6.1 INTRODUCTION

In the previous chapters, we have presented several methods to extract and regularize medial point clouds, and use such skeletal point clouds to construct the first steps towards more advanced shape processing methods. Specifically, Chapter 4 presented a point cloud processing method capable of identifying manifold structures embedded in noisy point clouds. This enabled us to extract noise-free and separate manifolds from 3D surface skeleton clouds. Chapter 5 extended the idea of refining medial descriptors in the direction of analyzing their density properties to enable shape segmentation applications.

However, the above medial properties (manifold sheets and local density characteristics) are not the only (useful) properties one can extract, and reason about, for surface skeletons. Specifically, surface skeletons have a complex structure, consisting of several so-called medial sheets, or manifolds. Manifold points have several types, which characterize the kind of surface points they correspond to via the MAT [58]. Boundaries of these curves, mapped via the MAT to shape edges [134, 154], can be used for shape segmentation [130]. The curves where sheets meet, also called Y-intersection curves or medial scaffold, can be used for shape re-

This chapter is based on the following papers:

construction and shape matching [23, 34, 93]. The individual sheets correspond to separate shape parts, enabling additional shape simplification and segmentation applications.

Although the usage of such features, for both curve and surface skeletons has been demonstrated and applied on voxel-based representations [130, 135], their computation for point-cloud or meshed medial surfaces (such as introduced in Chapter 3) is far from trivial. Extracting features such as endpoints, branches, and junctions from curve skeletons is much easier than for surface skeletons, due to the inherent simpler structure of curve skeletons. Among other aspects, such as a higher computational simplicity and robustness, this makes curve skeletons to be more frequently used in applications, even though they encode less information than surface skeletons. As such, to become useful and effective for a wide range of real-world shape processing applications, recent fast-and-accurate 3D medial surface point-cloud extraction methods, such as our own method presented in Chapter 3, need enhancement in the sense of robustly 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 chapter, 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 turns 3D medial clouds into shape representations which can be directly used for several shape analysis applications.

The structure of this chapter is as follows. Section 6.1.1 reviews the challenges associated with medial surfaces as described in Chapter 3 and motivates the remainder of the chapter. Section 6.2 presents our methods to compute these features from unstructured medial clouds. Section 6.3 shows the use of our medial features for part-based and patch-based shape segmentation and classification, to illustrate the applicability of our feature computation. Section 6.4 discusses our techniques. Section 6.5 concludes the chapter.

6.1.1 Motivation

The feature extraction or decomposition of medial surface features is an essential step towards its further usage in applications for shape analysis or classification. Examples of such features are the individual medial sheets [154, 165], Y-intersection curves or sheet boundaries, i.e., the medial scaffold [93]. Applications of such features has been previously demonstrated on finely-sampled voxel skeletons [132], leading to the usage of medial surfaces to create compelling multiscale shape segmentations [130]. Doing all above for medial clouds is, however, far from trivial. In previous chapters, this analysis has been previously attempted by using local medial geometry properties (Chapters 4 and 5). As shown in
Chapter 4, using generic point-cloud segmentation methods for 3D surface skeletons is doable, but quite challenging, since medial surfaces consist of numerous intersecting manifolds with boundaries, which are hard to capture even with very dense point clouds. In addition, the method presented in Chapter 4 requires parameter settings related to local medial geometrical properties, such as the maximum local connectivity angle. These parameters are specific to the point cloud properties, which only become available once the skeleton has been computed. Therefore, the automation or abstraction of shape processing pipelines becomes a non-trivial task for a high variability of input shapes.

Overall, the above observations make us believe that additional higher-level information computed for surface skeletons, that show how these skeletons relate to their input shapes, is useful and necessary for shape analysis applications. Indeed, on the one hand, having such information, beyond simple local connectivity or point-sampling density data, should enable us to reason about shape properties on a higher level. On the other hand, such higher level information may remove the need of setting various parameters and thresholds for the analysis of lower-level skeletal information – the desired information is readily available, and we do not need to ‘reverse engineer’ it from the actual skeleton geometry or sampling density.

