2

MEDIAL REPRESENTATIONS OF OBJECTS

Symmetry is what we see at a glance

Blaise Pascal

2.1 INTRODUCTION TO SHAPE REPRESENTATIONS

SHAPE noun the external form, contours, or outline of someone or something.

According to the above definition, taken from the Oxford Dictionary, shape is an intrinsic property of just about everything which we perceive as an individual object separated from its surroundings. However, shapes can also be abstract mathematical representations, not necessarily representing some entity of what we call the real-world – consider, for instance, an \( m \)-dimensional hypersurface embedded in \( n > m \) dimensions. The focus of the work presented in this thesis is to enable the use of medial representations to perform analysis of shapes of objects that we typically find in three dimensions.

To be able to analyze a shape computationally, we need some mathematical representation thereof. For instance, we can describe a shape by an explicit or implicit function; as being the solution of an equation; or by a (dense enough) set of sample points taken on its surface or its interior. In this sense, we can classify mathematical shape representations into analytic representations and discrete representations. Analytic representations follow the examples of the implicit/explicit function or equation solution listed above. They have the significant advantage of offering a way to exactly reason about the described shape – as long, of course, as the mathematical representation we use accurately captures the shape of interest. However, they also have the crucial disadvantage of being impractical: It is very hard to describe any possible (3D) shape in compact analytic form; moreover, even in cases where such an analytic representation may exist, it is not evident how to create it for a given shape.

Discrete representations offer an efficient and effective trade-off for the above problems. They essentially represent a shape as a dense-enough set of sample points taken either on the surface \( \partial \Omega \) of the shape, or in the interior \( \Omega \) of the shape. The key advantages of this representation are simplicity and generality: Given a dense-enough set of sample
points, we can represent any desired shape up to a given accuracy level, by a single, relatively simple, model: a set of sample points.

Many types of discrete (or sampled) representations of shape exist. They differ in terms of what is sampled (e.g., the boundary $\partial \Omega$ or the interior of the shape $\Omega$); the so-called interpolation functions used to construct a (piecewise) continuous representation of the shape from its samples; the distribution of the sample points within the sampled object; and storage schemes for the sample points. In the following subsections, we provide a brief overview of a number of popular discrete representations of shape.

2.1.1 Volumetric representations

Discrete volumetric representations essentially sample the ‘inside’ of a shape $\Omega$ into a set of sample points $x_i \in \mathbb{R}^n$, where $n$ is the dimension of the space in which the shape is embedded (typically 2D or 3D). The essential information recorded by such a sample point $x_i$, as such, is the shape property (or properties) recorded at the respective spatial location.

Arguably the best-known, and widest used, sampling scheme for such volumetric representations is the one using uniform samples. That is, sample points $x_i$ are distributed on a regular lattice covering a compact axis-aligned region of the embedding $n$-D space. Such samples are known as pixels (for $n = 2$) or voxels (for $n = 3$). The main advantage of this sampling scheme is its simplicity: Essentially, all we need to store is an $n$-D matrix of sample values; the actual sample coordinates can be next easily inferred from the so-called structured coordinates, or indices, of a sample (pixel or voxel) inside the dataset. The main disadvantage of this uniform sampling scheme is its (very) high memory demands: Essentially, we have a uniform sampling density, thus we need to dedicate the same amount of sample points (per unit of $n$-D volume) to any zone. However, some zones may contain more interesting information than other zones.

Uniform volumetric sampling is used by many shape processing applications, ranging from the ubiquitously known image viewers that show images acquired e.g. by digital cameras, to medical imaging applications that show 3D CT or MRI images acquired by volumetric scanning procedures, that subsequently encode the type, density, or dynamics of tissues in a 3D spatial region.

2.1.1.1 Binary Volumes

In many cases, we are not interested in representing, or reasoning about, the internals of a shape $\Omega$, but only about its boundary $\partial \Omega$. Typical examples include classical 3D computer graphics, where we want to show a number of surfaces (but usually are not concerned with what lies inside these surfaces); CAD-CAM applications, where we want first to design
2.1 INTRODUCTION TO SHAPE REPRESENTATIONS

the boundary of a 3D shape, but are not concerned (at this stage) about
the internals of the respective shape; or medical applications in which we
want to separate, or segment, a shape having a given morphology (structure)
from its surroundings, such as a tumor from enclosing tissue, a hard
bone from enclosing soft tissue, or an arterial tree from surrounding less-
vascularized tissue; or computer vision applications, where we want to
isolate specific shapes of interest, such as cars or pedestrians, from sur-
rounding landscape. In all these cases, we actually want to reason about
any point \( x \) in the embedding space \( (e.g. \mathbb{R}^2 \text{ or } \mathbb{R}^3) \) as being inside, on
the boundary, or outside one or several shapes of interest.

