1718                               |
| bunny   | 0.5094                          | 0.5621                               | 0.1840                          | 0.3893                               | <b>0.1715</b>                   | <b>0.3820</b>                        | 0.1689                          | 0.3835                               |
| fandisk | 0.5708                          | 0.6057                               | 0.2904                          | 0.3741                               | <b>0.2751</b>                   | 0.3684                               | 0.2768                          | <b>0.3666</b>                        |



**Fig. 3.** Total error  $\epsilon$  for different vector field models  $\mathbf{v}$  as function of the smoothing threshold  $\sigma_0$  for the *fandisk* model.

To compute  $\bar{\sigma}$ , we need to define the vector field  $\mathbf{v}$ . We next present four designs for  $\mathbf{v}$  which have different trade-offs between implementation simplicity and accuracy, as follows:

**Importance gradient:** As  $\rho$  is zero outside the skeleton by construction, and increases along branches as we approach the skeleton core [52,53],  $\mathbf{v} = \nabla \rho$  is a possible choice for  $\mathbf{v}$ ;

**Distance gradient:** As the distance transform increases as we advance away from  $\partial\Omega$  into the shape, we can set  $\mathbf{v} = \nabla DT_\Omega$ ;

**Bisector method:** For any skeletal point  $\mathbf{x} \in S_\Omega$ , its two feature points  $\mathbf{y}_1$  and  $\mathbf{y}_2$  are on different sides of the tangent plane to  $S_\Omega$  at  $\mathbf{x}$ . This tangent plane bisects the angle formed by the feature vectors  $\mathbf{y}_1 - \mathbf{x}$  and  $\mathbf{y}_2 - \mathbf{x}$  [64]. Hence, we can compute our vector field as  $\mathbf{v} = \mathbf{x} - (\mathbf{y}_1 + \mathbf{y}_2)/2$ .

**Advection method:** Several methods compute 3D skeletons by contracting the surface  $\partial\Omega$  in a vector field  $\mathbf{v}$  that simulates advection of uniformly-spread mass from  $\partial\Omega$  following a momentum conservation principle [54,65]. We use here the field  $\mathbf{v}$  proposed in [54], which has the desirable properties of being tangent to the skeleton surface and pointing towards its core.

**Best method choice:** We test which of the above four definitions of  $\mathbf{v}$  yields the best detector  $\bar{\sigma}$  as follows. We consider a noise-free shape  $\Omega$  and its un-regularized (full) skeleton  $S_\Omega$ . We next add noise to  $\Omega$ , yielding the shape  $\bar{\Omega}$ , and compute its skeleton  $\bar{S}_\Omega$ , regularized by  $\bar{\sigma}$ , for all vector fields  $\mathbf{v}$ , with gradients computed by central differences. Ideally,  $\bar{S}_\Omega$  should be very close to the ‘ground truth’  $S_\Omega$ , since then the shape reconstructed from it is very close to the original noise-free shape  $\Omega$ . We compare  $\bar{S}_\Omega$  and  $S_\Omega$  with the following two metrics

$$\nu(\bar{S}_\Omega, S_\Omega) = |\bar{S}_\Omega \setminus S_\Omega| / |\bar{S}_\Omega|, \quad (5)$$

$$\epsilon(\bar{S}_\Omega, S_\Omega) = 1 - |\bar{S}_\Omega \cap S_\Omega| / |\bar{S}_\Omega \cup S_\Omega| \quad (6)$$

that is, total error (false positives and false negatives), respectively false positive rate. Table 1 shows these errors for three shapes in Fig. 4 and the four considered vector fields  $\mathbf{v}$ . Bold figures indicate minimal errors per shape. We see that the bisector-method achieves the lowest errors except for one case, where the distance-based method is slightly better.

Fig. 3 shows the total error  $\epsilon$  (Eq. (6)) as function of the smoothing threshold  $\sigma_0$  for the *fandisk* model. The error monotonically decreases with  $\sigma_0$  for all four considered vector-field designs,

which implies detection robustness in terms of false positives. The feature, bisector, and advection models yield very similar errors, all lower than the importance gradient model. Hence, from an error perspective, the former three models are better. From an implementation perspective, the importance gradient, advection, and bisector methods are all trivial to implement, and equally fast. The advection method is considerably more complex to implement and over two orders of magnitude slower as it involves solving a system of partial differential equations on the voxel volume (for details, see [54,65]). Finally, the distance gradient involves differentiation (computing  $\nabla DT_\Omega$ ), which can be affected by numerical noise on low-resolution voxel models, whereas the bisector method does not. Hence, our final best-method choice is the bisector method.

Fig. 4 compares the use of the improved saliency  $\bar{\sigma}$  vs the original saliency  $\sigma$  for the three shapes in Table 1. We see that  $\bar{\sigma}$  removes more noise skeletal sheets close to skeleton boundaries than  $\sigma$  (compare markers in columns (c) vs (e)). Hence, the  $\bar{\sigma}$ -regularized skeleton yields better noise removal along the shape’s main edges (compare markers in columns (d) vs (f)).

### 3.2. Problem B: Removing edge-crossing noise

The improved saliency  $\bar{\sigma}$  removes noise close to the shape edges and most, but not all, small-scale noise located on such edges. Fig. 4 (insets in column (f)) shows some cases where noise on edges cannot be removed. This is due to the complex topology of 3D skeletal manifolds (Section 2.3): Even when we fully prune their ligature sheets, noise-induced sheets will cross salient-edge sheets, so we cannot disconnect the former from the latter. This was never a problem for 2D shapes, due to their far simpler-topology skeletons.

To solve this problem, we propose a new salience metric  $\sigma^*$  that combines the desirable properties of the importance  $\rho$  and saliency  $\bar{\sigma}$  (see next Fig. 5): We first compute the importance  $\rho$ , which is, as discussed earlier, low on both the noise sheets corresponding to the bump added atop the box, and on the ligatures linking these sheets to the core, high- $\rho$ , skeleton (Fig. 5b). Next, we compute  $\bar{\sigma}$  which is, as explained, low on ligatures but high on both the noise and core-skeleton sheets (Fig. 5c). Hence, thresholding  $\bar{\sigma}$  cannot disconnect the noise sheets from the core skeleton. We now introduce the computation of  $\sigma^*$ : From each skeletal point  $\mathbf{x} \in S_\Omega$ , we trace a streamline in the vector field  $\mathbf{v}$  (computed by the bisector method, see Section 3.1), as long as the encountered voxels  $\mathbf{y}$  have increasing importances  $\rho(\mathbf{y})$  and saliences  $\bar{\sigma}(\mathbf{y}) \geq \sigma_0$ . Fig. 5c shows two such streamlines with white arrows: If we start at point  $\mathbf{x}$  (step 1 in figure), the streamline advances as long as  $\rho$  increases and  $\bar{\sigma} \geq \sigma_0$  (step 2 in figure). We can see that  $\rho$  increases and  $\bar{\sigma}$  is indeed large along that path by looking at their values in Figs. 5b and c, respectively. When the streamline reaches the center  $\mathbf{x}'$  of the skeleton,  $\rho$  stops increasing since it is maximal here (red value in Fig. 5b), so the streamline stops (step 3 in Fig. 5d). Finally, we set  $\sigma^*(\mathbf{x})$  to the importance  $\rho$  of the last voxel (that is,  $\mathbf{x}'$ ) on the streamline (step 4 in figure). All points on core-skeleton sheets, like  $\mathbf{x}$ , will get a high  $\sigma^*$  value. In contrast, points on noise sheets (such as  $\mathbf{y}$  in Fig. 5e) get a low  $\sigma^*$  value, since streamlines starting at them enter, at point  $\mathbf{y}'$ , low- $\bar{\sigma}$



