Pattern Recognition Letters
journal homepage: www.elsevier.com

Computing Refined Skeletal Features from Medial Point Clouds

Jacek Kustra[a,**], Andrei Jalba[b], Alexandru Telea[b]

[a]*Institute Johann Bernoulli, University of Groningen, Nijenborgh 9, 9747AG Groningen, The Netherlands*
[b]*Dept. of Math. and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands*

ABSTRACT

Medial representations have been widely used for many shape analysis and processing tasks. Large and complex 3D shapes are, in this context, a challenging case. Recently, several methods have been proposed that extract point-based medial surfaces with high accuracy and computational scalability. However, the resulting medial clouds are of limited use for shape processing due to the difficulty of computing refined medial features from such clouds. In this paper, we show how to bridge the gap between having a raw medial cloud and enriching this cloud with feature points, medial-point classification, medial axis decomposition into sheets, robust regularization, and Y-network extraction. We further show how such properties can be used to support several shape processing sample applications including edge detection and shape segmentation, for a wide range of complex 3D shapes.

1. Introduction

3D shape processing is a topic of rising importance due to developments in 3D shape modeling and acquisition techniques, such as structured light scanners, time-of-flight cameras or surface extraction from 3D volumetric scans. Such shapes $\Omega \subset \mathbb{R}^3$ are typically represented by boundary sampling, by polygonal meshes or dense point clouds. Processing such shapes is a vast research area encompassing analysis (detecting shape features like edges, flat areas, or thin regions), shape matching and registration, fairing and denoising, and segmentation.

While the above-mentioned processing can be done by using the shape's surface or boundary $\partial\Omega$, an alternative way is given by medial representations (Siddiqi and Pizer, 2009). Since their introduction by Blum (1967), such representations have received increasing attention due to their potential to simplify and generalize many shape processing operations.

Computing medial axes for 2D shapes is a well-understood topic, with efficient, robust, and easy-to-use methods (Ogniewicz and Kubler, 1995; Telea and van Wijk, 2002; Falcão et al., 2004; Siddiqi and Pizer, 2009). The 2D medial axis transform (MAT) full encodes a shape's boundary $\partial\Omega$ by its medial axis. In contrast, 3D shapes admit two kinds of medial structures: *Surface skeletons* generalize 2D medial axes and their MATs to fully capture the shape's geometry and topology. *Curve skeletons* capture the shape topology as a set of 1D curves, but do not provide the full boundary encoding given by the MAT. Hence, curve skeletons are good descriptors for tubular shapes such as vessel structures, but are less suited as analysis and processing tools for generic 3D shapes.

Recently, fast and accurate medial-surface computation was made possible for complex 3D models of millions of polygons (Ma et al., 2012; Jalba et al., 2013). Yet, such methods provide only a medial *point cloud*, which is far from sufficient for shape analysis and processing. Indeed, medial surfaces have a complex structure, consisting of several so-called medial *sheets*, or manifolds. Medial points have several *types*, which describe the kind of surface points they correspond to via the MAT (Giblin and Kimia, 2004). *Boundaries* of these curves, mapped via the MAT to shape edges (Reniers et al., 2008a; Siddiqi and Pizer, 2009), can be used for shape segmentation (Reniers and Telea, 2008b). Curves where sheets meet, also called *Y-intersection* curves, help with shape reconstruction and matching (Damon, 2006; Leymarie and Kimia, 2007; Chang et al., 2009). The individual *sheets* map to separate shape parts, enabling shape simplification and segmentation.

While the above medial features can be relatively easily computed for voxel-based representations (Reniers and Telea, 2008a,b), their computation for point-cloud medial surfaces is far from trivial (Jalba et al., 2013; Ma et al., 2012; Telea and Jalba, 2012; Kustra et al., 2014). Extracting features such as endpoints, branches, and junctions from curve skeletons is much easier than for medial surfaces, making the former more fre-

**Corresponding author:
e-mail: jacek.kustra@gmail.com (Jacek Kustra)

quently used in applications, even though they encode less information than the latter. To become effectively useful for real-world applications, medial-surface point-cloud methods need enhancement in the sense of classifying medial points, computing separate medial sheets, extracting sheet boundaries and Y-intersection curves, and mapping all these higher-level medial features robustly and efficiently to the shape surface.

In this paper, we show how to efficiently and robustly construct all above features from medial surface point-clouds, by combining several input-shape and medial properties. This makes 3D medial clouds as easy to use as the more commonly used curve skeletons, and opens the way for using such medial clouds for many shape analysis and processing applications.

The structure of this paper is as follows. Section 2 reviews related work, with a focus on computing high-level features for medial surfaces. Section 3 presents our methods to compute such features from raw medial clouds. Section 4 illustrates the applicability of our computed medial features for part-based and patch-based shape segmentation and classification. Section 5 discusses our techniques. Section 6 concludes the paper.

2. Related work

2.1. Basic definitions

We first outline the definitions required to introduce and discuss related work. Given a shape $\Omega \subset \mathbb{R}^3$ with boundary $\partial\Omega$, we first define its distance transform $DT_{\partial\Omega} : \mathbb{R}^3 \rightarrow \mathbb{R}^+$

$$DT_{\partial\Omega}(\mathbf{x} \in \Omega) = \min_{\mathbf{y}\in\partial\Omega} \|\mathbf{x}-\mathbf{y}\|. \tag{1}$$

The surface skeleton of Ω is next defined as

$$\begin{aligned} S_\Omega = \{\mathbf{x} \in \Omega \,|\, \exists \mathbf{f}_1 \in \partial\Omega, \mathbf{f}_2 \in \partial\Omega, \mathbf{f}_1 \neq \mathbf{f_2}, \\ \|\mathbf{x}-\mathbf{f}_1\| = \|\mathbf{x}-\mathbf{f}_2\| = DT_{\partial\Omega}(\mathbf{x})\}, \end{aligned} \tag{2}$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are two of the contact points with $\partial\Omega$ of the maximally inscribed ball in Ω centered at $\mathbf{x}$, also called *feature points* (Strzodka and Telea, 2004; Reniers and Telea, 2007; Hesselink and Roerdink, 2008). The vectors $\mathbf{v}_1 = \mathbf{f}_1 - \mathbf{x}$ and $\mathbf{v}_2 = \mathbf{f}_2 - \mathbf{x}$ are called feature vectors or *spoke vectors* (Stolpner et al., 2009), and are locally parallel to $\nabla DT_{\partial\Omega}$ (Siddiqi et al., 2002). The tuple $(S_\Omega, DT_{\partial\Omega})$ is called the medial axis transform (MAT) of Ω, and provides a full description of the boundary $\partial\Omega$, *i.e.* allows one to fully reconstruct $\partial\Omega$ given the MAT. For 3D shapes, S_Ω is a set of manifolds with boundaries which meet along a set of so called Y-intersection curves (Damon, 2006; Leymarie and Kimia, 2007; Chang et al., 2009). The set of all feature points of a skeleton point

$$F(\mathbf{x} \in S_\Omega) = \{\mathbf{f} \in \partial\Omega |\ \|\mathbf{x}-\mathbf{f}\| = DT_{\partial\Omega}(\mathbf{x})\} \tag{3}$$

is called the *feature transform* (FT) of $\mathbf{x}$ (Hesselink and Roerdink, 2008; Jalba et al., 2013). The cardinality $\|F(\mathbf{x})\|$ of $F(\mathbf{x})$ is related to the location of $\mathbf{x}$ on skeletal manifolds (Giblin and Kimia, 2004; Pizer et al., 2003) (see also Fig. 3): In the continuous case, points $\mathbf{x}$ inside skeletal manifolds, also called A_2^1 points, have exactly two feature points; points on Y-intersection curves of $k \geq 3$ manifolds, also called A_k^1 points, have k feature points; and points on manifold boundaries, also called A_3 points, have for $F(\mathbf{x})$ an entire circle sector or spherical segment, whose size is proportional with $DT_{\partial\Omega}(\mathbf{x})$. F and its inverse F^{-1} establish a bidirectional mapping between S_Ω and $\partial\Omega$, and are thereby important shape analysis instruments (Siddiqi and Pizer, 2009).

Besides surface skeletons (Eqn. 2), 3D shapes also admit so-called *curve skeletons*. These are curvilinear descriptors locally centered in the shape, according to various definitions (Cornea et al., 2007). Curve skeletons are effective in capturing the *topology* of tubular 3D shapes and also allow a limited reconstruction and part-based segmentation of such shapes (Reniers and Telea, 2008a; Arcelli et al., 2011). However, they do not provide the full MAT given by surface skeletons, nor do they capture features such as shape edges or enable patch-based segmentation, fairing, or simplification for the more general non-tubular shape family (Reniers and Telea, 2008a; Giesen et al., 2009; Miklos et al., 2010). A recent comparison of curve and surface skeletons is given in Sobiecki et al. (2014). Given the above, we focus our discussion next on surface skeletons.