A simple and efficient way to encode the above point classification is
to use binary volumetric sampling, or binary volumes. In such a volume,
each sample point essentially has a binary value, indicating whether the
respective point is either inside or outside our given shape(s) of interest.
Points on the boundary between ‘inside’ and ‘outside’, or on the surfaces
\( \partial \Omega \) of our shapes, can be easily found using this binary classification
as e.g. all inner points having at least one outside neighbor point, or
conversely (all outside points having at least one inside neighbor point).

Binary volumes provide a straightforward way of representing a seg-
mentation in a volume. Each voxel is classified in a binary fashion an
the marked voxels usually represent the voxels belonging to the seg-
mented structure. The key disadvantage of this representation, is that
the segmentation resolution is limited to the resolution of the volume.
However, since voxel volumes in itself sample the space, the actual con-
tours lie somewhere in between the voxel volumes. Implicit surfaces pro-
vide means to overcome this limitation. The analysis of digital volumet-
ric shapes, or digital images, is the focus of a specific field of research,
called digital geometry [80].

2.1.1.2 Implicit surfaces

Although binary volumetric representations offer a way to identify sur-
faces separating the inside of our shapes of interest from surrounding
space, they present a key disadvantage: The represented surface is lim-
ited in resolution to the grid used to represent the binary function. In
many cases, we have datasets, where representing a surface in this way,
results in loss of accuracy. In addition, the representation accuracy is af-
fected by transformations to the original data, such as rotation, scale or
translation. Consider for example a 3D CT scan: This is essentially a
3D volume of points \( x_i \) whose scalar values \( s(x_i) \in \mathbb{R} \) represent some
shape property, such as e.g. tissue density. Although the acquired data is
sampled at the discrete positions \( x_i \), the boundaries of the various organs
typically lie in between these positions. A binary voxel surface representa-
tion cannot capture such boundaries, thereby generating potential prob-
lems for the further interpretation and/or processing of such surfaces.
**Implicit surfaces** offer an alternative representation. Consider again our example of a 3D CT scan represented as a volume of points \( \mathbf{x}_i \) whose scalar values \( s(\mathbf{x}_i) \in \mathbb{R} \) represent some shape property, such as e.g. tissue density. We can now define an entire family of surfaces \( S_j \subset \mathbb{R}^3 \) having the tissue density equal to some user-given value \( \tau_j \in \mathbb{R} \). These are also often called isosurfaces, contours, or level-set surfaces of the function \( s \) [146]. To define such isosurfaces, we must first be able to interpolate the sampled values \( s(\mathbf{x}_i) \) to all positions \( \mathbf{x} \) in our volume. This is typically done by using trilinear interpolation of the sampled values \( s(\mathbf{x}_i) \) at the vertices of cubic cells defined by the sample points \( \mathbf{x}_i \). Having the interpolation \( s(\mathbf{x}) \) at any point \( \mathbf{x} \), we can now define \( S_j = \{ \mathbf{x} \in \mathbb{R}^3 | s(\mathbf{x}) = \tau_j \} \).

Implicit surfaces are a particular class of isosurfaces. Given a 3D shape, consider that we can define a so-called scalar indicator function \( f : \mathbb{R}^3 \rightarrow \mathbb{R} \), so that \( f \) is negative inside the shape, and positive outside this shape. In this case, and assuming that \( f \) varies continuously, the actual surface of the shape is represented by the isosurface for scalar-value zero, or the zero level-set, of the indicator function. Indicator functions can be represented using a sampled voxel representation.

Implicit surfaces and isosurfaces can be extracted efficiently from a sampled volume \( s \) using e.g. marching cubes or dividing cubes algorithms [100]. The resulting implicit surfaces represent a piecewise-linear (polygonal mesh) approximation of the surface of our shape of interest. Several shape operations such as union or intersection can be easily computed using implicit representations, as they map to arithmetic and/or logical operations on the respective indicator functions. Other advantages of implicit representations are the ease of generation of a meshed surface representation (convenient for further processing and/or visualization), easy handling of surfaces having complex shapes and/or topologies, and guaranteed orientability and water-tightness of the resulting surfaces. Implicit representations are very popular among medical image segmentation algorithms [113].