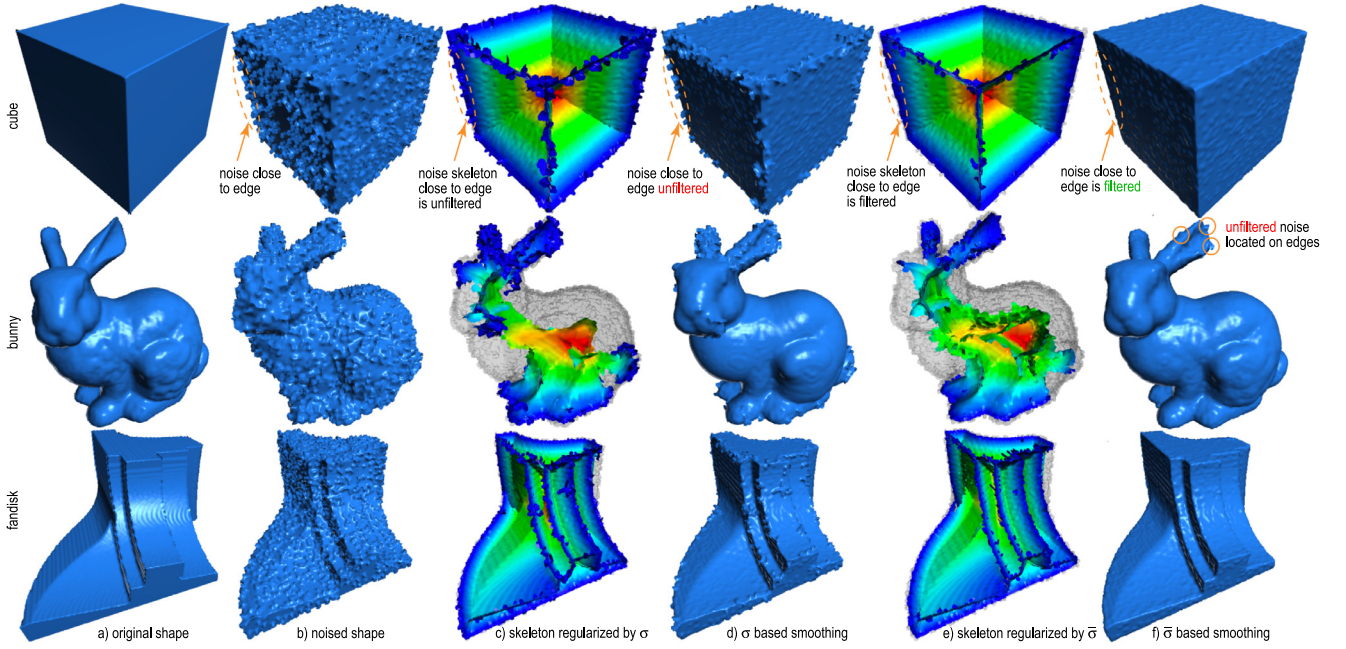


Fig. 4. Comparison of improved saliency metric  $\bar{\sigma}$  and original saliency  $\sigma$  for shape smoothing.

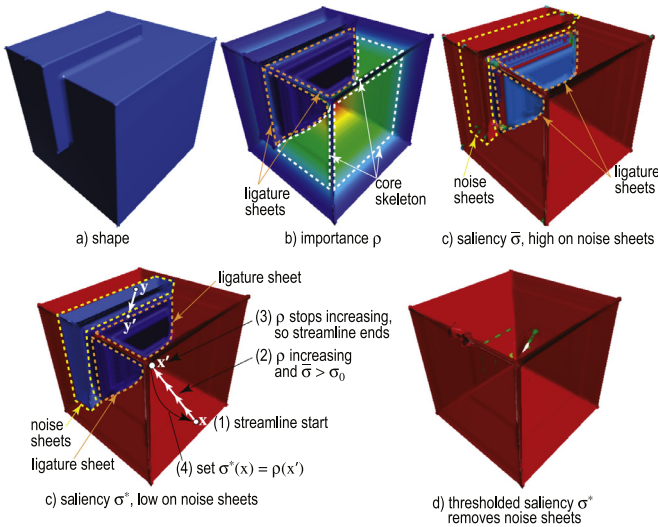


Fig. 5. Comparison of improved and global saliencies  $\bar{\sigma}$  and  $\sigma^*$ . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ligature sheets (blue in Fig. 5c). Summarizing, on the core skeleton,  $\sigma^*$  is high (like  $\bar{\sigma}$  but unlike  $\rho$ ); and on noise and ligature sheets,  $\sigma^*$  is low (like  $\rho$  but unlike  $\bar{\sigma}$ ). Hence, thresholding  $\sigma^*$  completely removes both the ligature and the noise sheets connected to it (Fig. 5f). In contrast, thresholding  $\bar{\sigma}$  does remove the ligatures but not the noise sheets, as shown earlier in Fig. 2(3c).

Computing  $\sigma^*$  by tracing streamlines from all skeletal points is very expensive –  $O(|S_\Omega|^2)$  complexity – as it visits the same points many times. We propose a faster method: First, we sort all voxels  $\mathbf{x} \in S_\Omega$  in decreasing  $\rho$  order, using radix sort. Next, we set  $\sigma^* = \rho$  for all ligature voxels (i.e. having  $\bar{\sigma} \leq \sigma_0$ ) and also for the voxel(s) having the maximal  $\rho$  value. From these ‘seeds’, we propagate  $\sigma^*$  to 27-neighbor skeleton voxels, using flood-fill, so that a voxel’s  $\sigma^*$  is always set to the largest of the assigned  $\sigma^*$  values to its neighbors. The entire process takes now  $O(|S_\Omega|)$  steps. Figs. 6b,c show

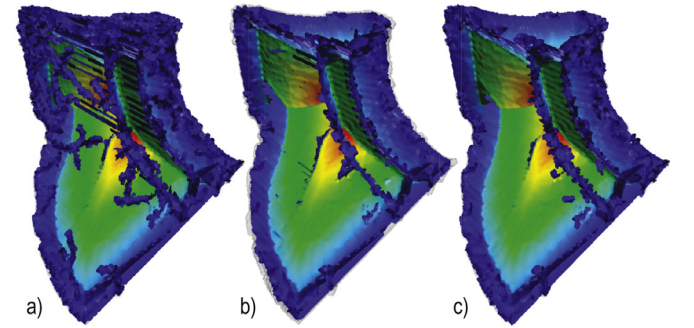


Fig. 6. Global saliency  $\bar{\sigma}$  computed by (b) streamlines and (c) seed method. (a) shows the saliency  $\bar{\sigma}$  for comparison purposes.

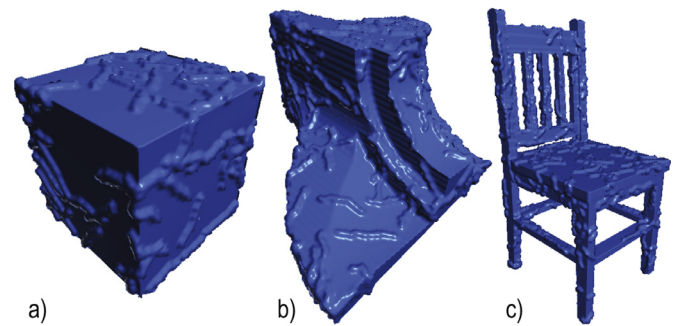


Fig. 7. Three models with curvilinear noise added (Section 3.2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$\sigma^*$  computed by the (slow) streamline method and the (fast) seed method for the fandisk shape in Fig. 7b. The two methods yield practically identical results, but the seed method removes slightly more noise sheets (blue), which is good. In contrast,  $\bar{\sigma}$  cannot disconnect the noise sheets from the core skeleton (Fig. 6a), just like in the case of the cube model discussed earlier (Fig. 5).