A key component of any skeletonization method is *regularization*, *i.e.* removing (pruning) from S_Ω all points which encode small-scale surface features, such as noise. This is usually done by computing an importance measure $\mu : S_\Omega \rightarrow \mathbb{R}^+$ and next upper-thresholding it (Shaked and Bruckstein, 1998; Damon, 2006). Both local and global measures exist. *Local measures* cannot separate locally-identical, yet globally-different, contexts (see *e.g.* Fig. 1 in Reniers et al. (2008b)). Thresholding local measures can disconnect skeletons (Siddiqi et al., 2002; Pizer et al., 2003; Sud et al., 2005) and makes pruning less intuitive (Shaked and Bruckstein, 1998). Local measures include the angle between feature vectors, the distance-to-boundary (Amenta et al., 2001; Foskey et al., 2003; Sud et al., 2005), divergence metrics (Siddiqi et al., 2002; Bouix et al., 2005), first-order moments (Rumpf and Telea, 2002), and indicators of the multi-valueness of $\nabla DT_{\partial\Omega}$ (Stolpner et al., 2009, 2011). A good survey of such methods is given in Siddiqi and Pizer (2009). *Global measures* monotonically increase from the skeleton boundary ∂S_Ω inwards. Thresholding them yields connected skeletons which capture details at a user-given scale. Miklos et al. (2010) propose a measure based on the geometric properties of an union of balls (UoB) approximation of Ω (Giesen et al., 2009). Dey and Sun propose the medial geodesic function (MGF), equal to the length of the shortest geodesic between feature points (Dey and Sun, 2006; Prohaska and Hege, 2002). Reniers et al. (2008b) extend the MGF for surface and curve skeletons using geodesic lengths and surface areas between geodesics, respectively, inspired by collapse metrics used to extract 2D multiscale skeletons (Ogniewicz and Kubler, 1995; Falcão et al., 2004; Telea and van Wijk, 2002).

2.2. Computing medial surfaces and their features

Many methods have been proposed in the last decade to compute medial surfaces. These can be classified in thinning, field, and geometric (mesh-based) methods (Jalba et al., 2013; Arcelli et al., 2011; Miklos et al., 2010). Thinning removes $\partial\Omega$ voxels while preserving homotopy (Palagyi and Kuba, 1999; Bai et al., 2007; Pudney, 1998; Arcelli et al., 2011). Thinning is

simple and fast, but can be sensitive to rigid Euclidean transformations. Field methods find S_Ω along singularities of $DT_{\partial\Omega}$ or related fields (Leymarie and Levine, 1992; Kimmel et al., 1995; Wan et al., 2001; Rumpf and Telea, 2002; Siddiqi et al., 2002; Hesselink and Roerdink, 2008; Rossi and Torsello, 2012) and can be efficiently implemented on GPUs (Sud et al., 2005; Cao et al., 2010). Mesh methods use a surface sampling of $\partial\Omega$ only, which is cheaper and faster than the volumetric sampling of Ω used by field and thinning methods. They include Voronoi diagrams (Amenta et al., 2001; Dey and Zhao, 2003) and finding maximally inscribed balls (Leymarie and Kimia, 2007; Miklos et al., 2010; Ma et al., 2012; Jalba et al., 2013). Given their scalability, we focus next on mesh-based methods only.

Recent mesh-based surface skeletonization methods (Ma et al., 2012; Jalba et al., 2013) can extract highly accurate medial surfaces of complex meshed surfaces of millions of polygons in subsecond time using GPU acceleration techniques, and thereby make medial surfaces usable in interactive contexts. However, such methods deliver only a medial *point cloud*, *i.e.*, an unorganized set of medial points with two corresponding feature points. The practical usefulness of such descriptors is quite limited, as mentioned already in Ma et al. (2012) and Jalba et al. (2013), since one cannot directly reason about the relationships between several medial-cloud points or the relationship of such points with the surface points. We identify two classes of challenges in this respect, as follows.

Feature points: Computing the correct feature transform F (Eqn. 3) for each medial point is crucial for several subsequent operations. For instance, any application that uses Giblin's classification (Giblin and Kimia, 2004) requires exactly estimating $\|F(\mathbf{x})\|$ for all $\mathbf{x} \in S_\Omega$. While this is relatively easy for voxel-based skeletons (Reniers et al., 2008a; Reniers and Telea, 2008b) due to the regular sampling of both $\partial\Omega$ and S_Ω, this is far from trivial for medial clouds. Incorrect or incomplete estimation of F creates challenges for all regularization methods that use feature points or feature vectors. Consider for instance the class of MGF methods, which measure the geodesic distance μ between two given feature points $\mathbf{f}_1 \in F(\mathbf{x})$ and $\mathbf{f}_2 \in F(\mathbf{x})$ of a medial point $\mathbf{x}$ (Dey and Sun, 2006; Reniers et al., 2008b; Jalba et al., 2013). A_3 and $A^1_{k,k\neq 2}$ points, in Giblin's terminology, have more than two feature points. Which pair $\{\mathbf{f}_1, \mathbf{f}_2\} \subset F(\mathbf{x})$ should we use here? Using *any* such pair breaks off the continuity of μ over S_Ω, which in turn means that we cannot obtain noise-free *and* compact medial surfaces by simply thresholding μ.

Skeleton decomposition: Decomposing the medial surface into separate sheets is a key step for using this descriptor for further shape analysis or classification (Stolpner et al., 2011; Siddiqi and Pizer, 2009). The same applies to finding Y-intersection curves and sheet boundaries, *i.e.*, the medial scaffold (Leymarie and Kimia, 2007). This decomposition has been previously attempted by using local medial geometry properties. This is relatively easy to do for finely-sampled voxel skeletons (Reniers and Telea, 2008c), leading to using medial surfaces to create compelling multiscale shape segmentations (Reniers and Telea, 2008b). Doing all above for medial clouds is, however, far from trivial. As shown in Kustra et al. (2014), using generic point-cloud segmentation methods for 3D skeletons is doable, but extremely challenging, since medial surfaces consist of numerous intersecting manifolds with boundaries, which are hard to capture even by very dense point clouds. In addition, the parameter settings in Kustra et al. (2014) are far from trivial, as they are related to local medial geometrical properties, such as *maximum local connectivity angle*. This adds difficulty in decomposing the medial axis, and requires a caseby-case parameter followed by a visual check to ensure the medial axis is decomposed in a satisfactory manner. In this paper, a different approach is taken to compute these abstractions. Rather than using the local geometric properties, the relationships between the medial points $\mathbf{x}_i$ feature vectors $\mathbf{f}_1, \mathbf{f}_2$ and their correspondence to surface properties is used. The next sections address this method.

3. Computing refined medial features

To address the challenges outlined in Sec. 2.2, we next present several new methods for computing the above-mentioned higher-level features from skeletal point clouds. We start by showing how to robustly classify medial points following Giblin and Kimia (2004), find skeletal boundaries and Y-curves, and robustly regularize the medial surface (Sec. 3.1). We next use these features to robustly segment the medial surface into separate manifolds (Sec. 3.2). Figure 1 shows the steps of our feature-computation pipeline and also the applications enabled by it. As input for all these operations, we only assume a surface skeleton represented by an unstructured and unoriented point-cloud having exactly one skeleton point per surface point and exactly two feature-points per skeleton point, as computed *e.g.* by Ma et al. (2012); Jalba et al. (2013); Hesselink and Roerdink (2008).

3.1. Medial points classification

3.1.1. Estimating the feature transform

To classify unstructured medial clouds following Giblin and Kimia (2004), we first need to estimate the feature transform $F(\mathbf{x} \in S_\Omega)$ (Eqn. 3). As explained earlier, F is not directly available in most skeletonization methods; in particular, our point-cloud methods (Ma et al., 2012; Jalba et al., 2013) only compute two feature points per skeleton point. To find all feature points, we proceed as follows. Let $DT_\mathbf{x}$ be next a shorthand for $DT_{\partial\Omega}(\mathbf{x})$. For each skeleton point $\mathbf{x} \in S_\Omega$, we first find the closest points $F_\tau(\mathbf{x}) \subset \partial\Omega$ in a radius $DT_\mathbf{x} + \tau$, where τ is defined as a fraction of $\varepsilon\rho_{\partial\Omega}(\mathbf{x})$, where $\rho_{\partial\Omega}(\mathbf{x})$ is the average point density on $\partial\Omega$ in a small neighborhood around $\mathbf{f}_1(\mathbf{x}) \cup \mathbf{f}_2(\mathbf{x})$, and ε is a small constant set to 0.1. The slightly increased radius determines that the set $F_\tau(\mathbf{x})$ will conservatively contain *all* feature points of $\mathbf{x}$, *i.e.* $F(\mathbf{x}) \subset F_\tau(\mathbf{x})$. Setting τ to track the local sampling density of $\partial\Omega$ allows us to conservatively estimate F_τ for non-uniformly sampled meshes without introducing too many false-positives, *i.e.*, minimizing the set $F_\tau \setminus F$.