### 2.1.2 Boundary sampling representations

Unlike the binary or implicit surface representations, which require an underlying voxel volume to represent the surface, boundary sampling representations encode the shapes surface explicitly, i.e. the boundary positions are directly encoded. In case we know that we want to represent, and next reason about, a single surface of our shape (rather than e.g. a family of surfaces, or the varying material properties of the interior of a volume), the only information we need to encode is the shape of the respective surface.

There exist several ways to represent such surfaces, as follows. Note that, for the sake of simplicity, we next consider only surfaces embedded in \( \mathbb{R}^3 \). The same reasonings apply to surfaces embedded in other dimensions.
2.1.2.1 **Point cloud representations**

A point cloud representation of a surface $S \subset \mathbb{R}^3$ is essentially a point sampling of this surface; in other words, it is a collection of points $x_i \in S \subset \mathbb{R}^3$. Such point sets, also called point clouds, can be acquired by computing points on an implicit surface; by computing points on the interface between inner and outer voxels in a binary volume representation; or, more interestingly, by using 3D scanning techniques which acquire points on the visible surface of a set of given 3D objects using a laser scanner or a time-of-flight camera [49, 104]. The key advantage of such point cloud representations is their compactness: we can easily encode high detail present on complex 3D surfaces up to sub-millimeter precision in point clouds having a few million samples with significantly lower cost than encoding the same information in binary voxel volumes. In many cases, such as the use-case outlined above of recording the visible surfaces of 3D natural objects, it is much easier (and cheaper) to construct a surface point cloud (using scanning techniques) than acquire a true volumetric representation of the same object e.g. using CT or MRI scanning techniques. A third advantage of point clouds is that they naturally support non-uniform spatial sampling schemes: We can distribute the 3D cloud points any way we want; in contrast, in a typical volumetric sampling, the sample points are organized in a regular lattice or uniform grid. Thus, 3D point clouds are significantly more flexible in recording 3D shapes having a highly non-uniform spatial variation, by e.g. allocating more sample points to high-detail or high-frequency areas than to low-frequency, uninteresting, areas.

However, a major limitation of point cloud representations is that they essentially only encode a set of samples on our surface of interest, but not the shape of the surface as such. In other words, point clouds do not (explicitly) say, or model, what happens between the samples. To have such information, we need to interpolate the sample information in-between. This is the area covered by the representations and methods discussed next.

2.1.2.2 **Polygonal representations**

Polygonal shape representations extend the point cloud surface representation by adding connectivity information to the points. From a practical (and historical) perspective, polygonal surface representations are probably the oldest, best known, and most frequently used representations of 3D surfaces. From a theoretical perspective, polygonal representations extend point cloud sample-sets by adding an interpolation mechanism that estimates the surface between sample points, using linear basis functions or interpolants. In simpler terms, this amounts to adding a mesh representation atop of a point cloud that describes a set of polygons (typically triangles) whose vertices lies at the sample point positions, whose union represents our surface of interest, and which do not intersect –
thus, they form a *partition* of the desired surface. Such mesh representations are computationally inexpensive, compact in terms of memory usage, and simple to implement. As such, the vast majority of applications representing or processing 3D surfaces use polygonal representations. To give just an example, the marching cubes implicit-surface extraction algorithm mentioned earlier represents implicit surfaces by triangle meshes.

If a polygonal representation can be constructed for a given point cloud, numerous surface analysis and processing operations can be easily performed on it, *e.g.* normal estimation (by averaging polygon vertices at sample points); smooth shading (using either the linear Gouraud interpolation of shading computed at sample points or the slightly more complex Phong interpolation of surface normals at each surface pixel); surface curvature estimation [176]; surface segmentation [26], and many others. Fundamentally, many of these operations become possible due to the partition of the surface of interest offered by the polygonal representation, and the subsequently easy construction of interpolation mechanisms on this polygonal representation.