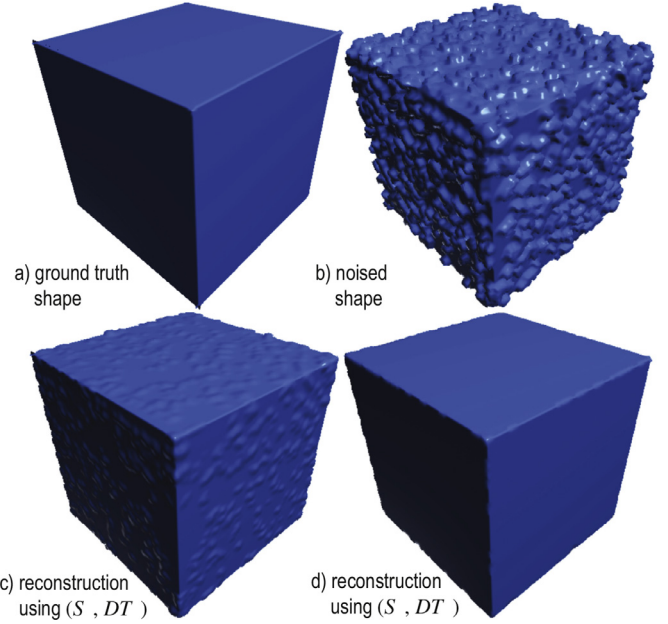
To quantitatively compare how well  $\sigma^*$  vs  $\bar{\sigma}$  remove edge-crossing noise, we designed an experiment similar to the one in

**Table 2**  
Comparison of detectors  $\bar{\sigma}$  and  $\sigma^*$  for removing curvilinear noise.

|         | $\bar{\sigma}$                  |                                      | $\sigma^*$ (streamlines)        |                                      | $\sigma^*$ (seed propagation)   |                                      |
|---------|---------------------------------|--------------------------------------|---------------------------------|--------------------------------------|---------------------------------|--------------------------------------|
|         | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ | $\nu(\bar{S}_\Omega, S_\Omega)$ | $\epsilon(\bar{S}_\Omega, S_\Omega)$ |
| bear    | 0.0139                          | <b>0.6362</b>                        | <b>0.0136</b>                   | 0.6393                               | 0.0137                          | 0.6370                               |
| bunny   | 0.0197                          | <b>0.6201</b>                        | <b>0.0192</b>                   | 0.6219                               | 0.0194                          | 0.6203                               |
| cat     | 0.0185                          | <b>0.6268</b>                        | <b>0.0175</b>                   | 0.6337                               | 0.0180                          | 0.6281                               |
| chair   | 0.1228                          | 0.1911                               | 0.1181                          | 0.2213                               | <b>0.1181</b>                   | <b>0.1859</b>                        |
| fandisk | 0.0327                          | <b>0.4538</b>                        | 0.0330                          | 0.4556                               | <b>0.0319</b>                   | <b>0.4563</b>                        |
| cube    | 0.1174                          | 0.1264                               | 0.1176                          | 0.1258                               | <b>0.1152</b>                   | <b>0.1233</b>                        |
| pot     | 0.0085                          | <b>0.6791</b>                        | 0.0084                          | 0.6797                               | <b>0.0084</b>                   | 0.6794                               |

**Section 3.1.** We added curvilinear noise to several 3D shapes – the idea behind is that such noise has a much higher chance to intersect shape edges than the zero-dimensional point-like noise in Fig. 4b, and creates precisely the challenges shown in Fig. 2b. To synthesize noise, at  $p$  randomly selected points on the shape surface  $\partial\Omega$ , we define a random tangent direction  $\mathbf{d}$ , and trace a curve  $\mathcal{C}$  of length  $l$  voxels on  $\partial\Omega$  in direction  $\mathbf{d}$ . During tracing, we jitter  $\mathbf{d}$  in the tangent plane to  $\partial\Omega$ . Finally, we deform  $\partial\Omega$  by convolving  $\mathcal{C}$  with a 3D ellipsoid-kernel filter of radius  $r$  in the tangent plane to  $\partial\Omega$  and height  $h$  in normal-to-surface direction. For our experiments, we set  $p = 0.001|\partial\Omega|$ ,  $r = 3$ ,  $h = 3$ , and  $l = 50$  (all in voxels).

We compare  $\sigma^*$ , computed both by streamline tracing and the seed-based propagation method, with  $\bar{\sigma}$  by computing the errors  $\nu$  and  $\epsilon$  (Eqs. (5), (6)) to measure how well the ground-truth (noise-free) shape can be recovered. Table 2 shows the results. The  $\sigma^*$  detector, computed by seed propagation, is the most accurate with respect to ground-truth for roughly half the shapes. For the other half,  $\sigma^*$  is below one percent less accurate than  $\bar{\sigma}$  or  $\sigma^*$  computed by streamline tracing. Denoising using  $\bar{\sigma}$  has barely any effects here (images not shown for space constraints), since most noise intersects the shape edges. In contrast, denoising using  $\sigma^*$  removes almost all the curvilinear noise (see results further in Fig. 11). Hence, we next use  $\sigma^*$  as our regularization metric in the remainder of this paper.



**Fig. 8.** Removing reconstruction ripples by postprocessing either the skeleton (c) or the distance transform (d). See Section 3.3.

### 3.3. Problem C: Removing reconstruction ripples

As explained in Section 2.3, noise added to a shape perturbs both its skeleton and its distance transform. To achieve a better reconstruction, we could (a) either postprocess the (regularized) noisy skeleton  $\bar{S}_\Omega$  so it gets closer to the skeleton  $S_\Omega$  of the clean shape  $\Omega$ , or (b) postprocess the distance transform  $DT_\Omega$  so it gets closer to  $DT_\Omega$ . To study which option yields better results, we take a ground-truth (noise-free) cube shape  $\Omega$  and create a noised version  $\bar{\Omega}$  by adding noise with  $p = 0.02|\partial\Omega|$ ,  $r = 2$ , and  $l = 5$ , following the model described in Section 3.2. Next, we reconstruct smoothed shapes from the MAT combinations  $(S_\Omega, DT_\Omega)$  and  $(\bar{S}_\Omega, DT_\Omega)$  respectively. The first MAT corresponds to situation (a) in which we would be able to postprocess the noisy  $\bar{S}_\Omega$  to perfectly recover the clean  $S_\Omega$ . The second MAT corresponds to situation (b) in which we would be able to postprocess the noisy  $DT_\Omega$  to perfectly recover the clean  $DT_\Omega$ . Fig. 8 shows the results. Option (b) yields a far closer result to the ground truth. Also, since postprocessing a surface skeleton consisting of many complex 3D sheets is technically much more complex than postprocessing a distance transform voxel-volume, we next settle for option (a).

We postprocess the noisy  $DT_\Omega$  by computing  $\mathcal{K}(\{DT_\Omega(\mathbf{y}) | \mathbf{y} \in \Omega \wedge \|\mathbf{y} - \mathbf{x}\| \leq r\})$ , i.e., convolving it with a kernel  $\mathcal{K}$  of radius  $r$ . We tested four kernels: mean ( $\mathcal{K}_E$ ), median ( $\mathcal{K}_{med}$ ), constrained opening ( $\mathcal{K}_{sup}(\mathcal{K}_{inf})$ ), where  $\mathcal{K}_{sup}$  and  $\mathcal{K}_{inf}$  are the supremum, opening

tively infimum kernels), and minification, defined by

$$\mathcal{K}_{minf}(\mathbf{x}) = \begin{cases} \mathcal{K}_{inf}(\mathbf{x}) + r, & \text{if } DT_\Omega(\mathbf{x}) \leq r \\ \mathcal{K}_{inf}(\mathbf{x}), & \text{otherwise} \end{cases} \quad (7)$$

All kernels can be applied *implicitly*, i.e., on the entire 3D volume  $\Omega$ , or *explicitly*, i.e., using only the distance transform values of points in the regularized skeleton. Table 3 compares the reconstruction error between a noise-free shape  $\Omega$  and the smoothed shape  $\bar{\Omega}$  obtained from a noised version of  $\Omega$ , computed as  $\epsilon(\Omega, \bar{\Omega}) = 1 - |\Omega \cap \bar{\Omega}| / |\Omega \cup \bar{\Omega}|$ , for all four kernels, implicit and explicit versions, all with a radius  $r = 3$  voxels. Noise is generated using  $p = 0.014|\partial\Omega|$ ,  $r = 2.1$ ,  $h = 3$ , and  $l = 1$ , following the model in Section 3.2. We use here a zero-dimensional (point-like,  $l = 1$ ) noise so as to remove edge-crossing effects, which were handled separately in Section 3.2. Table 3 shows that, except  $\mathcal{K}_{minf}$ , all kernels are almost always more accurate in their explicit versions. This is explained by the fact that only regularized skeleton points contribute to filtering, thereby excluding ligature sheets, which are caused by the added noise. In contrast,  $\mathcal{K}_{minf}$  gets better results in its implicit version as it evaluates more neighbor points (not only those on the skeleton), so it has a better chance in finding a minimum value.