In this chapter, we present several methods that enrich a 3D medial point cloud with several refined abstractions, or features. Rather than using the local medial cloud geometric properties, the relationships between the medial points \( x_i \) feature vectors \( f_1, f_2 \) and their correspondence to surface \( \Omega \) properties are used. Figure 6.1 overviews our proposal. Given a meshed surface, we first compute its ‘raw’ skeleton point cloud. Next, we classify medial points following [58]. This classification further enables us to extract separate medial point-sets corresponding to medial sheets (and subsequently detect the skeletal Y-network). Separately, all above features (medial point types and sheets) enable us to support several practical applications: simple and efficient medial point-cloud regularization, medial sheet reconstruction, and part-based and patch-based shape segmentation. Computing and using these refined skeletal features is discussed next.

6.2 Computing Refined Medial Features

To address the interpretation challenges outlined in Sec. 6.1.1, 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 [58], find skeletal boundaries and Y-curves, and robustly regularize the medial surface (Sec. 6.2.1). We next use these features to robustly segment the medial surface into separate manifolds (Sec. 6.2.2). As input for all these operations, we only assume a surface skeleton represented by an unstructured and unoriented
Figure 6.1: Refined skeletal features computed from medial point clouds (top row) and subsequently enabled applications (bottom row).

point-cloud having exactly one skeleton point per surface point and exactly two feature-points per skeleton point, as computed e.g. by our skeletonization method presented in Chapter 3.

6.2.1 Medial points classification

6.2.1.1 Estimating the feature transform

To classify unstructured medial clouds following [58], we first need to estimate the feature transform $FT(x \in S_{\partial\Omega})$ (Eqn. 2.3). As explained earlier, the entire $FT$ is not directly available in most skeletonization methods; in particular, our point-cloud methods (Chapter 3) only compute two feature points per skeleton point (Section 3.5.1). To find all feature points, we proceed as follows. Let $DT_x$ be next a shorthand for the input shape’s distance transform $DT_{\partial\Omega}(x)$. For each skeleton point $x \in S_{\partial\Omega}$, we first find the closest points $FT_\tau(x) \subset \partial\Omega$ in a radius $DT_x + \tau$, where $\tau$ is defined as a fraction of $\varepsilon \rho_{\partial\Omega}(x)$, where $\rho_{\partial\Omega}(x)$ is the average point density on $\partial\Omega$ in a small neighborhood around $\{f_1(x)\} \cup \{f_2(x)\}$, and $\varepsilon$ is a small constant set to 0.1. The slightly increased radius determines that the set $F_\tau(x)$ will conservatively contain all feature points of $x$, i.e. $FT(x) \subset FT_\tau(x)$. Setting $\tau$ to track the local sampling density of $\partial\Omega$ allows us to conservatively estimate $F_\tau$ for non-uniformly sampled meshes without introducing too many false-positives, i.e., minimizing the set $FT_\tau \setminus FT$.

Given the finite tolerance $\tau$ and the discrete sampling of $\partial\Omega$, $FT_\tau(x)$ will also contain surface points which are slightly further from $x$ than feature points; this is especially salient for points $x$ 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 $FT(x)$ given by $FT_\tau(x)$ does not pose any problems to our medial attribute computation.
6.2.1.2 Classification of medial points

Since $FT_\tau(x)$ is essentially a dilated, or fuzzy, version of $FT(x)$, it consists of one or several point clusters centered around actual feature points. A cluster $C \subset FT_\tau(x)$ can be defined as

$$C = \{ f \in FT_\tau(x) | \forall y \in C, z \in (FT_\tau(x) \setminus C), \|f - y\| < \|f - z\| \} \ (6.1)$$

i.e., all points which are closer to each other than to any point from another cluster.

We observed that the number of these clusters is a good indicator of the type of the medial point $x$: For $A_3$ points, there is one such cluster, whose diameter is proportional to $DT_x$; for $A_2^1$ points, we find two clusters; and for $A_k^1$ points, we find $k$ clusters. To compute $k$, we cluster the point-set $FT_\tau(x)$ 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 $p_{\partial\Omega}$. This results in $k$ clusters. The value of $k$ gives us the point type, as explained above.