We should note that the need for an interpolation mechanism on a surface $S$ that (in turn) interpolates or approximates our point cloud samples $x_i$ does not directly and always require the construction of a polygonal interpolation (mesh) from $x_i$. Mesh-less, or grid-less methods have been proposed. Such methods essentially provide most of the shape processing operations targeted by mesh representations, by creating different sets of basis functions than the classical bilinear ones that live on mesh triangles. Examples are constructing local point couplings by using nearest-neighbor projections to local tangent planes [27, 28, 133]. Such couplings essentially define a stiffness matrix, which next enables the direct application of many processing methods to take place on the ‘implicit’ surface defined by the couplings, *e.g.* anisotropic diffusion, denoising, texture generation, inpainting, or simplified rendering. While the coupling estimation using local tangent planes strongly resembles methods used for reconstructing meshed surfaces from point clouds [1, 195], such mesh-less methods do not explicitly deliver a manifold representation of a surface. As such, mesh-less shape processing methods are less attractive than methods that extract an explicit mesh surface from a point cloud and deliver it to any subsequent processing task.

However, given just a raw point cloud of positions $x_i \in \mathbb{R}^3$, computing a polygonal representation of a surface that interpolates (or approximates) these points is far from trivial. To be useful in subsequent surface processing and/or analysis operations, such a polygonal surface should meet several criteria, *e.g.* be free of self-intersections; contain only cells (polygons) having a non-zero area and good aspect ratio; be orientable; contain a simple manifold structure; or even be watertight (describe a closed volume in $\mathbb{R}^3$ without boundaries). The class of methods aiming at creating such surfaces from 3D raw (or unstructured) point clouds,
known also under the name of surface reconstruction methods, is briefly outlined next.

2.1.2.3 **Surface reconstruction**

Given the usefulness of polygonal surface representations outlined above, and the prevalence of ‘raw’ surface representations in terms of unstructured point clouds, many methods have emerged that aim to create polygonal surfaces from such point clouds, satisfying various quality criteria for the generated polygonal mesh (such as polygon aspect ratio, manifoldness, orientability, and watertightness). Such methods are globally known under the name of **surface reconstruction** methods.

Globally speaking, surface reconstruction methods can be classified into (1) approximation and (2) interpolation methods. Given a point cloud \( \{x_i\} \in \mathbb{R}^3 \), approximation methods generate a surface \( S \) that is globally as close as possible to the points \( x_i \), and also meets a number of global quality criteria, such as the ones stated at the beginning of this section. The degree of freedom of not having to pass precisely through the sample points gives some additional room for optimizing desirable surface properties, such as smoothness, orientability, or polygon size. A salient example of approximation methods is the Poisson surface reconstruction technique of Kazhdan et al., which computes the approximating surface \( S \) by minimizing a global distance function between \( S \) and the sample points \( x_i \), subject to certain smoothing assumptions [77], or similar global functionals [138]. As it will be discussed later in this thesis, this technique works well for certain point-sampling distributions, but creates unwanted results for highly non-uniform point distributions. Interpolation methods generate a surface \( S \) that is guaranteed to pass through the sample points \( x_i \). This creates an overall better control of the result, but imposes implicit constraints on global surface properties. For instance, if \( \{x_i\} \) contains (many) noisy samples, interpolation will necessarily create a noisy surface with limited smoothness. Methods in this class involve 3D Delaunay triangulations and variants thereof, such as the power crust and variations [2, 3]; the ball pivoting method [12]; or local triangulations based on the point cloud covariance matrix [27, 96]. We examine the extraction of surfaces having desirable properties from point clouds, e.g. smooth self-intersecting manifolds with boundaries embedded into sampling noise, further in Chapter 4.

2.2 **MEDIAL REPRESENTATIONS**

In the previous sections, we discussed several representations that encode a shape \( \Omega \) (or, alternatively, its boundary \( \partial \Omega \)) by various ways of sampling and reconstruction of the respective surface from data samples. In this section, an alternative shape representation is introduced – the **medial** representation. As we shall see, this representation offers several
important advantages compared to more traditional volumetric or boundary representations; however, at the same time, additional challenges are brought on, which have to be addressed.

The medial approach, first introduced by Harry Blum in 1967 [15], captures the boundary $\partial \Omega$ of a shape $\Omega$ in terms of its so-called ‘symmetry-set’ or ‘symmetry locus’. That is, instead of explicitly encoding the points on the shape boundary, we encode points which are (in some sense) symmetric with respect to this boundary.