Reconstruction errors are quite similar for different filters. There does not seem to be a best filter for all models. To get more insight, we visually examine the filters' effects. Fig. 9 shows the reconstruction results using the explicit version of  $\mathcal{K}_E$ ,  $\mathcal{K}_{med}$ ,  $\mathcal{K}_{sup}(\mathcal{K}_{inf})$ , and the implicit version of  $\mathcal{K}_{minf}$ , selected as such



**Table 3**

Comparison of four distance-transform smoothing kernels, implicit and explicit versions. The smallest reconstruction error (implicit vs explicit) for each kernel is in bold. The smallest reconstruction error for a model, all kernels, is in italic.

|         | $\mathcal{K}_E$ |               | $\mathcal{K}_{minf}$ |          | $\mathcal{K}_{sup}(\mathcal{K}_{inf})$ |               | $\mathcal{K}_{med}$ |               |
|---------|-----------------|---------------|----------------------|----------|--|---------------|---------------------|---------------|
|         | implicit        | explicit      | implicit             | explicit | implicit                               | explicit      | implicit            | explicit      |
| bear    | 0.0325          | <i>0.0129</i> | <b>0.0758</b>        | 0.1151   | 0.0481                                 | <b>0.0145</b> | 0.0359              | <b>0.0134</b> |
| bunny   | 0.0272          | <b>0.0161</b> | <b>0.0702</b>        | 0.1032   | 0.0366                                 | <i>0.0154</i> | 0.0305              | <b>0.0171</b> |
| cat     | 0.0479          | <i>0.0268</i> | <b>0.0862</b>        | 0.1162   | 0.0749                                 | <b>0.0294</b> | 0.0509              | <b>0.0276</b> |
| chair   | 0.0516          | <b>0.0489</b> | <i>0.0132</i>        | 0.1906   | 0.1015                                 | <b>0.0453</b> | 0.0736              | <b>0.0499</b> |
| fandisk | 0.0184          | <b>0.0183</b> | <b>0.0216</b>        | 0.0955   | 0.0305                                 | <i>0.0151</i> | 0.0245              | <b>0.0190</b> |
| hammer  | 0.0734          | <b>0.0329</b> | <b>0.0557</b>        | 0.1835   | 0.1314                                 | <i>0.0279</i> | 0.0856              | <b>0.0368</b> |
| cube    | <b>0.0076</b>   | 0.0171        | <b>0.0008</b>        | 0.0715   | <i>0.0002</i>                          | 0.0141        | 0.0242              | <b>0.0170</b> |
| pot     | 0.0097          | <b>0.0047</b> | <b>0.0277</b>        | 0.0306   | 0.0100                                 | <i>0.0043</i> | 0.0130              | <b>0.0047</b> |

**Fig. 9.** Comparison of different filters  $\mathcal{K}$  for distance transform smoothing. See Section 3.3.

given the earlier findings from Table 3. We see that all filters  $\mathcal{K}$  improve upon the filter-less reconstruction (Fig. 9b).  $\mathcal{K}_{minf}$  yields the visually smoothest result, with noticeably cleaner surfaces, even though it yields a slightly higher reconstruction error than  $\mathcal{K}_{sup}(\mathcal{K}_{inf})$  (Table 3). Hence, we choose  $\mathcal{K}_{minf}$  as our filter of choice to use next.

#### 4. Implementation

We implemented our method in C++ using OpenGL point-based rendering of  $\partial\Omega$  with radial splat kernels for fast display of binary voxel models (see Fig. 10 for the full pipeline) and the following components:

**Skeletonization:** We tested several methods for computing the MAT ( $S_\Omega$ ,  $DT_\Omega$ ) and feature transform  $FT_\Omega$ : mass advection [54], multiscale skeletons [50], the method of Reniers et al. [52], and the Integer Medial Axis (IMA) [46]. Among these, IMA is the simplest to implement, and of complexity  $O(|\Omega|)$ , which is in practice over one order of magnitude faster than the other methods. IMA delivers noisier skeletons than the other methods. This is not a problem since we anyways regularize the skeleton next. Hence, we choose IMA to compute  $S_\Omega$ ,  $DT_\Omega$ ) and  $FT_\Omega$  (Fig. 10, step 1). Several other

mesh-based methods for computing surface skeletons exist – that is, which represent both  $\Omega$  and  $S_\Omega$  as a mesh rather than in voxel space [53,66]. We cannot easily use such mesh-based methods as several steps of our framework are designed to work in a voxel setting, i.e., computing  $\sigma^*$  by the flood fill method (Section 3.2) and postprocessing  $DT_\Omega$  to remove ripples (Section 3.3); and since not all these methods deliver the feature transform  $FT_\Omega$ . Moreover, as stated in the introduction, our focus is feature-preserving smoothing of voxel shapes. As such, we considered only voxel-based 3D skeletonization methods, of which IMA is our choice.

**Importance metric:** Following [50,52], we define  $\rho$  as the geodesic distance between a skeleton's feature points. We compute  $\rho$  following the Dijkstra shortest-path search method on the voxel connectivity graph of the  $\partial\Omega$  voxels (Fig. 10, step 1). For details and source code, we refer to [52].

**Salience:** We next compute the vector field  $\mathbf{v}$  needed to evaluate  $\bar{\sigma}$  (Fig. 10, step 2) using one of the four methods described in Section 3.1. As explained there, these methods offer different trade-offs between implementation simplicity and accuracy, which one can choose from. Following this, we compute the global salience  $\sigma^*$  (Fig. 10, step 4) following the simple flood-fill process explained in Section 3.2. The simplified skeleton  $\bar{S}_\Omega$  is then trivially



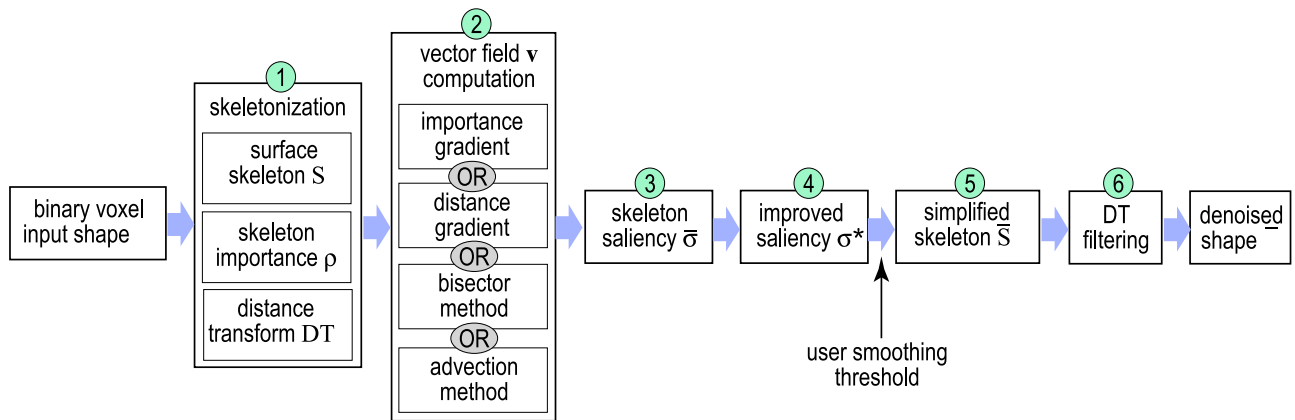


Fig. 10. Pipeline of the proposed feature-preserving smoothing method. See Section 4.