Given the finite tolerance τ and the discrete sampling of $\partial\Omega$, $F_\tau(\mathbf{x})$ will also contain surface points which are slightly further from $\mathbf{x}$ than feature points; this is especially salient for points $\mathbf{x}$ of type A_3, that map to circular or spherical sectors on $\partial\Omega$ via the feature transform. However, as we shall see next, the conservative estimation of $F(\mathbf{x})$ given by $F_\tau(\mathbf{x})$ does not pose any problems to our medial attribute computation.

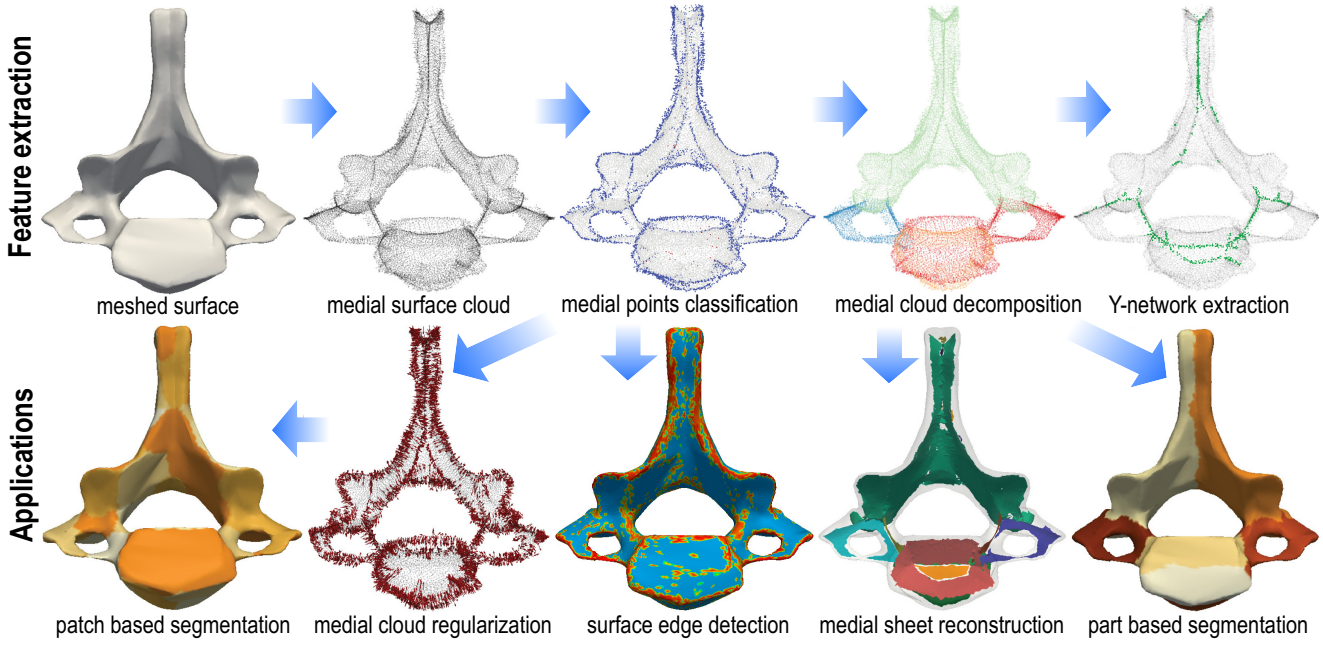

Fig. 1. Refined skeletal features computed from medial point clouds (top row) and subsequently enabled applications (bottom row).

3.1.2. Classification of medial points

Since $F_\tau(\mathbf{x})$ is essentially a dilated, or fuzzy, version of $F(\mathbf{x})$, it consists of one or several point *clusters* centered around actual feature points. A cluster $C_i(\mathbf{x}) \subset F_\tau(\mathbf{x})$ can be defined as

$$C_i(\mathbf{x}) = \{\mathbf{f} \in F_\tau(\mathbf{x}) | \max_{\mathbf{f}\in C_i} \min_{\mathbf{g}\neq\mathbf{f}\in C_i} \|\mathbf{f}-\mathbf{g}\| < \min_{\mathbf{f}\in C_i} \min_{\mathbf{h}\notin C_i} \|\mathbf{f}-\mathbf{h}\|\} \quad (4)$$

i.e., all points which are closer to each other than to any point from another cluster $C_{j\neq i}(\mathbf{x})$ for the same skeleton-point $\mathbf{x}$.

We observed that the number of these clusters is a good indicator of the type of the medial point $\mathbf{x}$: For A_3 points, there is one such cluster, whose diameter is proportional to $DT_\mathbf{x}$; for A_2^1 points, we find two clusters; and for $A^1_{k,k\geq 3}$ points, we find k clusters. To compute k, we cluster the point-set $F_\tau(\mathbf{x})$ by a single-linkage hierarchical agglomerative method based on the Euclidean distance between the points. Next, we cut the resulting dendrogram, or cluster-tree, at a distance value equal to the average local sampling density $\rho_{\partial\Omega}$. This results in k clusters. The value of k gives us the type of point $\mathbf{x}$, as explained above.

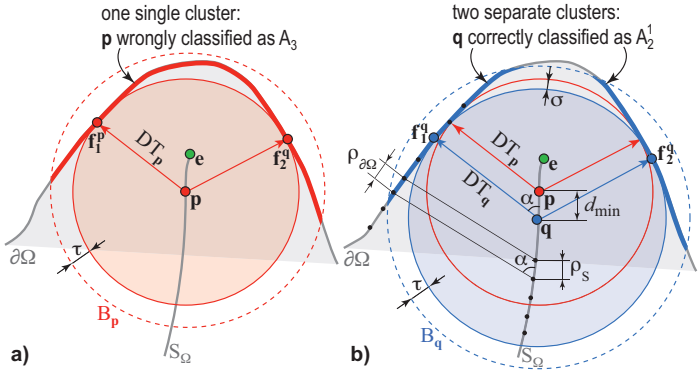

Fig. 2. Skeleton point classification based on fuzzy F_τ analysis. The figure is drawn for a 2D skeleton, for illustration simplicity.

Let us justify why k is a good point-type indicator. Fig. 2 a shows an *incorrect* classification of medial point $\mathbf{p}$ which is on the skeleton S_Ω branch ended by point $\mathbf{e}$. Since the intersection of $\partial\Omega$ with a ball $B_\mathbf{p}$ of radius $DT_\mathbf{p}+\tau$ and center $\mathbf{p}$ (dotted red circle) yields a single cluster (thick red line), $\mathbf{p}$ is incorrectly marked as A_3 rather than as A_2^1. This is caused by (1) the value τ used to compute F_τ being too large; (2) $\mathbf{p}$ being close to $\mathbf{e}$; and (3) the bump on $\partial\Omega$ corresponding to $\mathbf{e}$ being too shallow. Consider now the minimal distance d_{min} from $\mathbf{p}$ that we have to move on S_Ω away from $\mathbf{e}$ to find a point $\mathbf{q}$ which is *correctly* classified as A_2^1 (Fig. 2 b). This happens when the intersection of the ball $B_\mathbf{q}$ of radius $DT_\mathbf{q}+\tau$ and center $\mathbf{q}$ (dotted blue circle) yields two disconnected clusters on $\partial\Omega$ (marked thick blue). To find d_{min}, note that the maximal 'inward shift' between the upper parts of $B_\mathbf{p}$ and $B_\mathbf{q}$ equals $\sigma = DT_\mathbf{p} - DT_\mathbf{q} + d_{min}$. To cause the disconnection of the compact cluster in Fig. 2 a, σ must be larger than the maximal bump height on $\partial\Omega$ that fits in the sphere-shell of thickness τ, *i.e.*, $DT_\mathbf{p} - DT_\mathbf{q} + d_{min} > \tau$. Since $DT_\mathbf{q} - DT_\mathbf{p} = d_{min}\cos\alpha$, where α is the angle between a feature vector and the tangent plane to S_Ω, it follows that

$$d_{min} > \frac{\tau}{1-\cos\alpha}. \quad (5)$$

Separately, for a $\partial\Omega$ with local sampling density $\rho_{\partial\Omega}$, the corresponding skeleton sampling density is

$$\rho_S = \frac{\rho_{\partial\Omega}}{\sin\alpha}. \quad (6)$$

A correct classification should mark only a one sampling-point-thin 'band' of skeleton points as A_3 (skeleton boundary). If ρ_S is smaller than the minimal ball-shift d_{min} needed to change point type from A_3 to A_2^1, this band gets thicker, leading to the incorrect classification in Fig. 2 a. Substituting our value of $\tau = \varepsilon\rho_{\partial\Omega}$ (Sec. 3.1.1) in Eqn. 5, and next the values of d_{min} and ρ_S from Eqns. 5 and 6 in the inequality $\rho_S < d_{min}$, it follows that incorrect classification can only appear if $\varepsilon \geq \frac{1-\cos\alpha}{\sin\alpha}$. For our chosen value of $\varepsilon = 0.1$, this implies $\alpha \lesssim 11.4°$. In other words, for any medial sheets except those corresponding to highly obtuse angles on $\partial\Omega$, our method finds skeleton boundaries (A_3 points) which are precisely one sample-point thick.