![Figure 6.2: Skeleton point classification based on fuzzy $FT_\tau$ analysis. The figure is drawn for the case of 2D skeletons, for illustration simplicity.](image-url)

Let us justify why $k$ is a good point-type indicator. Figure 6.2 a shows an incorrect classification of medial point $p$ which is on the skeleton $S_\Omega$ branch ended by point $e$. Since the intersection of $\partial\Omega$ with a ball $B_p$ of radius $DT_p + \tau$ and center $p$ (dotted red circle) yields a single cluster (thick red line), $p$ is incorrectly marked as $A_3$ rather than as $A_2^1$. This is caused by (1) the value $\tau$ used to compute $FT_\tau$ being too large; (2) $p$ being close to $e$; and (3) the bump on $\partial\Omega$ corresponding to $e$ being too shallow. Consider now the minimal distance $d_{min}$ from $p$ that we have to move on $S_\Omega$ away from $e$ to find a point $q$ which is correctly classified as $A_2^1$ (Figure 6.2 b). This happens when the intersection of the ball $B_q$ of radius $DT_q + \tau$ and center $q$ (dotted blue circle) yields two
disconnected clusters on \( \partial \Omega \) (marked thick blue). To find \( d_{\text{min}} \), note that the maximal ‘inward shift’ between the upper parts of \( B_p \) and \( B_q \) equals 
\[
\sigma = DT_p - DT_q + d_{\text{min}}. 
\]
To cause the disconnection of the compact cluster in Figure 6.2a, \( \sigma \) must be larger than the maximal bump height on \( \partial \Omega \) that fits in the sphere-shell of thickness \( \tau \), i.e., \( DT_p - DT_q + d_{\text{min}} > \tau \). Since \( DT_q - DT_p = d_{\text{min}} \cos \alpha \), where \( \alpha \) is the angle between a feature vector and the tangent plane to \( S_\Omega \), it follows that
\[
d_{\text{min}} > \frac{\tau}{1 - \cos \alpha}. \tag{6.2} 
\]
Separately, for a \( \partial \Omega \) with local sampling density \( \rho_{\partial \Omega} \), the corresponding skeleton sampling density is
\[
\rho_S = \frac{\rho}{\sin \alpha}. \tag{6.3} 
\]
Incorrect classification (Figure 6.2a) can only occur when \( \rho_S < d_{\text{min}} \). Indeed, a correct classification should mark only a one sampling-point-thin ‘band’ of skeleton points as \( A_3 \) (surface skeleton boundary). If \( \rho_S \) is smaller than the minimal ball-shift \( d_{\text{min}} \) required to change point type from \( A_3 \) to \( A_1^2 \), this band gets thicker, which blurs our classification. Substituting our value of \( \tau = \epsilon \rho_{\partial \Omega} \) (Sec. 6.2.1.1) in Eqn. 6.2, it follows that this problem can only appear when \( \epsilon \geq \frac{1 - \cos \alpha}{\sin \alpha} \). For our chosen value of \( \epsilon = 0.1 \), this implies \( \alpha \lesssim 11.4^\circ \). 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 sampling-point thick.

Following the above, if \( k = 1 \) or \( k = 2 \), we can confidently say that to have found an \( A_3 \), respectively \( A_1^2 \), 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 \( A_1^2 \) up to \( A_1^4 \) 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. 6.2.2.2, based on the segmentation of \( S_\Omega \) into individual medial sheets.

Related to our work, [134] found \( A_3 \) points by computing the set difference between the full medial surface \( S \) and a simplified version \( S_\tau \) of \( S \), where \( \tau \) is a small fixed value and simplification uses the MGF metric (Sec. 2.3.4). Compared to our approach, their method does not find \( A_1^{k,k>1} \) points, and does not give an analysis of how to set parameter values.