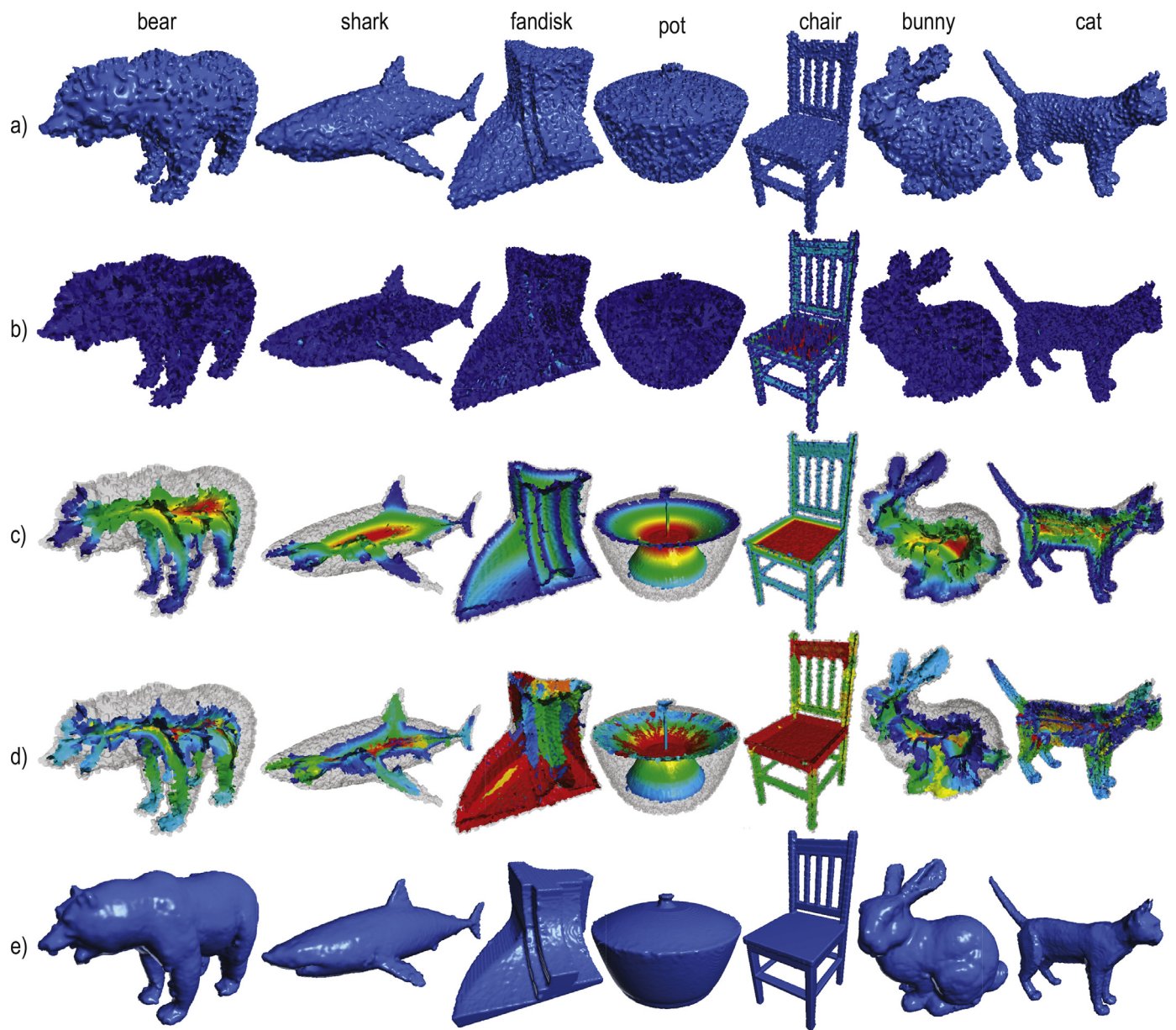


Fig. 11. Noisy models (a) with raw skeletons (b), regularized skeletons colored by distance transform (c) and saliency metric  $\sigma^*$  (d), and final smoothed shapes where salient shape features such as flat areas and edges are recovered from the noise (e). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

obtained by selecting all voxels of  $S_\Omega$  where  $\sigma^*$  is above the user-chosen smoothing level (Fig. 10, step 5), whose setting is discussed next in Section 6.

**Reconstruction:** We compute the filtered distance transform  $DT_\Omega * \mathcal{K}_{minf}$  (Fig. 10, step 6), and finally reconstruct the smoothed shape  $\bar{\Omega}$  as the union-of-balls centered on the regularized skeleton  $\bar{S}_\Omega$  having as radii the We use for this the fast reverse Euclidean distance transform implementation in [67] which is  $O(|\Omega|)$ .

## 5. Results

Fig. 11 shows our method applied on a mix of synthetic shapes with smooth surfaces separated by sharp edges (*fandisk*, *pot*, *chair*) and organic shapes having more curved surfaces separated by less sharp edges (*bear*, *dolphin*, *bunny*, *cat*). The shapes also feature thick parts (all except *chair*), tubular thin parts (*cat*, *chair*) and slab-like parts (*dolphin*, *chair*). For each model, we show the noised shape  $\Omega$ , its raw skeleton  $S_\Omega$ , the skeleton  $\bar{S}_\Omega$  regularized by  $\sigma^*$ , and the smoothed shape  $\bar{\Omega}$ . The latter is computed using a smoothing threshold  $\sigma_0 = 18$ , experimentally chosen by increasing  $\sigma_0$  until all noise is visually gone.

**Noise model:** We use here denser, and more challenging, noise than in all examples shown so far, given by the parameters  $p = 0.02|\partial\Omega|$ ,  $r = 2.5$ ,  $h = 3$ ,  $l = 6$  (see Section 3.2). This creates multiscale noise of quite high amplitude, densely covering the shape surface, which is challenging to remove. The use of synthetic noise in testing denoising methods is known in the literature, see e.g. [68] and [69]; for instance, the noised *fandisk* in Fig. 11 is very similar to Fig. 12 in [68]. Such noise is stronger (denser and/or higher-amplitude) than typical scanning noise, and it is hard to find real-world scans exhibiting this noise level. Moreover, for voxel models, scanning noise only shows up at high resolutions, roughly  $1000^3$  voxels or higher; to demonstrate our method's denoising abilities on voxel models of lower resolutions, we need to generate noise synthetically.

The smoothed shapes produced by our method are virtually overall noise-free and recover the underlying edges, flat surfaces, curved surfaces, and tubular parts well. We also see that the surface skeletons extracted by the IMA method are quite complex for all models (Fig. 11b). This is not an issue in our context, in contrast to other applications of skeletons, e.g. segmentation [14] or classification [64]. As explained, our method does not need a *clean* skeleton, since its regularization (done by upper thresholding  $\sigma^*$ ) is sufficient for the union-of-ball reconstruction for smoothing purposes. Fig. 11c,d show the regularized skeletons  $\bar{S}_\Omega$  colored by the distance transform  $DT_\Omega$ , respectively the salience  $\sigma^*$ . We see how the regularization removes the noise and ligature sheets of the raw skeletons so as to deliver the desired feature-preserving smoothing. Also, we see that the distance skeleton transform has low values on large portions of the surface skeleton such as the bear's muzzle and paws, shark's fins, fandisk edges, pot lid rim, bunny's ears, and cat's paws and tail (blue in Fig. 11c). These skeleton parts, close to the shape's surface, would have been removed by naively thresholding the importance  $\rho$ , resulting in unwanted loss or smoothing of the corresponding shape features. In contrast, the  $\sigma^*$  metric has high values in these areas (warmer colors in Fig. 11d), so the corresponding shape salient features are kept after smoothing, as visible in Fig. 11e. Running times for our method on a Linux desktop PC at 3.2 GHz with 16 GB RAM are shown in Table 4b.

**Comparison with mesh-based methods:** We further compare our method with six well-known techniques for 3D shape smoothing: surface-preserving Laplacian smoothing [59], two-step smoothing [7], algebraic point-set surfaces (APSS) [8], robust improved moving least squares (RIMLS) [11], Wei et al.'s kernel low-rank recovery [69], and the rolling guidance normal filter (RGN) [68]. The first four methods are readily available in MeshLab [70], which fa-