The basic concept underlying the medial representation is, in terms of intuition, quite simple: The shape boundary $\partial \Omega$ is represented as the locus of the centers of maximally inscribed discs ($\mathbb{R}^2$) or spheres ($\mathbb{R}^3$) located inside the shape $\Omega$ (that is, maximal disks or spheres fully located inside $\Omega$). Together with the centers $p_i$ of these disks, we can store their respective radius values $r_i \in \mathbb{R}^+$. The set $\{(p_i, r_i)\}$ is known under many names – skeleton, medial axis, and medial axis transform. We will use next these terms interchangeably, pointing out to relevant differences in respective contexts. For a more formal definition of these terms, see e.g. [154].

Several alternative definitions (and, subsequently, computation methods) have been proposed for the medial axis. One of the most popular is the grassfire analogy [94]. Here, one assumes an isotropic (uniform) grass field covering the extent (interior) of our shape $\Omega$. Next, we assume that the full boundary is set on fire at the exact same time $t_0$, and that the grass burns next with equal speed in all directions and along all points in $\Omega$. The grass points where the fire front clashes at some given time $t > t_0$ represent the medial axis. As we shall see next, this model can be used to derive several computation techniques for the medial axis, having their intrinsic strengths and limitations. Alongside this definition, alternative definitions exist for the medial axis, each having in turn their advantages and limitations. Such definitions and computation methods are discussed in Section 2.3.

However, the medial axis definition (and, subsequently, computation methods thereof) is not free from problems: Small perturbations on the boundary $\partial \Omega$ lead to large variations on the medial axis. Such issues are discussed in Section 2.3.4.

A separate aspect to mention at this moment is the goal behind medial axis computations. While, theoretically, computing medial axes is an interesting (mathematical) problem, the main practical aim we see is, ultimately, using such medial axes to enable the implementation of relevant shape processing operations with a clear practical relevance. As such, we do not see the medial axis as an ultimate goal, or product, of our work. Rather, the medial axis should be a simple, efficient and robust to compute, descriptor or tool that allows us to either perform relevant shape processing operations, or compute derived higher-level shape features that ultimately enable us to perform such operations. We shall cover the computation of such derived features and their use to perform useful
2.2 MEDIAL REPRESENTATIONS

shape analysis operations in Chapters 5 and 6. In this sense, a (loose) analogy could be made between medial vs boundary representations and time vs frequency representations of signals: Both representations in the above-mentioned pairs encode the exact same information in a different format and the usability of each representation is dependent on the application. Additionally, since we can convert between representations (boundary vs medial axis transform, or space vs frequency respectively), we can choose to use whichever representation suits us best for the desired processing operations in a given domain. Our shapes, or input signals, can be always transformed into the other domain as needed.

2.2.1 Definitions

Let us next present the formal definitions of medial structures we will be using in the context of this thesis.

2.2.1.1 2D skeletons and surface skeletons

Given a 2D or 3D binary shape $\Omega \subseteq \mathbb{R}^{n \in \{2,3\}}$ with boundary $\partial \Omega$, we first define its Euclidean distance transform $DT_{\partial \Omega} : \Omega \rightarrow \mathbb{R}^+$ as

$$DT_Y (x \in \Omega) = \min_{y \in Y} \|x - y\|$$

(2.1)

For all definitions in the context of this thesis, $Y$ is set to $\partial \Omega$. For text conciseness purposes, the $DT_{\partial \Omega}$ notation is used. $DT_{\partial \Omega}$ is called Euclidean distance transform as it is based on the Euclidean distance metric $\| \cdot \|$ between points. Other distance metrics induce different distance transforms, see e.g. [166]. However, in the context of this thesis, we will use only the Euclidean distance transform, as this is the most used metric for computing medial descriptors in practice.

Intuitively, the distance transform gives us, for each point inside the shape $\Omega$, the minimal distance to any point on the shape’s boundary $\partial \Omega$. From the definition, we already see that the distance transform is a positive scalar field which assumes increasingly large values as the point $x$ is deeper situated in $\Omega$ with respect to its boundary $\partial \Omega$. Also, we see that $DT_{\partial \Omega}$ takes zero values on the entire boundary $\partial \Omega$. As we shall see next, $DT_{\partial \Omega}$ is an important ingredient in defining (and computing) the medial axis of $\Omega$.