6.2.1.3 Skeleton regularization using \( A_3 \) edge filtering

As outlined in Sec. 2.3.4, the MGF metric [37, 135] provides very good regularization properties such as separating spurious skeleton points
from important ones while maintaining skeleton connectivity. The MGF importance $\rho(x)$ of a medial point $x$ equals the length of the longest shortest-geodesic on $\partial \Omega$ between any two feature points of $FT(x)$. Hence, the MGF requires an accurate computation of the feature transform $FT$ (Eqn. 2.3). As discussed in Sec. 6.2.1.1, we compute a conservative $FT_\tau$ 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, (Chapter 3) and [135] compute the MGF using only two feature points per skeleton point, i.e. implicitly consider all medial points to be of type $A_2^2$. This has two problems. First, the importance $\rho$ 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 (Chapter 3).

We propose here an alternative way to regularize medial surfaces by simply filtering $A_3$ points found by our classification. Figure 6.3 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 6.3 a shows the medial cloud with feature vectors (in red) for the $A_3$ points. Figure 6.3 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. 6.4 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 regular-

![Figure 6.3: Skeleton regularization. (a) Rounded spleen shape with feature vectors shown for $A_3$ points. (b) Skeleton regularized by filtering $A_3$ points.](image)
Refined Abstractions for Medial Point Clouds

6.2.2 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 [93, 130]. While such decompositions can be computed relatively easy for voxel skeletons [132], this is challenging for medial clouds, especially when these contain a large number of complex sheets (Chapter 4).

We address this task by clustering the medial cloud based on a novel definition of medial sheets that uses the medial cloud properties (Sec. 6.2.2.1). Next, we use this decomposition to robustly find Y-intersection curves where several such sheets meet (Sec. 6.2.2.2). Finally, we use the feature transform to construct compact (meshed) sheet representations (Sec. 6.2.2.3).

6.2.2.1 Medial sheet computation

We first define a distance for a pair of two medial points \(x\) and \(y\) as

\[
\delta(x, y) = \sum_{a \in F(x)} \min_{b \in F(y)} MGF(a, b), \tag{6.4}
\]

where \(MGF(a, b)\) is the medial geodesic function, i.e. the length of the shortest geodesic on \(\partial \Omega\) between feature points \(a\) and \(b\) [135]. Next, we define a medial sheet \(\gamma\) as all medial points having a distance \(\delta\) lower than a threshold \(\tau\)

\[
\gamma = \{x \in S_\Omega | \exists y \in \gamma, \delta(x, y) < \tau\}. \tag{6.5}
\]

The value of \(\tau\) is set to a relative low geodesic distance in the distance range between two points on \(\partial \Omega\). In practice, this resulted in setting it in the range \([3 \times \rho_{\partial \Omega}, 10 \times \rho_{\partial \Omega}]\), where \(\rho_{\partial \Omega}\) is the average surface \(\partial \Omega\) sampling density. The rationale behind Eqn. 6.5 is that medial points \(x\) and \(y\) 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 [122], 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 6.4 illustrates this for a 2D shape (for simplicity): Medial points \(x\) and \(y\) are on the same sheet, and have small MGF distances between their feature points, thus a small \(\delta(x, y)\). In contrast, medial points \(w\) and \(z\), which belong to different sheets, have at least two feature vectors pointing in different directions, thus a large \(\delta(x, y)\).
Equations 6.4 and 6.5 define medial sheets without using any medial connectivity information. We can thus use them to segment a medial cloud into its sheets, as follows. First, we define a distance matrix \( M \) encoding the distances \( \delta(x, y) \) between all medial point pairs. For efficiency, we only compute matrix entries that correspond to \( \delta \) values below our chosen threshold \( \tau \), since the sheet definition (Eqn. 6.5) only requires to know when \( \delta < \tau \). To do this, we first notice that medial points \( x \) and \( y \) which are far apart will also have large values of \( \delta(x, y) \). As such, for a point \( x \), we only consider in \( M \) those entries that correspond to its \( K \) nearest neighbours \( y \). Secondly, when computing \( \delta(x, y) \), if the length of the geodesic traced on \( \partial \Omega \) from \( x \) to \( y \) exceeds \( \tau \), 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. Finally, we use \( M \) as input for a single-linkage hierarchical clustering [71], which outputs a partition of \( S_\Omega \) into a set of medial sheets \( \gamma \), so that \( \gamma_i \cap \gamma_j = \emptyset \) and \( \bigcup_i \gamma_i = S_\Omega \). Figure 6.4 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. 6.2.2.3.