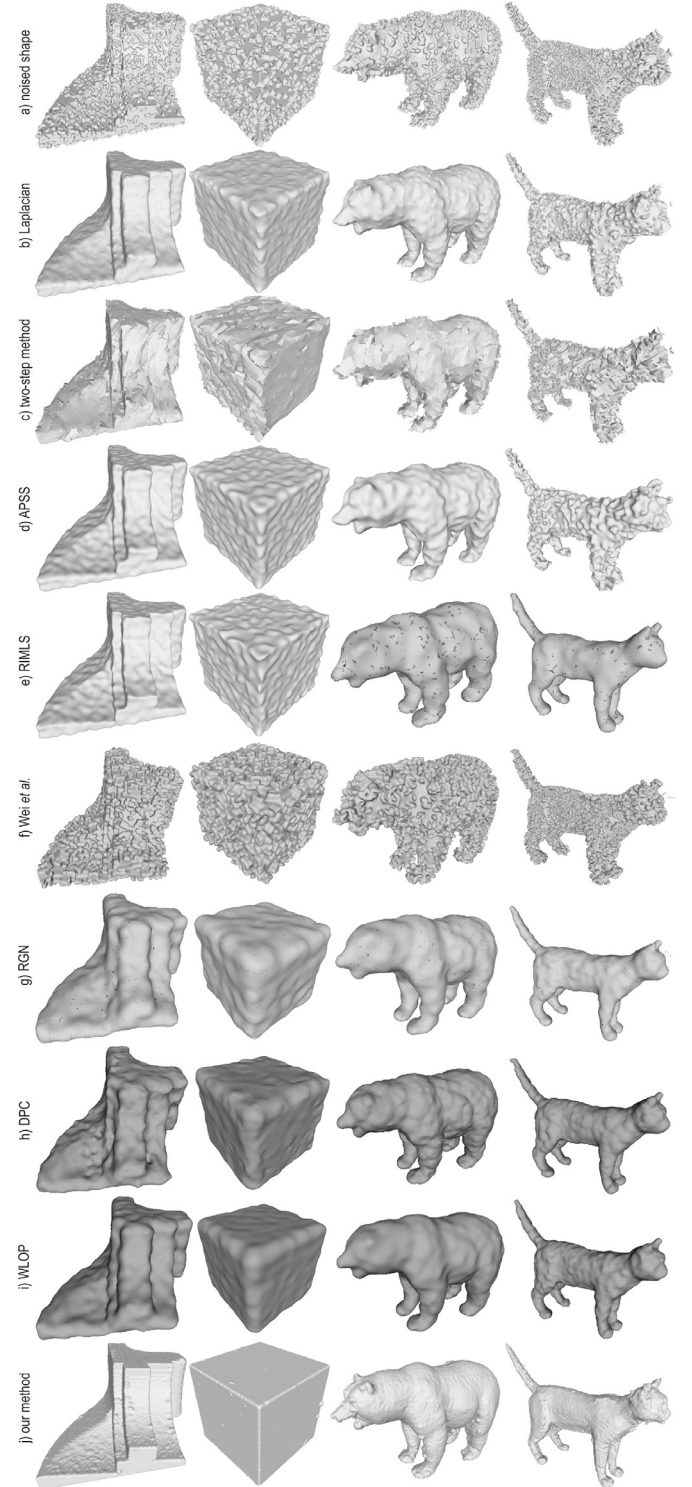


Fig. 12. Comparison of our method with other shape denoising techniques. See Section 5.

vors comparison replicability. For the method of Wei et al. [69], we did not have the implementation, but provided the noised meshes to the authors who delivered us the results they obtained from them. RGN iteratively applies a joint bilateral filter to face normals at a specified scale, which empirically smooths small-scale geometric features while preserving large-scale ones. We used the public RGN implementation on the website of the first author of [68]. Table 4a lists the parameter values for the compared methods that



**Table 4**

(a) Parameters of compared methods. (b) Our method's processing time for different shapes.

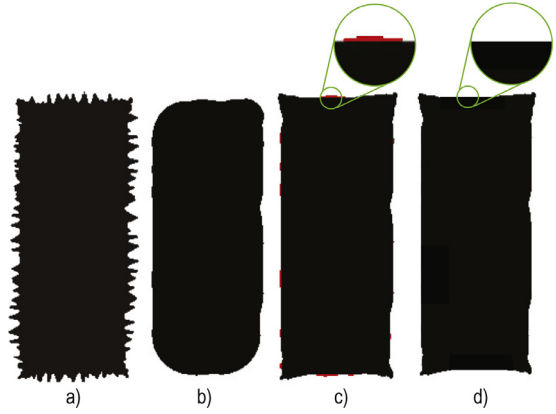
| Method          | Parameters (other than default)       |       | Shape   | Resolution       | Time (s) |
|-----------------|---------------------------------------|-------|---------|------------------|----------|
| Laplacian       | max. normal deviation                 | 60°   | shark   | 420 <sup>3</sup> | 52.3     |
| Two-step method | smoothing steps                       | 40    | bear    | 276 <sup>3</sup> | 63.7     |
|                 | degree                                | 60    | bunny   | 216 <sup>3</sup> | 98.09    |
|                 | normal smoothing steps                | 20    | cat     | 420 <sup>3</sup> | 108.7    |
|                 | vertex fitting steps                  | 20    | chair   | 420 <sup>3</sup> | 173.1    |
|                 | scale                                 | 15    | fandisk | 276 <sup>3</sup> | 481.7    |
| APSS            | scale                                 | 15    | hammer  | 520 <sup>3</sup> | 39.2     |
| RIMLS           | scale                                 | 15    | cube    | 148 <sup>3</sup> | 31.7     |
| Wei et al.      | $\sigma_5 = 0.45, n_1 = 30, n_2 = 20$ |       | pot     | 276 <sup>3</sup> | 203.6    |
| WLOP            | scale                                 | 7..15 |         |                  |          |

we had to change from the MeshLab defaults so as to obtain the best feature-preserving smoothing for each method. For Wei et al., the authors provided the optimal parameters  $\sigma_5$ ,  $n_1$ ,  $n_2$  themselves after experimentation (for explanations of these, we refer to [69]). For RGN, we used the parameter values  $\sigma_5 = 5$ ,  $\sigma_r = 0.8$ , and  $n = 5$  iterations.

To make the task more challenging, we noised the input shapes using dense and high-amplitude curvilinear noise with parameters  $p = 0.01|\partial\Omega|$ ,  $r = 2.1$ ,  $h = 7$ , and  $l = 4$ , following the rationale outlined at the beginning of this section. Since all above methods are mesh-based, we converted our voxel shapes to meshes using isosurfacing. Our method recovers the original shapes significantly better than the other tested methods (Fig. 12). Note that the wavy artifacts in Fig. 12f are due to the isosurface extraction from the limited-resolution voxel models. The method of Wei et al. only removed a very small amount of the present noise from all the tested models.

**Comparison with point-cloud methods:** We also compare our method with two point-cloud denoising methods: WLOP consolidation [71] and deep points consolidation (DPC) [72]. WLOP uses a weighted locally-optimal projection operator to produce a set of denoised points over the original point cloud, trying to improve normal estimates through local PCA. These normal estimates are further consolidated by an iterative, priority-driven, normal propagation step. DPC improves upon WLOP by augmenting each surface point by an inner point that resides on the so-called meso-skeleton. The strength of this representation is that it combines both local and non-local geometric information. Both WLOP and DPC take as input the point cloud given by the vertices of the noisy meshes which we generated from our noisy voxel models via isosurfacing. Since both methods redistribute consolidated points over the original point cloud, we downsampled the input point cloud to about 30K points. For both methods, we used the code available on the website of the first author of [72], with default parameters, except the WLOP scale. For each model, we set the WLOP scale to values between 7 and 15 to strike a tradeoff between feature preservation and outlier removal. With these settings, WLOP took between 120 to 340 s, while DPC took between 221 and 430 s. Similar to the mesh methods, WLOP and RPC succeed in removing almost all noise, but also smooth out salient edges and corners.

Several observations emerge from the above comparison. We see that both mesh-based and point-cloud-based methods have some issues in removing noise and preserving salient edges or corners. Depending on the method and model, there are areas in which either noise is not removed, or salient features are smoothed out. This is explained by the nature of the noise present in these shapes (strong and multiscale) – it is hard to find parameters that separate such noise from features since they both occur on a range of scales. Our method succeeds in doing this separation better since its operation is driven by the shape's surface skeleton and global saliency  $\sigma^*$ , which, as explained, are both global and multiscale descriptors: The classification of a surface voxel in noise vs feature depends on its corresponding skeleton voxel and the  $\sigma^*$



**Fig. 13.** Comparison of our method (d) with naive skeleton pruning (b) and saliency-based denoising [19] (c) for a 2D binary pixel shape (a). Insets show distance-transform ripples (red in (c)) that our method can remove. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

value thereof, which both in turn are computed from non-local shape properties. Actually,  $\sigma^*$  is computed *globally* from the entire shape via the importance analysis (Section 3.2, see again Fig. 4).

## 6. Discussion

We next discuss several aspects of our method:

**Generality:** Our method can remove noise of any type and from any kind of voxelized models.

**Parameters:** We have a single free parameter: the saliency  $\sigma^*$  to threshold to remove the noise (Section 3.2). As explained there,  $\sigma^*$  has the same dimensionality as the importance  $\rho$ , which in turn equals the length, in voxels, of the shortest-path on  $\partial\Omega$  between a skeleton's feature points. Hence, to remove a noise bump of base thickness  $2r$  and height  $h$  created by our noise model (Section 3.2), we can set  $\sigma^* = 2rh$ . This explains the values used for  $\sigma^*$  in this paper.