The skeleton of $\Omega$ is next defined as

$$S_{\partial \Omega} = \{ x \in \Omega \mid \exists f_1, f_2 \in \partial \Omega, f_1 \neq f_2, \| x - f_1 \| = \| x - f_2 \| = DT_{\partial \Omega}(x) \}$$

(2.2)
where \( f_1 \) and \( f_2 \) are two of the contact points with \( \partial \Omega \) of the maximally inscribed disc in \( \Omega \) centered at \( x \). The mapping

\[
FT_{\partial \Omega}(x \in \Omega) = \arg\min_{y \in \partial \Omega} \|y - x\| \tag{2.3}
\]

also called the feature transform (FT) of the boundary \( \partial \Omega \) [66, 166], contains all boundary points \( y \) that are at minimal distance from any interior point \( x \). These points \( y \) are also called feature points. The vectors \( y - x \) are also called spoke vectors, in analogy to the spokes of a wheel (whose circumference is represented by the boundary \( \partial \Omega \)) [164, 165].

Note that \( FT_{\partial \Omega} \) is multi-valued – or, in other words, its co-domain is the power set of \( \mathbb{R}^3 \). Indeed, an inscribed ball can have at least two, but potentially many more, contact points with \( \partial \Omega \), depending on the actual position of the ball center \( x \in \Omega \). For instance, if \( \Omega \) is a disk, the value for \( FT_{\partial \Omega} \) for the circle center will be the entire circle boundary \( \partial \Omega \). As we shall see next in Chapter 6, the cardinality of \( FT_{\partial \Omega} \) allows us to reason about the (local) variations of the boundary \( \partial \Omega \).

Skeletons defined by Equation 2.2 have an interesting intrinsic relation to the dimension \( n \) of the space embedding our shape \( \Omega \). That is, if \( n = 2 \), skeletons will be sets of 1D curves; these are called 2D skeletons (with reference to the embedding dimension \( n = 2 \)), medial axes, or even simpler, skeletons. If \( n = 3 \), skeletons will be sets of 2D manifolds (surfaces). Such skeletons are typically called surface skeletons or medial surfaces. Note that in the trivial case \( n = 1 \) (our shapes are compact 1D line segments), skeletons will be sets of isolated points. As such, the skeleton dimension is one less than the dimension of the space embedding \( \Omega \). For \( n = 3 \), this already highlights an inherent problem concerning the accurate and efficient computation and analysis of such skeletal descriptions, which are sets of complex 2D manifolds. Obviously, the complexity of such skeletons will only increase with the size of the dimension \( n \). Since the case \( n = 3 \) is challenging enough, this thesis will not focus on skeletons of shapes in higher dimensional spaces such as given by \( n > 3 \).

The skeleton defined by Equation 2.2 is homotopic to the input shape \( \Omega \) – that is, it preserves the topological elements of the input shape [126, 154, 168]. For instance, if the input shape is of genus \( g \), then the skeleton also has to be of genus \( g \). The definition results in a connected skeleton, as long as the input shape \( \Omega \) consists of a single connected surface. In the case that \( \Omega \) consists of multiple disconnected components, the solutions presented for computing the skeletons still hold, however the discussion of homotopy becomes implicitly more complex[185][186]. In the context of this thesis, unless explicitly expressed otherwise, the input shape \( \Omega \) is considered to consist of one and only one connected component.

Having the skeleton of a shape, the medial axis transform (MAT) \( M_{\partial \Omega} \) of a shape \( \Omega \) can be next defined as a function that associates, to each skeletal point \( x \), the minimal distance of \( x \) to the boundary \( \partial \Omega \). This
distance is equal to the value of the distance transform $DT_{\partial \Omega}$ at point $x$. Thus, we can define the MAT as

$$MAT_{\partial \Omega}(x \in S_{\partial \Omega}) = (x, DT_{\partial \Omega}(x)).$$

(2.4)

An important property of the MAT is that it provides a full encoding of the shape described in $\partial \Omega$. Indeed, given the MAT of a shape embedded in $\mathbb{R}^n$, we can reconstruct the respective shape $\Omega$ as being the union of $n$-dimensional balls centered at the MAT points and having as radii the distance values provided by the respective MAT points. In other words

$$\Omega = \bigcup_{x \in S_{\partial \Omega}} B(x, DT_{\partial \Omega}(x)).$$

(2.5)

Here, $B(x, \rho > 0)$ denotes an $n$D ball centered at $x$ and of radius $\rho$. Subsequently, given the reconstruction of the shape $\Omega$, we can easily find its boundary $\partial \Omega$, if so desired.