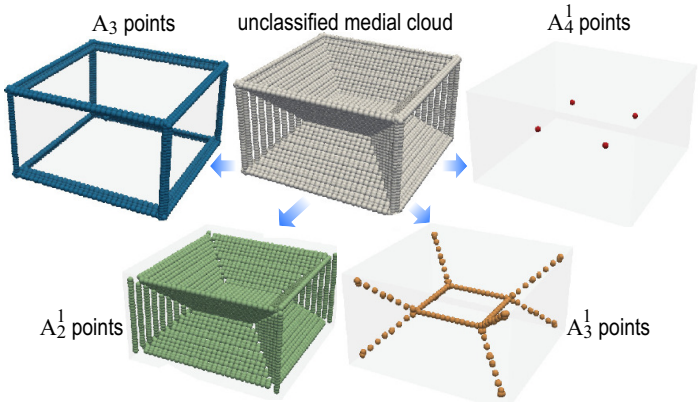

Fig. 3. Medial cloud classification into different point types.

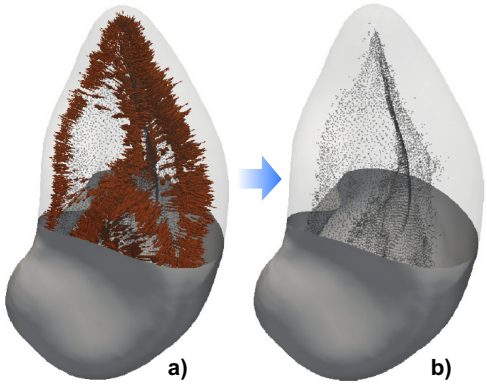

Fig. 4. Skeleton regularization. (a) Rounded spleen shape with feature vectors shown for A_3 points. (b) Skeleton regularized by filtering A_3 points.

Following the above, if $k = 1$ or $k = 2$, we can confidently say that we have found an A_3, respectively A_2^1, skeleton point. As k increases, the spatial separation of the clusters decreases too, so k does not reflect accurately the skeleton point type. We have empirically verified that the cluster count k accurately finds A_3^1 up to A_4^1 points for densely-sampled surfaces $\partial\Omega$. A more robust way to find such Y-curve points, that is far less sensitive on the sampling density of $\partial\Omega$, is described further in Sec. 3.2.2, based on the segmentation of S_Ω into individual medial sheets.

Related to our work, Reniers et al. (2008a) found A_3 points by computing the set difference between the full medial surface S and a simplified version S_τ of S, where τ is a small fixed value and simplification uses the MGF metric (Sec. 2.1). Compared to our approach, their method does not find $A_{k,k>1}^1$ points, and does not give an analysis of how to set parameter values.

3.1.3. Skeleton regularization using A_3 edge filtering

As outlined in Sec. 2.1, the MGF metric (Dey and Sun, 2006; Reniers et al., 2008b) provides very good regularization properties such as separating spurious skeleton points from important ones while maintaining skeleton connectivity. The MGF importance $\mu(\mathbf{x})$ of a medial point $\mathbf{x}$ equals the length of the longest shortest-geodesic on $\partial\Omega$ between *any* two feature points of $F(\mathbf{x})$. Hence, the MGF requires an accurate computation of the feature transform F (Eqn. 3). As discussed in Sec. 3.1.1, we compute a conservative F_τ which may contain tens of feature points for A_3-type points. Computing shortest-geodesics between all such point-pairs is very expensive. Given this cost, Jalba et al. (2013) and Reniers et al. (2008b) compute the MGF using only two feature points per skeleton point, *i.e.* implicitly consider all medial points to be of type A_2^1. This has two problems. First, the importance μ for A_3 points will be typically underestimated, since one has no guarantee of finding the *longest* shortest-geodesic connecting any two feature points. This, in turn, creates a relatively jagged appearance of the simplified skeleton. Secondly, computing the MGF is expensive for large models, even when using only two feature points per medial point and highly optimized GPU implementations (Jalba et al., 2013).

We propose here an alternative way to regularize medial surfaces by simply filtering A_3 points found by our classification. Figure 4 shows this for a shape having highly rounded edges, *i.e.* whose A_3 points have many feature points. This is the kind of shape where the aforementioned problem of the MGF metric occurs. Figure 4 a shows the medial cloud with feature vectors (in red) for the A_3 points. Figure 4 b shows our regularized skeleton, with all noisy points being removed. Since A_3 points appear only on the medial boundary by definition, our regularization does not create gaps or disconnect the medial surface. Since our method requires only a simple clustering of feature points based on their Euclidean distance, it is considerably faster than the MGF metric (see Sec. 5 for details). However, in contrast to the MGF, our method cannot deliver a *multiscale* of progressively simplified skeletons; we can only remove the finest scale of noisy boundary points. As such, our regularization is useful when one needs a clean and detail-rich surface skeleton for further processing, rather than a multiscale skeleton representation.

3.2. Surface skeleton decomposition

Besides classifying skeleton points, higher level features can be computed. One such feature is the decomposition of the medial surface into separate sheets, used in shape analysis and segmentation tasks (Leymarie and Kimia, 2007; Reniers and Telea, 2008b). While such decompositions can be computed relatively easy for voxel skeletons (Reniers and Telea, 2008c), this is challenging for medial clouds, especially when these contain a large number of complex sheets (Kustra et al., 2014).

We address this task by clustering the medial cloud based on a novel definition of medial sheets that uses the medial cloud properties (Sec. 3.2.1). Next, we use this decomposition to robustly find Y-intersection curves where several such sheets meet (Sec. 3.2.2). Finally, we use the feature transform to construct compact (meshed) sheet representations (Sec. 3.2.3).

3.2.1. Medial sheet computation

We first define the distance of two A_2^1 points $\mathbf{x}$ and $\mathbf{y}$ as

$$\delta(\mathbf{x},\mathbf{y}) = \sum_{\mathbf{a}\in F(\mathbf{x})} \min_{\mathbf{b}\in F(\mathbf{y})} MGF(\mathbf{a},\mathbf{b}), \tag{7}$$

where $MGF(\mathbf{a},\mathbf{b})$ is the medial geodesic function, *i.e.* the length of the shortest geodesic on $\partial\Omega$ between feature points $\mathbf{a}$ and $\mathbf{b}$ (Reniers et al., 2008b). Next, we define a medial sheet γ as all medial points having a distance δ lower than a threshold τ

$$\gamma = \{\mathbf{x} \in S_\Omega \mid \exists \mathbf{y} \in S_\Omega, \delta(\mathbf{x},\mathbf{y}) < \tau\}. \tag{8}$$

The rationale behind Eqn. 8 is that medial points $\mathbf{x}$ and $\mathbf{y}$ which are close *and* belong to the same sheet have small distances

(along $\partial\Omega$) between their corresponding feature points. This statement is supported as follows: Since medial sheets are locally smooth and have a low curvature (Pizer et al., 2003), their feature vectors vary smoothly and slowly locally; in turn, this implies that the corresponding feature points vary slowly and smoothly across $\partial\Omega$. Figure 5 illustrates this for a 2D shape (for simplicity): Medial points $\mathbf{x}$ and $\mathbf{y}$ are on the same sheet, and have small MGF distances between their feature points, thus a small $\delta(\mathbf{x},\mathbf{y})$. In contrast, medial points $\mathbf{w}$ and $\mathbf{z}$, which belong to different sheets, have at least two feature vectors pointing in different directions, thus a large $\delta(\mathbf{w},\mathbf{z})$.

Equations 7 and 8 define medial sheets without using any medial connectivity information, and thus allow us to segment a medial cloud into its sheets, as follows. First, we define a distance matrix M encoding the distances $\delta(\mathbf{x},\mathbf{y})$ between all medial point pairs. For efficiency, we only compute matrix entries that correspond to δ values below our chosen threshold τ, since the sheet definition (Eqn. 8) only requires to know when $\delta < \tau$. Secondly, when computing $\delta(\mathbf{x},\mathbf{y})$, if the length of the geodesic traced on $\partial\Omega$ from $\mathbf{x}$ to $\mathbf{y}$ exceeds τ, we stop tracing it and skip the respective matrix entry. Overall, this turns the computation and storage of M from a quadratic process in the number of medial points into a linear process, since any medial point $\mathbf{x}$ has only a limited number of points $\mathbf{y}$ at close distance δ from it. Finally, we use M as input for a single-linkage hierarchical clustering (Jain and Murty, 1999), which outputs a partition of S_Ω into a set of medial sheets γ_i, so that $\gamma_i \cap \gamma_{j\neq i} = \varnothing$ and $\cup_i \gamma_i = S_\Omega$. Figure 5 c illustrates the separated sheets of the medial surface of a palatine bone shape. Same-sheet points are marked by the same color. Although the medial cloud is quite complex, its sheets are cleanly separated. Such sheets can be processed to create compact representations thereof, as discussed next in Sec. 3.2.3.