**Implementation:** Our method is simple to implement, needing only the IMA method for the raw skeleton  $S_\Omega$  [46] and Dijkstra's shortest-path tracing on  $\partial\Omega$  for  $\rho$  [52]. While its performance is still slower than mesh-based denoising alternatives (Table 4a), massive speed-ups can be obtained by computing the skeleton, distance, and feature transforms on the GPU as shown in [73]. For replication purposes, the source code of the method is available at [74].

**Comparison to 2D:** Our method shares goals, and technical points, with its 2D precursor [19], so it is useful to compare these methods for denoising 2D binary pixel shapes. Fig. 13 does this for the original noisy shape in [19]. Naive skeleton pruning based on the importance  $\rho$  eliminates noise but also rounds off salient corners (b). Saliency-based pruning of the 2D skeleton removes noise and keeps salient corners, but also creates some unwanted ripples



(marked red in Fig. 13c). Our method removes these (Fig. 13c). In detail, our method proposes mechanisms to handle three problems (A, B, C, see Section 3): Problem A is far less visible for 2D shapes, since the small-scale noise referred to in Section 3.1 are quite hard to see in 2D binary images; in contrast, due to the shading typically used when viewing 3D shapes, such small-scale noise is much easier to spot. Problem B is inherent to the 3D case, and does not occur in 2D shapes – there is no equivalent 2D configuration to the ‘noise crossing edges’ situation depicted in Fig. 2 (1b). Problem C is similar to A – the small-scale ripples due to the distance transform (see Section 3.3) are easier to spot for 3D shapes due to their shading than in 2D shapes – compare, for example, the ripples in Fig. 2 (4c) with the red insets in Fig. 13c. Concluding, our 3D method can be applied to denoise 2D binary pixel shapes, but its improvements vs the original method in [19] are quite small, which, we believe, do not justify the added implementation complexity. The main added value of our method is for 3D binary shape denoising, a case where the naive application of [19] yields poor results (Fig. 2).

**Limitations:** The smoothing quality of our method is influenced by the precision (centeredness) of the computed surface skeleton, which is in turn determined by the voxelization resolution. Higher-resolution models have skeletons that capture more surface details, thus allow to better detect and remove noise, but need more memory and processing time. More importantly, our method cannot distinguish between *detail* surface perturbations which are located far away from the salient shape edges and actual noise, and will remove the former while preserving the latter. However, we argue that this is an intrinsic problem of all smoothing methods – without additional context-specific information, such noise cannot be distinguished from actual details.

## 7. Conclusion

We have presented a new method for feature-preserving denoising of 3D volumetric models using surface skeletons. For this, we identify skeletal sheets created by undesired noise using an extension of the saliency metric earlier proposed in [19] for 2D shapes. Following simple threshold-based removal of these sheets, we reconstruct noise-free models using a modified Euclidean distance transform to remove low-frequency ripples. To our knowledge, our method is the first one demonstrating the added value of 3D surface skeletons for shape denoising. Our method has a single free parameter denoting the geometric scale of noise to be removed. In contrast to local denoising methods, our method fully preserves salient shape features such as edges and corners, as captured by the surface skeleton. We demonstrate our method for a variety of 3D shapes contaminated by significant amounts of noise and show that we can achieve better denoising than known alternative techniques.

All steps our method are amenable to SIMD parallelization, so exploiting architectures such as GPUs can lead to significant performance gains. Secondly, we consider adapting our method to handle mesh-based shape representations and their surface skeletons using recent GPU-accelerated mesh skeletonization methods [53]. Finally, we plan to extend our comparison with additional feature-preserving denoising methods.

## 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.

## CRediT authorship contribution statement

**Herman R. Schubert:** Conceptualization, Methodology, Software, Visualization, Investigation. **Andrei C. Jalba:** Software, Investigation, Visualization, Writing - original draft, Writing - review & editing. **Alexandru C. Telea:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision.

## Acknowledgments

We are highly grateful to J. Huang and co-authors for their effort in applying their method [69] on our benchmark datasets to provide us with comparison results.

## References

- [1] Wei M, Shen W, Qin J, Wu J, Wong T-T, Heng P-A. Feature preserving optimization for noisy mesh using joint bilateral filter and constrained Laplacian smoothing. *Opt Lasers Eng* 2013;51(11):1123–34.
- [2] Sun X, Rosin PL, Martin RR, Langbein FC. Noise in 3D laser range scanner data. In: *Proceedings of the IEEE SMA*; 2008. p. 37–45.
- [3] Wood Z, Hoppe H, Desbrun M, Schröder P. Removing excess topology from isosurfaces. *ACM Trans Graph* 2004;23(2):190–208.
- [4] Taubin G. A signal processing approach to fair surface design. In: *Proceedings of the ACM CGI*; 1995. p. 351–8.
- [5] Jones T, Durand F, Desbrun M. Non-iterative feature-preserving mesh smoothing. *ACM TOG* 2003;22:943–9.
- [6] Zhang W, Deng B, Zhang J, Bouaziz S, Liu L. Guided mesh normal filtering. *Comput Graph Forum* 2015;34:23–34.
- [7] Belyaev A, Ohtake Y. A comparison of mesh smoothing methods. In: *Proceedings of the conference of geometric modeling and computer graphics*; 2003. p. 83–7.
- [8] Guennebaud G, Gross M. Algebraic point set surfaces. *ACM TOG* 2007;26:23–36.
- [9] He L, Schaefer S. Mesh denoising via  $l_0$  minimization. *ACM TOG* 2013;32(4):64–72.
- [10] Digne J, Cohen-Steiner D, Alliez P, Goes FD, Desbrun M. Feature-preserving surface reconstruction and simplification from defect-laden point sets. *J Math Imag Vis* 2014;48(2):369–82.
- [11] Öztireli A, Guennebaud G, Gross M. Feature preserving point set surfaces based on non-linear kernel regression. *Comput Graph Forum* 2009;28:493–501.
- [12] Lange C, Polthier K. Anisotropic smoothing of point sets. *Comput Aid Geom Design* 2005;22(7):680–92.
- [13] Siddiqui K, Pizer S. *Medial representations – mathematics, algorithms and applications*. Springer; 2008.
- [14] Leymarie F, Kimia B. The medial scaffold of 3D unorganized point clouds. *IEEE TVCG* 2007;29(2):313–30.
- [15] Aylward S, Jomier J, Weeks S, Bullitt E. Registration and analysis of vascular images. *IJCV* 2003;55(2):123–38.
- [16] Sundar H, Silver D, Gagvani N, Dickinson S. Skeleton based shape matching and retrieval. In: *Proceedings of the SMI*; 2003. p. 130–7.
- [17] Storti D, Turkiyyah G, Lim C, Stal D. Skeleton-based modeling operations on solids. In: *Proceedings of the ACM SMA*; 1997. p. 141–54.
- [18] Liu H, Wu Z-H, Zhang X, Hsu D. A skeleton pruning algorithm based on information fusion. *Pattern Recogn Lett* 2013;34(10):1138–45.
- [19] Telea A. Feature preserving smoothing of shapes using saliency skeletons. In: *Proceedings of the VMLS*. Springer; 2012. p. 153–70.
- [20] Tagliasacchi A, Delame T, Spagnuolo M, Amenta N, Telea A. 3D skeletons: a state-of-the-art report. *CGF* 2016;35(2):573–97.
- [21] Saha PK, Borgefors G, di Baja GS. A survey on skeletonization algorithms and their applications. *Pattern Recogn Lett* 2016;76:3–12.
- [22] Guskov I, Sweldens W, Schröder P. Multiresolution signal processing for meshes. In: *Proceedings of the ACM SIGGRAPH*; 1999. p. 325–34.
- [23] Clarenz U, Diewald U, Rumpf M. Anisotropic geometric diffusion in surface processing. In: *Proceedings of the IEEE Visualization*; 2000. p. 397–405.
- [24] Rumpf M, Preusser T. A level set method for anisotropic geometric diffusion in 3D image processing. *SIAM J Appl Math* 2002;62(5):1772–93.
- [25] Bajaj C, Xu A. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans Graph* 2003;22(1):4–32.
- [26] Weickert J. *Anisotropic diffusion in image processing*. ECI Series. Teubner-Verlag; 1998.
- [27] Clarenz U, Rumpf M, Telea A. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE TVCG* 2004;10(5):516–24.
- [28] Tasdizen T, Whitaker R, Burchard P, Osher S. Geometric surface processing via normal maps. *ACM TOG* 2003;22(4):1012–33.
- [29] Taubin G. Linear anisotropic mesh filtering. IBM Research Report RC22213(W0110-051). IBMTJ. Watson Research; 2001.
- [30] Caissard T, Coeurjolly D, Lachaud J-O, Roussillon T. Laplace–beltrami operator on digital surfaces. *J Math Imag Vis* 2019;61(3):359–79.
- [31] Desbrun M, Meyer M, Schröder P, Barr AH. Implicit fairing of irregular meshes using diffusion and curvature flow. In: *Proceedings of the ACM SIGGRAPH*; 1999. p. 317–24.