Figure 6.5 compares our method for sheet extraction from a medial cloud with two other methods. Image (a) shows the method of [132], 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 \( A^3 \) points (Sec. 2.2.2). Next, separate medial sheets are found as being the connected components of the voxel set \( S \setminus S_Y \). While this method gives good results, it is quite sensitive to the voxel sampling of the input shape. For instance, the cog wheel detail in Figure 6.5 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 halfway and (2) sheet detection is based on connected component finding.
Image (b) shows the method presented in Chapter 4. Image (c) shows our method. As visible, both local (Chapter 4) and our presented 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 our method presented earlier in Chapter 4 for two shapes and two different values of the nearest-neighbour count $NN$ (one of the parameters of the method in Chapter 4). Two observations can be made here. First, we see how the results depend quite strongly on the choice for $NN$. In contrast, the proposed method does not require tuning such a parameter. Secondly, and more importantly, the point-similarity used in Chapter 4 is essentially purely local, as it involves only inter-point distances and local flatness of the sheets. In contrast, the proposed method uses a distance function (Eqn. 6.4) which captures global shape properties, due to the underlying
MGF function. This makes the sheet computation far less sensitive to shape variations.

### 6.2.2.2 Y-intersection curve extraction

Once the medial sheets are found, the Y-intersection curves can be found as those points $x \in S_\Omega$ 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. $\gamma_i \cap \gamma_{j\neq i} = \emptyset$. Hence, we find Y-curve points as those skeleton points $x$ which have at least one $k$-nearest neighbour belonging to a different sheet than the one containing $x$. Tuning $k$ allows controlling the thickness of the Y network being computed. Figure 6.6 shows three examples of Y networks, computed for $k = 3$.

![Figure 6.6: Y-network extraction with Y-curve points colored green.](image)

### 6.2.2.3 Computing compact medial sheets

In Sec. 6.2.2.1, we computed medial sheets as unstructured point clouds. Many shape processing operations require the input to be a triangle mesh. We show next how such meshes can be created based on an analysis of the feature vectors $v_1(x)$ and $v_2(x)$ of each skeleton point $x$ (Chapter 3). The key idea is to use the feature vectors to back-project the connectivity information captured by the $\partial \Omega$ mesh onto each sheet $\gamma$. The method has two steps, as follows.

**Feature vector alignment:** The projection

$$P(\gamma) = P_1(\gamma) \cup P_2(\gamma) = \{x \in \partial \Omega \mid \exists y \in \gamma, x \in \{f_1(y), f_2(y)\}\} \quad (6.6)$$

of a sheet $\gamma$ consists of two triangulated areas $P_1(\gamma)$ and $P_2(\gamma)$ of $\partial \Omega$, one for each side of $\gamma$. If we can isolate any of these two areas, we can next simply transfer its connectivity information onto $\gamma$ to obtain our desired sheet mesh. For this, we reorder, or align, the feature vectors $v_1(x)$ and $v_2(x)$ of all sheet points so that all $f_1$ are included in $P_1(\gamma)$, and all $f_2$ are included in $P_2(\gamma)$, as follows. First, we select a reference point $x_{ref} \in \gamma$
and mark it as visited. We next visit all other points $x \in \gamma$ in order of increasing distance to $x_{ref}$ and redefine their feature points as

$$f_i = \arg\min_{f \in \{f_1, f_2\}} MGF(f, f_i^{vis}), \quad i \in \{1, 2\},$$

(6.7)

where $f_i^{vis}$ is the closest visited (aligned) feature point to $f_i$, and mark $f$ as visited. When all points of $\gamma$ are visited, all feature vectors $v_1$ will be on the same side of $\gamma$ as $v_1(x_{ref})$, while all $v_2$ will be on the other side. We can next find the projection of side $i \in \{1, 2\}$ of $\gamma$ as $P_i(\gamma) = \{x \in \partial \Omega \mid \exists y \in \gamma, x = f_i(y)\}$.