Figure 6 compares our method for extracting sheets from a medial cloud with two other methods. Figure 6 a shows the method of Reniers and Telea (2008c), which works in brief as follows: Given a (voxel) medial surface S, its Y-network voxels S_Y are found based on the cardinality of the feature transform for A_3^1 points (Sec. 2.1). Next, separate medial sheets are found as being the connected components of the voxel set $S \setminus S_Y$. This method is quite sensitive to the voxel sampling of the input shape. For example, the cog wheel detail in Fig. 6 a (128^3 voxels) shows two separate components c_1 (red) and c_2 (purple), which actually are part of the same sheet. These are wrongly separated since (1) the sampling resolution disconnects the detected medial sheet half-way and (2) sheet detection is based on connected component finding. Figure 6 b shows the method of Kustra et al. (2014), which is a general-purpose clustering of 3D point clouds based on a similarity metric that accounts for both distance (computed between nearest neighbors in the point cloud) and local sheet flatness. Figure 6 c shows our method. As visible, both Kustra et al. (2014) and our method correctly detect a single sheet c instead of the two separate fragments c_1 and c_2. Images (d-i) further compare our method with Kustra et al. (2014) for two shapes and two different values of the nearest-neighbor count NN, the key parameter of Kustra et al. (2014). Two observations can be made here. First, we see how the results of Kustra et al. (2014) depend quite strongly on the NN choice. In contrast, our method does not need such parameter tuning. Secondly, and more importantly, the point-similarity used by Kustra et al. (2014) is purely *local*, as it uses only inter-point distances and local flatness of the sheets. In contrast, our method uses a distance function (Eqn. 7) which captures *global* shape properties, due to the underlying MGF function. This makes the sheet computation far less sensitive to local shape variations or small-scale noise.

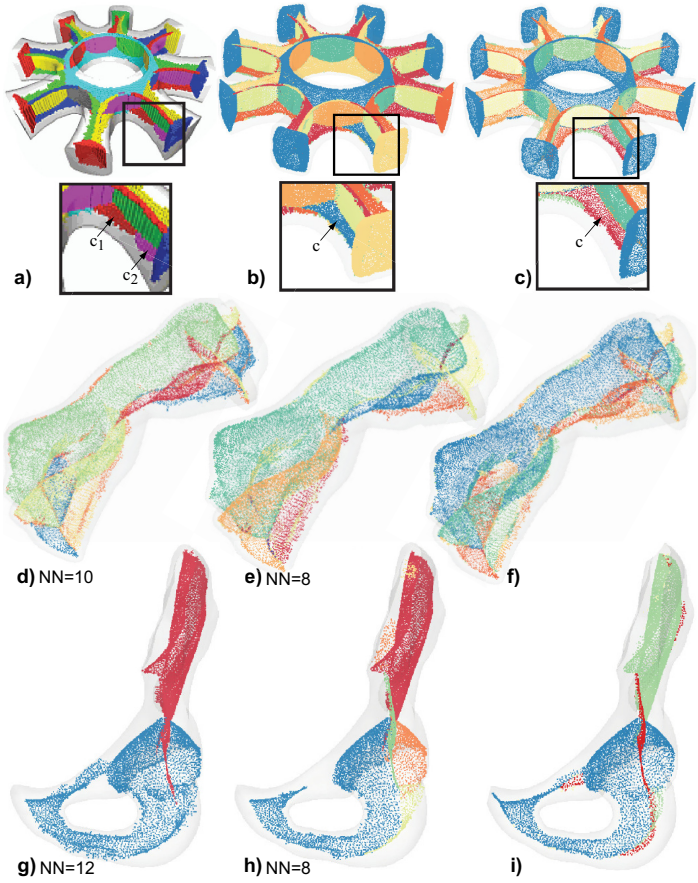

Fig. 6. Medial sheet extraction: a) Reniers and Telea (2008c); b,d,e,g,h) Kustra et al. (2014); c,f,i) our method. For Kustra et al. (2014), its key parameter, the number of nearest neighbors *NN*, is indicated.

3.2.2. *Y-intersection curve extraction*

Once the medial sheets γ_i are found, the Y-intersection curves can be found as those points $\mathbf{x} \in S_\Omega$ that belong to at least two different sheets. However, performing this test directly on the medial sheet-set is not possible, since our sheets are disjoint, *i.e.* $\gamma_i \cap \gamma_{j\neq i} = \varnothing$. Hence, we find Y-curve points as those skeleton points $\mathbf{x}$ which have at least one k-nearest neighbor belonging to a different sheet than the one containing $\mathbf{x}$. Tuning k allows controlling the thickness of the Y network being computed. Figure 7 shows three examples of Y networks, computed for $k = 3$.

3.2.3. *Computing compact medial sheets*

In Sec. 3.2.1, we computed medial sheets as unstructured point clouds. Many shape processing operations require *compact* sheets, *e.g.* in the form of a triangle mesh. We show next how such meshes can be created based on an analysis of the feature vectors $\mathbf{v}_1(\mathbf{x})$ and $\mathbf{v}_2(\mathbf{x})$ of each skeleton point $\mathbf{x}$ (Sec. 2.1). The key idea is to use the feature vectors to back-project the connectivity information captured by the $\partial\Omega$

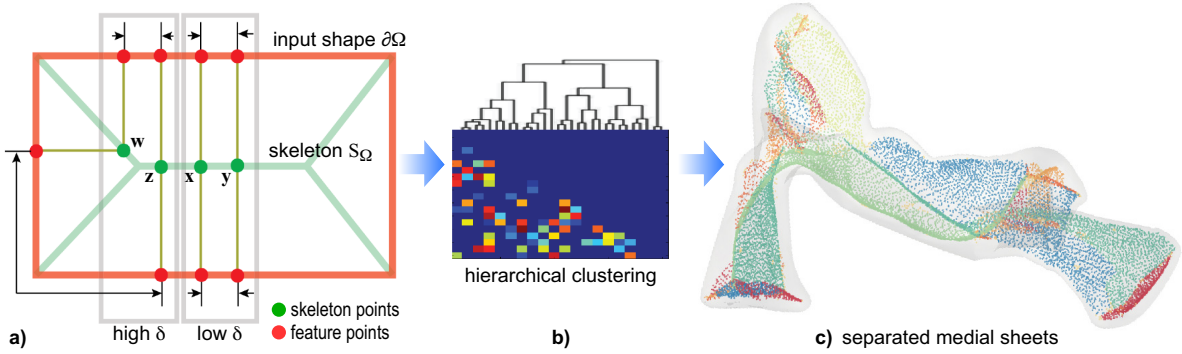

Fig. 5. Medial sheet computation: a) Distance function δ, illustrated in 2D. b) Sparse distance matrix, used as an input for hierarchical point clustering. c) Medial sheets found for a palatine bone shape (see Sec. 3.2.1).

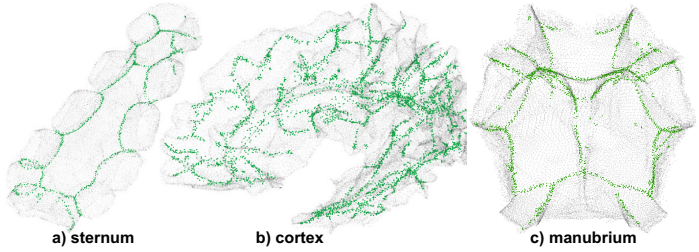

Fig. 7. Y-network extraction with Y-curve points colored green.

mesh onto each sheet γ. The method has two steps, as follows.