- [32] Schneider R, Kobbelt L. Geometric fairing of irregular meshes for free-form surface design. *Comput Aid Geom Des* 2001;18(4):359–79.
- [33] Li B, Schnabel R, Klein R, Cheng Z, Dang G, Jin S. Robust normal estimation for point clouds with sharp features. *Computers & Graphics* 2010;34(2):94–106.
- [34] Huang H, Li D, Zhang H, Ascher U, Cohen-Or D. Consolidation of unorganized point clouds for surface reconstruction. *ACM TOG* 2009;28(5):176–85.
- [35] Zheng Y, Fu H, Au O, Tai C. Bilateral normal filtering for mesh denoising. *IEEE TVCG* 2011;16(10):1521–30.
- [36] Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. In: *Proceedings of the ACM SIGGRAPH 2003 Papers SIGGRAPH '03*. New York, NY, USA: ACM; 2003. p. 950–3. ISBN 1-58113-709-5. doi:10.1145/1201775.882368.
- [37] Hildebrandt K, Polthier K. Constraint-based fairing of surface meshes. In: *Proceedings of the SGP. Eurographics*; 2007. p. 203–12.
- [38] Shen Y, Barner KE. Fuzzy vector median-based surface smoothing. *IEEE TVCG* 2004;10(3):252–65.
- [39] Yagou H, Ohtake Y, Belyaev A. Mesh smoothing via mean and median filtering applied to face normals. In: *Proceedings of the GMP*; 2002. p. 124–31.
- [40] Yagou H, Ohtake Y, Belyaev AG. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. In: *Proceedings of the CGI. IEEE*; 2003. p. 43–51.
- [41] Malladi R, Sethian J. Image processing: flows under min/max curvature and mean curvature. *Graph Mod Imag Process* 1996;58(2):127–41.
- [42] Catté F, Dibos F, Koepfler G. A morphological scheme for mean curvature motion and applications to anisotropic diffusion and motion of level sets. *SIAM J Numer Anal* 1995;32(6):1895–909.
- [43] Guichard F, Morel J-M. Geometric partial differential equations and iterative filtering. In: *Proceedings of the ISMM*; 1998. p. 127–38.
- [44] Maragos P, Schafer R. Morphological filters. part ii: their relations to median, order-statistic, and stack filters. *IEEE TASSP* 1987;35:1170–84.
- [45] Blum H, Nagel R. Shape description using weighted symmetric axis features. *Pattern Recognit* 1978;10:167–80.
- [46] Hesselink WH, Roerdink JBTM. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE TPAMI* 2008;30(12):2204–17.
- [47] Strzodka R, Telea A. Generalized distance transforms and skeletons in graphics hardware. In: *Proceedings of the VisSym*; 2004. p. 221–30.
- [48] Ogniewicz RL, Kubler O. Hierarchic Voronoi skeletons. *Pattern Recognit* 1995;28:343–59.
- [49] Telea A, van Wijk JJ. An augmented fast marching method for computing skeletons and centerlines. In: *Proceedings of the VisSym*; 2002. p. 251–9.
- [50] Falcao A, Feng C, Kustra J, Telea A. Multiscale 2D medial axes and 3D surface skeletons by the image foresting transform. In: Saha P, Borgefors G, di Baja GS, editors. *Skeletonization – theory, methods, and applications*. Elsevier; 2017. p. 132–51.
- [51] Dey T, Sun J. Defining and computing curve skeletons with medial geodesic functions. In: *Proceedings of the IEEE SGP*; 2006. p. 143–52.
- [52] Reniers D, van Wijk JJ, Telea A. Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG* 2008;14(2):355–68.
- [53] Jalba A, Kustra J, Telea A. Surface and curve skeletonization of large 3D models on the GPU. *IEEE TPAMI* 2013;35(6):1495–508.
- [54] Jalba A, Sobiecki A, Telea A. An unified multiscale framework for planar, surface, and curve skeletonization. *IEEE TPAMI* 2015;38(1):38–45.
- [55] Siddiqi K, Bouix S, Tannenbaum A, Zucker S. Hamilton–Jacobi skeletons. *IJCV* 2002;48(3):215–31.
- [56] Bouix S, Siddiqi K, Tannenbaum A. Flux driven automatic centerline extraction. *Med Imag Anal* 2005;9(3):209–21.
- [57] Sud A, Foskey M, Manocha D. Homotopy-preserving medial axis simplification. In: *Proceedings of the SPM*; 2005. p. 103–10.
- [58] Chazal F, Lieutier A. The  $\lambda$ -medial axis. *Graph Models* 2005;67(4):234–45.
- [59] Sapiro G. *Geometric partial differential equations and image analysis*. Cambridge, University Press; 2001.
- [60] Tek H, Kimia B. Boundary smoothing via symmetry transforms. *J Math Imag Vision* 2001;14(3).
- [61] Bai X, Latecki L, Liu W-Y. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE TPAMI* 2007;29(3):449–62.
- [62] Liu H, Wu Z, Hsu D, Peterson B, Xu D. On the generation and pruning of skeletons using generalized Voronoi diagrams. *Pattern Recogn Lett* 2012;33(16):2113–19.
- [63] Pizer S, Siddiqi K, Szekely G, Damon J, Zucker S. Multiscale medial loci and their properties. *IJCV* 2003;55(2-3):155–79.
- [64] Giblin P, Kimia BB. A formal classification of 3D medial axis points and their local geometry. *IEEE TPAMI* 2004;26(2):238–51.
- [65] Rossi L, Torsello A. An adaptive hierarchical approach to the extraction of high resolution medial surfaces. In: *Proceedings of the 3DIMPVT*; 2012. p. 371–8.
- [66] Miklos B, Giessen J, Pauly M. Discrete scale axis representations for 3D geometry. *ACM TOG* 2010;29(4).
- [67] Coeurjolly D, Montanvert A. Optimal separable algorithms to compute the reverse Euclidean distance transformation and discrete medial axis in arbitrary dimension. *IEEE TPAMI* 2007;29(3):437–48.
- [68] Wang P-S, Fu X-M, Liu Y, Tong X, Liu S-L, Guo B. Rolling guidance normal filter for geometric processing. *ACM TOG* 2015;34(6):173:1–173:9.
- [69] Wei M, Huang J, Xie X, Liu L, Wang J, Qin J. Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery. *IEEE TVCG* 2019;25(10):2910–26.
- [70] Cignoni P, Callieri M, Corsini M, Dellepiane M, Ganovelli F, Ranzuglia G. MeshLab: an open-source mesh processing tool. In: *Proceedings of the Eurographics*; 2008. p. 129–36. <http://www.meshlab.net>.
- [71] Huang H, Li D, Zhang H, Ascher U, Cohen-Or D. Consolidation of unorganized point clouds for surface reconstruction. *ACM TOG* 2009;28(5):176:1–176:7.
- [72] Wu S, Huang H, Gong M, Zwicker M, Cohen-Or D. Deep points consolidation. *ACM TOG* 2015;34(6):176:1–176:13.
- [73] Cao T-T, Tang K, Mohamed A, Tan T-S. Parallel banding algorithm to compute exact distance transform with the GPU. In: *Proceedings of the I3D*; 2010. p. 83–90.
- [74] The Authors. Source code of proposed method. 2019. <http://www.staff.science.uu.nl/~telea001/Shapes/Salience3D>.