**Connectivity projection:** We finally construct a meshed version of $\gamma$ by simply copying all triangle information from $P_i(\gamma)$ to $\gamma$, with $i$ being either 1 or 2 (both sides are equally good). That is, for any triangle $t = \{x^i\}_{1 \leq i \leq 3}$ in e.g. $P_1(\gamma)$, we construct a triangle $t_\gamma = \{y^i\}_{1 \leq i \leq 3}$ where $x^i = f^i_1$. Figure 6.7 illustrates the resulting meshed sheets for the surface skeletons of several complex anatomical shapes from the open database in [110], where neighbour sheets have different colors for illustration purposes. Given these meshed sheets, we can now use any polygon-based geometric algorithm to analyse or process them further, e.g., to estimate curvature, areas, elongation, or compute shortest paths or distance fields.

![Figure 6.7: Compact medial sheets computed for several anatomical models (Sec. 6.2.2.3).](image)

### 6.3 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 qual-
Figure 6.8: Soft edge detection on surfaces using curvature estimation [176] (a); skeleton method of [134] (b); our method (c-f).

ity 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.

### 6.3.1 Surface edge detection

Finding edges on a surface in 3D space has many applications in segmentation and classification. Most existing edge detectors are based on the surface’s curvature tensor [26, 111, 176]. A challenge 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: Following the observation that medial surface boundaries ($A_3$ points) correspond to curvature maxima or edges on the input surface [122], Reniers et al. compute surface edges by finding $A_3$ points as explained in Sec. 6.2.1.2, and next back-projecting these on the input surface by the feature transform [134]. We propose here an alternative approach: For each $A_3$ skeletal point $x$, detected as explained in Sec. 6.2.1.2, we find all surface points enclosed in a sphere of radius $DT_x + \tau$, with $\tau$ set as explained in Sec. 6.2.1.1, and set each surface point with the smallest $DT_x$ value which enclosed it. Remaining surface points are set to $\max(DT_x)$. Figure 6.8 (a-c) compares our method with the classical curvature detector of [176] and with [134] 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 [176] (Figure 6.8 a), geodesic distance to back-projected $A_3$ points [134] (Figure 6.8 b), and our sphere
radius metric (Figure 6.8 c). Our method achieves a sharper sulci separation than [134], which in turn performs better than [176]. Images (c-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.

![Figure 6.9: Patch based segmentation (Sec. 6.3.2).](image)

6.3.2 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 [151]. Since medial surfaces fully capture the surface information via the MAT (Chapter. 3), these medial
surfaces can be used for PBS. For this, Reniers et al. 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 \( p \) (type \( A_3 \)) to \( \partial \Omega \) via our extended feature transform \( FT_\tau \), i.e. compute the set \( E = \{ x \in FT_\tau(p) | p \in S_{\partial \Omega} \land \text{type}(p) = A_3 \} \subset \partial \Omega \). The set \( E \) consists of a thick version of the edges of \( \partial \Omega \). Due to the conservativeness of \( F_\tau \) (Sec. 6.2.1.1), \( E \) 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 6.9 shows our results, using the same color scheme as Figure 6.7. For models with clear, sharp, edges, we see how patches neatly follow these edges (e.g. Figure 6.9 a, rib sockets in Figure 6.9 g, skull concavity in Figure 6.9 i). More importantly, our method handles equally well models with soft edges (Figs. 6.9 b,c,f) and/or mixes of sharp and soft edges (Figs. 6.9 d,g,h).