Feature vector alignment: The projection

$$P(\gamma) = P_1(\gamma) \cup P_2(\gamma) = \{\mathbf{x} \in \partial\Omega \mid \exists \mathbf{y} \in \gamma, \mathbf{x} \in \{\mathbf{f}_1(\mathbf{y}), \mathbf{f}_2(\mathbf{y})\}\} \quad (9)$$

of a sheet γ consists of two compact areas $P_1(\gamma)$ and $P_2(\gamma)$ of $\partial\Omega$, one for each side of γ. If we can isolate any of these two areas, we can next simply transfer its connectivity information onto γ to obtain our desired sheet mesh. For this, we reorder, or align, the feature vectors $\mathbf{v}_1(\mathbf{x})$ and $\mathbf{v}_2(\mathbf{x})$ of all sheet points so that all $\mathbf{f}_1$ are included in $P_1(\gamma)$, and all $\mathbf{f}_2$ are included in $P_2(\gamma)$, as follows. First, we select an arbitrary reference point $\mathbf{x}_{ref} \in \gamma$ and mark it as visited. We next visit all other points $\mathbf{x} \in \gamma$ in order of increasing distance to $\mathbf{x}_{ref}$ and redefine their feature points as

$$\mathbf{f}_i = \underset{\mathbf{f} \in \{\mathbf{f}_1, \mathbf{f}_2\}}{\arg\min}\ MGF(\mathbf{f}, \mathbf{f}_i^{vis}), \quad i \in \{1,2\}, \quad (10)$$

where $\mathbf{f}_i^{vis}$ is the closest visited (aligned) feature point to $\mathbf{f}_i$, and mark $\mathbf{f}$ as visited. When all points of γ are visited, all feature vectors $\mathbf{v}_1$ will be on the same side of γ as $\mathbf{v}_1(\mathbf{x}_{ref})$, while all $\mathbf{v}_2$ will be on the other side. We can next find the projection of side $i \in \{1,2\}$ of γ as $P_i(\gamma) = \{\mathbf{x} \in \partial\Omega \mid \exists \mathbf{y} \in \gamma, \mathbf{x} = \mathbf{f}_i(\mathbf{y})\}$.

Connectivity projection: We finally construct a meshed version of γ by simply copying all triangle information from $P_i(\gamma)$ to γ, with i being either 1 or 2 (both sides are equally good). That is, for any triangle $t = \{\mathbf{x}^i\}_{1 \le i \le 3}$ in *e.g.* $P_1(\gamma)$, we construct a triangle $t_\gamma = \{\mathbf{y}^i\}_{1 \le i \le 3}$ where $\mathbf{x}^i = \mathbf{f}_1(\mathbf{y}^i)$. Figure 8 illustrates the resulting meshed sheets for the surface skeletons of several complex anatomical shapes from the open database in Mitsuhashi et al. (2009), where neighbor sheets have different colors for illustration purposes. Given these meshed sheets, we can now use any polygon-based geometric algorithm to analyze or process them further, *e.g.*, to estimate curvature, areas, elongation, or compute shortest paths or distance fields.

4. Applications

We next use our computed medial features (point classification, regularization, and medial surface decomposition into sheets) to support several shape analysis applications. These examples implicitly illustrate the quality and robustness of our feature computation methods. Secondly, they show how such features enhance the added-value of surface skeletons by allowing it to support the construction of surface processing tools.

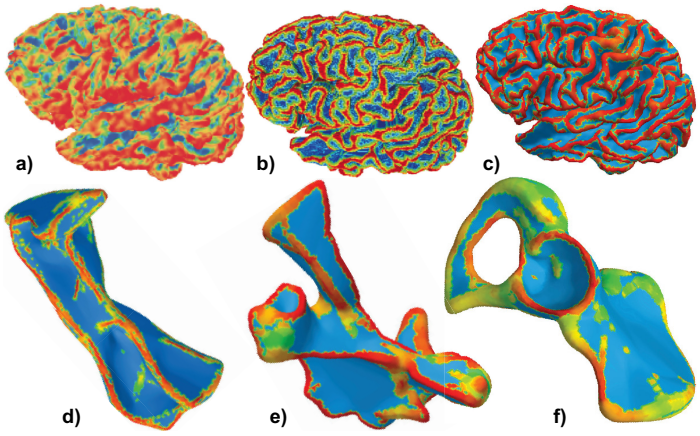

Fig. 9. Soft edge detection using a) curvature estimation (Taubin, 1995); b) skeleton method of Reniers et al. (2008a); c-f) our method.

4.1. Surface edge detection

Finding edges on a 3D surface has many applications in segmentation and classification. Most existing edge detectors are based on the surface's curvature tensor (Moreton and Séquin, 1992; Taubin, 1995; Clarenz et al., 2004). A problem of such detectors is that they operate at a given scale, *i.e.* find edges of a sharpness range which must be specified. Using skeletons allows finding both sharp and blunt edges, *i.e.*, removes the need to specify an edge-sharpness range: Following the observation that medial surface boundaries (A_3 points) correspond to curvature maxima or edges on the input surface (Pizer et al., 2003), Reniers *et al.* compute surface edges by finding A_3 points as explained in Sec. 3.1.2, and next back-projecting these on the input surface by the feature transform (Reniers et al., 2008a). We propose here an alternative approach: For each A_3 skeletal point $\mathbf{x}$, detected as explained in Sec. 3.1.2, we find all surface points enclosed in a sphere of radius $DT_\mathbf{x} + \tau$, with τ set as explained in Sec. 3.1.1, and assign to each surface point the smallest $DT_\mathbf{x}$ value which encloses it. Remaining surface points

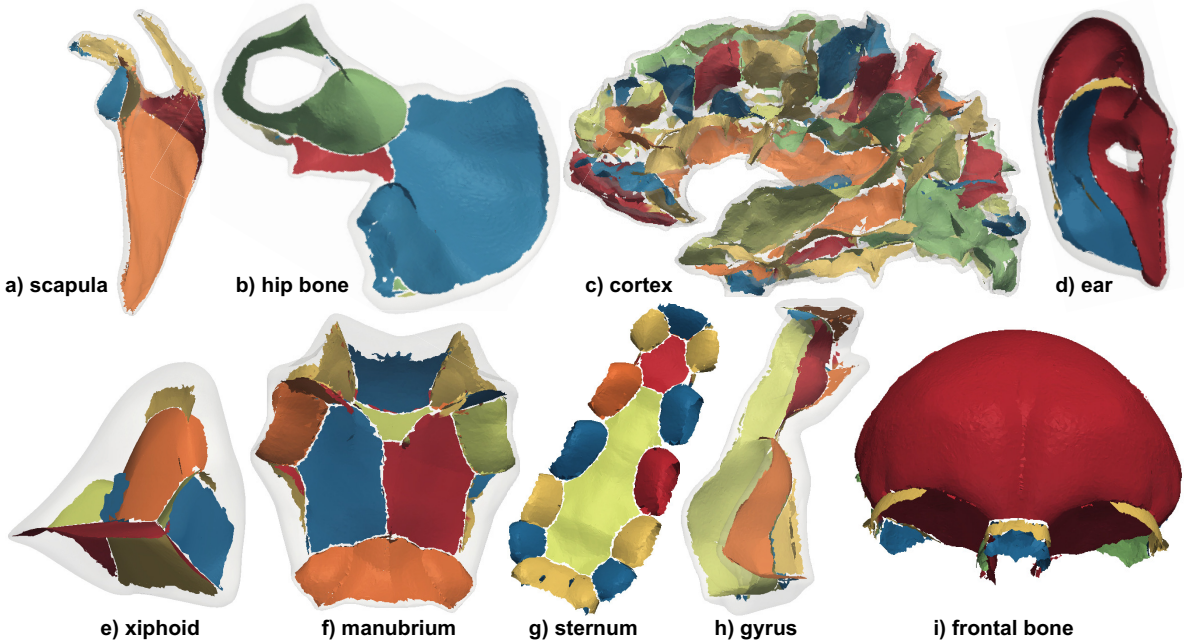

Fig. 8. Compact medial sheets computed for several anatomical models (Sec. 3.2.3).

are assigned a value of $\max(DT_{\mathbf{x}})$. Figure 9 (a-c) compares our method with the classical curvature detector of Taubin (1995) and with Reniers et al. (2008a) for a brain cortex surface. The goal is to find the sulcal brain structures, which correspond to (soft) convex surface edges, an important task in many structural and functional anatomic brain analyses. The presence of sulci is shown using a blue (concave) to red (sharp convex) rainbow colormap, mapping the three studied detectors: mean curvature (Taubin, 1995) (Fig. 9 a), geodesic distance to back-projected A_3 points (Reniers et al., 2008a) (Fig. 9 b), and our sphere-radius metric (Fig. 9 c). Our method achieves a sharper sulci separation than Reniers et al. (2008a), which in turn performs better than Taubin (1995). Figure 9 d-f show our method applied to three additional shapes which exhibit a mix of sharp and blunt edges. As visible, our detector finds both sharp (and thus, thin) and blunt (and thus, thick) edges. The edge sharpness and thickness is also visible in the color mapping.

4.2. Patch-based segmentation

Patch-based segmentation (PBS) divides a shape $\partial\Omega$ into *patches*, *i.e.* quasi-flat areas which are separated by sharp creases. Most PBS methods work by clustering surface points using, as similarity metric, the surface curvature or similar quantities (Shamir, 2008). Since medial surfaces fully capture the surface information via the MAT (Sec. 2.1), these medial surfaces can be used for PBS. For this, Reniers and Telea (2008b) compute soft edges by using the feature transform of low-importance medial-surface points, and next use these thick edges to segment the shape. However, their method needs to handle a large number of special cases (and is thereby quite complicated), and only works for voxel shapes. We propose here a much simpler approach: We project all skeleton-boundary points $\mathbf{p}$ (type A_3) to $\partial\Omega$ via our extended feature transform F_τ, *i.e.* compute the set $E = \{\mathbf{x} \in F_\tau(\mathbf{p}) | \mathbf{p} \in S_\Omega \wedge type(\mathbf{p}) = A_3\} \subset \partial\Omega$. The set E consists of a thick version of the edges of $\partial\Omega$. Due to the conservativeness of F_τ (Sec. 3.1.1), E will contain connected edges, in contrast to *e.g.* a naive thresholding of the curvature of $\partial\Omega$ or other similar local surface classifiers. Hence, we next find patches by simply computing connected components of $\partial\Omega \setminus E$. Finally, we add the points in E to their closest patch, thereby making the resulting patches become a partition of $\partial\Omega$.

Figure 10 shows our results, using the same color scheme as Fig. 8. For models with clear, sharp, edges, we see how patches neatly follow these edges (*e.g.* Fig. 10 a, rib sockets in Fig. 10 g, skull concavity in Fig. 10 i). More importantly, our method handles equally well models with soft edges (Figs. 10 b,c,f) and/or mixes of sharp and soft edges (Figs. 10 d,g,h).

4.3. Medial sheet mapping segmentation

In contrast to patch-based segmentation (Sec. 4.2), part-based segmentation (pBS) separates a shape $\partial\Omega$ into components that are perceived as being the natural 'parts' of the shape (Shamir, 2008). Among the many methods for pBS, *curve* skeletons are often used, as they readily capture the part-whole topology of shapes having elongated protrusions. One way to compute a pBS is to find the so-called junction points of curve skeletons (equivalent to Y-curves for surface skeletons), and then cut the shape with curves that go around these points (Reniers and Telea, 2008a). Such methods are robust and relatively simple to implement, but work well only for shapes with a tubular structure, *i.e.*, which have a meaningful curve skeleton. We propose here to use the surface skeleton for pBS. For this, we compute its medial sheets γ (Sec. 3.2.1), and next project these into $\partial\Omega$ using $P(\gamma)$ (Eqn. 9). Since all points on $\partial\Omega$ have a skeleton point by construction (Jalba et al., 2013), the entire shape is covered by such projections, which give us the 'parts' of the shape. The borders separating two such neighbor parts are nothing but the projections of the Y-curves. Since such curves are smooth (Siddiqi and Pizer, 2009), and the feature-vector field used for projection is also smooth (since parallel to $\nabla DT_{\partial\Omega}$ which is divergence-free away from the skeleton, see Siddiqi et al. (2002)), the resulting

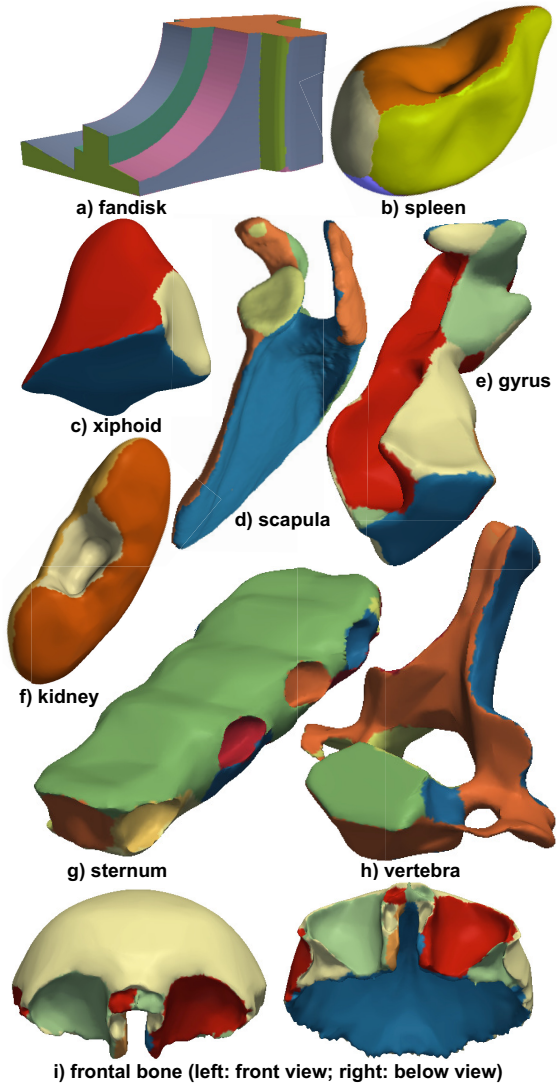

Fig. 10. Patch based segmentation (Sec. 4.2).

part-borders will also be smooth. Figure 11 show several part-based segmentation examples. Although many alternative pBS segmentations are possible, we argue that the found segments match well the perceived distinct shape parts. We note that such segmentations cannot be achieved using only curve-skeletons, since the shown shapes do not have a tubular structure.

5. Discussion

We discuss next several aspects related to our contribution – showing that we can efficiently and easily compute high-level medial features from large point-cloud skeletons, and that using such features in various applications is a practical proposition.

Generality: We require as input only a raw medial 3D cloud with two feature points per skeleton point. Such clouds can be very efficiently computed by recent GPU methods (Ma et al., 2012; Jalba et al., 2013) or older CPU methods (Hesselink and Roerdink, 2008), for any type of 3D shape topology or geometry.

Point classification: To our knowledge, our work is the first attempt to compute Giblin's medial point classification (Giblin and Kimia, 2004) for raw medial clouds. Using this classification to regularize skeletons (Sec. 3.1.3) is much simpler

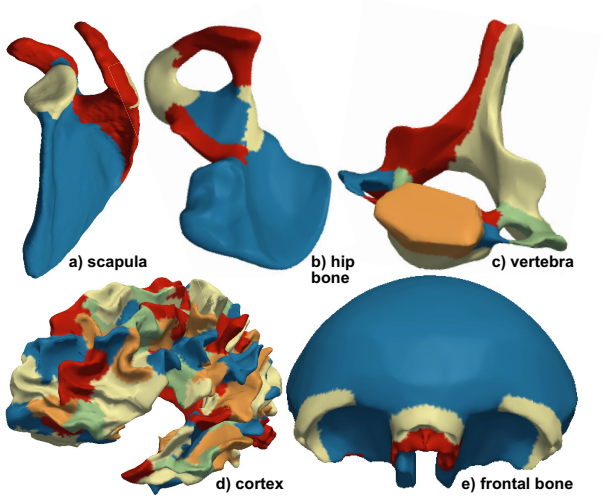

Fig. 11. Medial sheet mapping segmentation (Sec. 4.3).

to implement, and also much faster, than the MGF metric. Compared to local regularization metrics (Siddiqi et al., 2002; Pizer et al., 2003; Sud et al., 2005), we do not disconnect the skeleton, and only remove a thin layer of boundary points. This gives a simple, fast, automatic, and effective way to create clean medial surfaces that preserve relevant skeletal details.

Medial sheet extraction: Separating sheets from a raw medial cloud is a hard task, for which few methods exist, and for which generic point-cloud clustering tools cannot be used (Chang et al., 2009; Kustra et al., 2014). Our contribution – the similarity metric (Eqn. 7) – combines both local and global shape information, and enables a medial cloud segmentation into sheets which is noise-resistant and has a simple parameter setting. In detail, Kustra et al. (2014) requires tuning three parameters: The number of nearest neighbors of each skeletal point, the maximal allowed local-flatness of each sheet, and the sheet similarity. In contrast, our method requires a single parameter, the maximum MGF distance between feature-points of two skeleton-points that are on the same sheet (τ in Eqn. 8). For all tested shapes, a value of τ equal to four times the local point-density $\rho_{\partial\Omega}$ on the input surface yielded optimal results such as shown in this paper.

Y-network extraction: Our Y-network extraction finds the points *around* the Y-network of the skeleton. Exact Y-network points are, by definition, not explicitly found by the core skeletonization method we build atop, since this method always assumes two contact points for each medial point (Eqn. 2).

Scalability: On an Intel Core i7 3.8 GHz computer, our method, implemented in single-threaded C++, performs all described steps (medial feature computation, edge detection, PBS and pBS) in under 3 seconds for all shapes in this paper, which range between 30K and 230K skeleton points. Memory used is linear with the input point count. Medial sheet extraction is more costly, as it uses the expensive MGF metric (Eqn. 7). Timings for this step are given in Tab. 1, for CPU-based MGF computation. If desired, higher performance can be easily obtained by GPU-based MGF computation (Jalba et al. (2013))

Limitations: The quality of our medial features highly depends on the quality of the input medial cloud. This depends next on the sampling density of the input shape, since we require only

Table 1. Medial sheet extraction times on the CPU.

Model	Points $\|\partial\Omega\|$	Sheet extraction (sec.)
Hip bone	24852	46.87
Xiphoid	64690	68.88
Vertebra	48924	49.09
Manubrium	65038	64.22
Spleen	29507	55.35
Sternum	27525	62.26
Scapula	233856	436.57
Ear	24900	49.58
Cortex	58490	133.98
Gyrus	29116	56.75
Frontal bone	48171	131.54

two feature points per medial point (Sec. 3).