### 6.3.3 Medial sheet mapping segmentation

In contrast to patch-based segmentation (Sec. 6.3.2), part-based segmentation (pBS) separates a shape \( \partial \Omega \) into its meaningful components that are perceived as being the natural ‘parts’ of the shape [151]. Among the many methods for pBS, curve skeletons are often used, as they readily capture the part-in-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 [135]. 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 \( \gamma \) (Sec. 6.2.2.1), and next project these into \( \partial \Omega \) using \( P(\gamma) \) (Eqn. 6.6). Since all points on \( \partial \Omega \) have a skeleton point by construction (Chapter 3), the entire shape is covered by such projections, which give us the ‘parts’ of the shape. The borders separating two such neighbour parts are nothing but the projections of the Y-curves. Since such curves are smooth [154], and the feature-vector field used for the projection is also smooth (since parallel to \( \nabla DT_{\partial \Omega} \) which is divergence-free away from the skeleton [155]), the resulting part borders will also be smooth. Figure 6.10 show several part-based segmentation examples. Although many alternative pBS segmentations are possible, we argue that the identified segments coincide well with what one would regard to be the distinct shape parts. Notably, such segmentations cannot
be achieved using a curve-skeleton, since the shown shapes do not have a tubular structure.

Figure 6.10: Medial sheet mapping segmentation (Sec. 6.3.3).

6.4 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:** As input for all our methods, we require only a raw medial 3D cloud having two feature points per skeleton point. Such point clouds can be very efficiently and easily computed by recent GPU methods [73, 101] or older CPU methods [66], 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 [58] for unstructured medial clouds. In addition, using this classification for skeleton regularization (Sec. 6.2.1.3) is considerably simpler to implement, and also much faster, than the alternative MGF metric. Compared to local regularization metrics [122, 155, 168], we do not disconnect the skeleton, and guarantee to remove only a thin layer of boundary points. This gives a simple, fast, 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 challenging task for which few methods exist, and generic point-cloud clustering tools cannot be easily used (Chapter 4). Our contribution here is the similarity metric (Eqn. 6.4) which combines both local and global shape information. This enables a medial cloud segmentation into sheets which is noise-resistant and has a simple parameter setting. In detail, [88] requires tuning three parameters: The number of nearest neighbours 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 ($\tau$ in Eqn. 6.5). For all tested shapes, a value of $\tau$ equal to four times the local point-density $\rho_{\partial\Omega}$ on the input surface yielded optimal results.

**Y-network extraction:** Our Y-network extraction finds the points around the Y-network of the skeleton. Exact Y-network points are not (by definition) directly extracted by the core skeletonization method we use, since this method always assumes two contact points for each medial point (Eqn. 2.2).

**Scalability:** We implemented our medial feature computation methods in C++. On an Intel Core i7 3.8 GHz computer, our single-threaded code computes all medial features, except the medial sheets, in a few seconds for all shapes shown in this chapter, which range between 30K and 230K skeleton points. Medial sheet extraction is more costly, as it uses the expensive MGF metric (Eqn. 6.4). On the same platform, using the proposed CPU-based MGF computation (Chapter 3), sheet extraction takes under one minute for all tested shapes. Higher performance can be easily obtained, if desired, e.g. using the GPU-based MGF computation from [73].

**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 two feature points per medial point (Sec. 6.2).

**Applications:** For the segmentation and classification applications in Sec. 6.3, 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 final answer to the underlying use-cases.
6.5 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 abstraction level on which one can reason about medial surfaces, and open ways for constructing new shape processing applications that use medial clouds. We provide, for illustration, sample applications for edge detection and shape segmentation. Overall, our work shows that the more complex surface skeletons can be, technically, used with the same ease and computational efficiency as the simpler, and more frequently used, curve skeletons, without significant additional costs, and with actual significant benefits.

Together with the manifold extraction techniques presented in Chapter 4 and the skeletal density analysis techniques presented in Chapter 5, the techniques presented in this chapter enrich the set of refined medial descriptors that can be easily computed from point-cloud surface skeletons obtained from complex and large 3D models. Along this, we show how these descriptors are useful and usable in supporting a range of surface processing and analysis applications, such as segmentation and classification. All in all, the work in this chapter builds towards our general claim that medial descriptors, complemented by suitable refined descriptors, are efficient and effective tools for shape processing and analysis.