Applications: For the segmentation and classification applications in Sec. 4, we note that better specific techniques (not using medial descriptors) exist. Our sample applications are aimed at showing the *possibilities* that refined medial features open, as alternatives and in contrast to established approaches, and not as a definitive solution to the underlying use-cases.

6. Conclusions

We have presented a set of techniques for computing refined medial features from raw medial-surface point clouds. These features include medial point classification, skeleton regularization, Y-network extraction, separating medial sheets, and reconstructing meshed sheets. Such features enrich the level on which one can reason about medial surfaces, and open new ways for shape processing applications using medial clouds. We provide, for illustration, sample applications for edge detection and shape segmentation. Overall, our work shows that the more complex (and information-richer) surface skeletons can be, technically, used with the same ease and computational efficiency as the simpler, and so far more frequently used, curve skeletons.

References

Amenta, N., Choi, S., Kolluri, R., 2001. The power crust, in: Proc. SMA, ACM. pp. 65–73.

Arcelli, C., Sanniti di Baja, G., Serino, L., 2011. Distance-driven skeletonization in voxel images. IEEE TPAMI 33, 709–720.

Bai, X., Latecki, L., Liu, W.Y., 2007. Skeleton pruning by contour partitioning with discrete curve evolution. IEEE TPAMI 3, 449–462.

Blum, H., 1967. A transformation for extracting new descriptors of shape, in: Models for the Perception of Speech and Visual Form. MIT Press, pp. 362–380.

Bouix, S., Siddiqi, K., Tannenbaum, A., 2005. Flux driven automatic centerline extraction. Med. Imag. Anal. 9, 209–221.

Cao, T., Tang, K., Mohamed, A., Tan, T., 2010. Parallel banding algorithm to compute exact distance transform with the GPU, in: Proc. SIGGRAPH I3D Symp., pp. 134–141.

Chang, M., Leymarie, F., Kimia, B., 2009. Surface reconstruction from point clouds by transforming the medial scaffold. CVIU , 1130–1146.

Clarenz, U., Rumpf, M., Telea, A., 2004. Robust feature detection and local classification of surfaces based on moment analysis. IEEE TVCG 10, 516–524.

Cornea, N., Silver, D., Min, P., 2007. Curve-skeleton properties, applications, and algorithms. IEEE TVCG 13, 87–95.

Damon, J., 2006. Global medial structure of regions in $\mathbb{R}^3$. Geom. Topol. 10, 2385–2429.

Dey, T., Sun, J., 2006. Defining and computing curve skeletons with medial geodesic functions, in: Proc. SGP, IEEE. pp. 143–152.

Dey, T., Zhao, W., 2003. Approximating the medial axis from the Voronoi diagram with a convergence guarantee. Algorithmica 38, 179–200.

Falcão, A.X., Stolfi, J., Lotufo, R.A., 2004. The image foresting transform: Theory, algorithms, and applications. IEEE TPAMI 26, 19–29.

Foskey, M., Lin, M., Manocha, D., 2003. Efficient computation of a simplified medial axis, in: Proc. IEEE SMA, pp. 135–142.

Giblin, P., Kimia, B., 2004. A formal classification of 3D medial axis points and their local geometry. IEEE TPAMI 26, 238–251.

Giesen, J., Miklos, B., Pauly, M., Wormser, C., 2009. The scale axis transform, in: Proc. Annual Symp. Comp. Geom., pp. 106–115.

Hesselink, W., Roerdink, J., 2008. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. IEEE TPAMI 30, 2204–2217.

Jain, A.K., Murty, M.N., 1999. Data clustering: A review. ACM Computing Surveys 31, 264–323.

Jalba, A., Kustra, J., Telea, A., 2013. Surface and curve skeletonization of large 3D models on the GPU. IEEE TPAMI 35, 1495–1508.

Kimmel, R., Shaked, D., Kiryati, N., Bruckstein, A., 1995. Skeletonization via Distance Maps and Level Sets. CVIU 62, 382–391.

Kustra, J., Jalba, A., Telea, A., 2014. Robust segmentation of multiple intersecting manifolds from unoriented noisy point clouds. CGF 33, 73–87.

Leymarie, F., Kimia, B., 2007. The medial scaffold of 3D unorganized point clouds. IEEE TVCG 29, 313–330.

Leymarie, F., Levine, M., 1992. Simulating the grassfire transform using an active contour model. IEEE TPAMI 14, 56–75.

Ma, J., Bae, S., Choi, S., 2012. 3D medial axis point approximation using nearest neighbors and the normal field. Vis. Comput. 28, 7–19.

Miklos, B., Giesen, J., Pauly, M., 2010. Discrete scale axis representations for 3D geometry, in: Proc. ACM SIGGRAPH, pp. 394–493.

Mitsuhashi, N., Fujieda, K., Tamura, T., Kawamoto, S., Takagi, T., Okubo, K., 2009. BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Research 37, 782–785.

Moreton, H., Séquin, C., 1992. Functional optimization for fair surface design, in: Proc. ACM SIGGRAPH, pp. 167–176.

Ogniewicz, R.L., Kubler, O., 1995. Hierarchic Voronoi skeletons. Pattern Recognition 28, 343–359.

Palagyi, K., Kuba, A., 1999. Directional 3D thinning using 8 subiterations, in: Proc. DGCI, Springer. pp. 325–336.

Pizer, S., Siddiqi, K., Szekely, G., Damon, J., Zucker, S., 2003. Multiscale medial loci and their properties. IJCV 55, 155–179.

Prohaska, S., Hege, H.C., 2002. Fast visualization of plane-like structures in voxel data, in: Proc. IEEE Visualization, p. 2936.

Pudney, C., 1998. Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images. CVIU 72, 404–413.

Reniers, D., Jalba, A., Telea, A., 2008a. Robust classification and analysis of anatomical surfaces using 3D skeletons, in: Proc. VCBM, pp. 61–68.

Reniers, D., Telea, A., 2007. Tolerance-based feature transforms, in: Advances in Comp. Graphics and Comp. Vision, Springer. pp. 187–200.

Reniers, D., Telea, A., 2008a. Part-type segmentation of articulated voxel-shapes using the junction rule. CGF 27, 1837–1844.

Reniers, D., Telea, A., 2008b. Patch-type segmentation of voxel shapes using simplified surface skeletons. CGF 27, 1954–1962.

Reniers, D., Telea, A., 2008c. Segmenting simplified surface skeletons, in: Proc. DGCI, Springer. pp. 132–145.

Reniers, D., van Wijk, J.J., Telea, A., 2008b. Computing multiscale skeletons of genus 0 objects using a global importance measure. IEEE TVCG 14, 355–368.

Rossi, L., Torsello, A., 2012. An adaptive hierarchical approach to the extraction of high resolution medial surfaces, in: Proc. 3DIMPVT, pp. 371–378.

Rumpf, M., Telea, A., 2002. A continuous skeletonization method based on level sets, in: Proc. VisSym, pp. 151–158.

Shaked, D., Bruckstein, A., 1998. Pruning medial axes. CVIU 69, 156–169.

Shamir, A., 2008. A survey on mesh segmentation techniques. CGF 27, 1539–1556.

Siddiqi, K., Bouix, S., Tannenbaum, A., Zucker, S., 2002. Hamilton-Jacobi skeletons. IJCV 48, 215–231.

Siddiqi, K., Pizer, S., 2009. Medial Representations: Mathematics, Algorithms and Applications. Springer.

Sobiecki, A., Jalba, A., Telea, A., 2014. Comparison of curve and surface skeletonization methods for voxel shapes. Patt. Recog. Lett. 47, 147–156.

Stolpner, S., Whitesides, S., Siddiqi, K., 2009. Sampled medial loci and boundary differential geometry, in: Proc. IEEE 3DIM, pp. 87–95.

Stolpner, S., Whitesides, S., Siddiqi, K., 2011. Sampled medial loci for 3D shape representation. CVIU 115, 695–706.

Strzodka, R., Telea, A., 2004. Generalized distance transforms and skeletons in graphics hardware, in: Proc. VisSym, pp. 221–230.

Sud, A., Foskey, M., Manocha, D., 2005. Homotopy-preserving medial axis simplification, in: Proc. SPM, pp. 103–110.

Taubin, G., 1995. Estimating the tensor of curvature of a surface from a polyhedral approximation, in: Proc. ICCV, pp. 902–907.

Telea, A., Jalba, A., 2012. Computing curve skeletons from medial surfaces of 3D shapes, in: Proc. TPCG, EG Press. pp. 273–280.

Telea, A., van Wijk, J.J., 2002. An augmented fast marching method for computing skeletons and centerlines, in: Proc. VisSym, pp. 251–259.

Wan, M., Dachille, F., Kaufman, A., 2001. Distance-field based skeletons for virtual navigation, in: Proc. IEEE Visualization, pp. 239–246.