This equivalence of a shape with its medial axis transform is an important element for the work presented in this thesis. In brief, it suggests that shape analysis and processing operations that are traditionally defined on a surface $\partial \Omega$ can be adapted to work on the medial axis transform representation $MAT_{\partial \Omega}$. As we shall see, this transposition is possible, and can lead to several interesting insights and practical advantages.

### 2.2.1.2 Curve Skeletons

As we have seen above, surface skeletons are relatively complex shapes consisting of a set of intersecting manifolds with boundaries. Both researchers and practitioners have since long recognized many of the computational and practical challenges of using such skeletons in real-world applications, where the input shapes are embedded in 3D. We discuss several such challenges next in Section 2.3.

As such, other simpler and thus more practical descriptors of ‘centrality’ for 3D shapes have been proposed. One such descriptor type is formed by curve skeletons. Topologically speaking, curve skeletons $C_{\partial \Omega}$ are sets of 1D curves. Given their lower dimensionality as compared to surface skeletons, curve skeletons are simpler to represent, analyze, and use. However, in contrast to the formal definition of surface skeletons (Eqn. 2.2), curve skeletons know several definitions in the literature. Apart from their dimensionality (sets of 1D curves), such definitions have in common the idea that curve skeletons should be ‘locally’ centered within the shape $\Omega$, in analogy to the local centeredness of surface skeletons. However, there are many ways in which one can define curves that are locally centered within a shape. As such, many curve skeleton definitions and computation methods have emerged. Since there is no uniquely accepted formal definition of curve skeletons, researchers have...
tried to characterize these by a number of qualitative ‘desirable properties’ [32, 158]. Such properties include

- **centeredness**: Each point $\mathbf{x} \in C_{\partial \Omega}$ should be centered with respect to a neighborhood of points $\mathbf{y} \in \partial \Omega$ that is close to $\mathbf{x}$;

- **thinness**: Curve skeletons $C_{\partial \Omega}$ of 3D shapes are sets of 1D curves;

- **topology**: The topology of curve skeletons $C_{\partial \Omega}$ should capture the topology of the input shape $\Omega$. More specifically, $C_{\partial \Omega}$ should be homotopic to $\Omega$;

- **transformation invariance**: Given an isometric transformation $T : \mathbb{R}^3 \to \mathbb{R}^3$, such as translation, scaling, or rotation, the skeleton should be invariant to it. That is, $C_{T(\partial \Omega)} = T(C_{\partial \Omega})$, for any such $T$.

Other desirable properties of curve skeletons are mentioned such as computational stability in the presence of noise on the input shape, computational efficiency, and minimal thickness. While important, such additional properties are related to choices involving space discretization and computation algorithms, and therefore not key to our discussion here on the definition of curve skeletons. Separately, note that all above desirable properties apply as well to 2D medial axes and 3D surface skeletons.

From a definition viewpoint, two additional noteworthy approaches should be mentioned. One of the first formal and explicit definitions of curve skeletons is the locus of points $\mathbf{x} \in \Omega$ which admit at least two (equal-length) shortest paths, or geodesics on $\partial \Omega$, between their feature points [37, 125, 135]. This definition is equivalent to stating that the curve skeleton of a shape $\Omega$ is the ridge set, or local maxima in at least one dimension of the so-called medial geodesic function $MGF : \mathbb{R}^3 \to \mathbb{R}^+$, defined as the length of the shortest path on $\partial \Omega$ determined by any two feature points of the point $\mathbf{x}$. By definition, $MGF$ can be only computed on the surface skeleton $S_{\partial \Omega}$, since points $\mathbf{x} \notin S_{\partial \Omega}$ have a single feature point, thus cannot allow constructing the above-mentioned shortest paths. This leads to the implicit additional property that curve skeletons are contained in the surface skeletons of the respective shapes, or $C_{\partial \Omega} \subset S_{\partial \Omega}$. This observation has been further exploited in a second, separate, definition of curve skeletons as the ridge set of the distance transform $DT_{\partial S_{\partial \Omega}}$, which measures the shortest (geodesic) distance from any point $\mathbf{x} \in S_{\partial \Omega}$ to the boundary $\partial S_{\partial \Omega}$ of the surface skeleton [181]. Several relations between curve and surface skeletons, as computed by a number of recent methods, are discussed in [159].

Due to their simple structure, curve skeletons are useful in a variety of applications such as shape matching, shape registration, path planning, 3D metrology, and virtual navigation [32]. In particular, curve skeletons are effective shape descriptors for *tubular* objects, *i.e.* objects which can be well described by the extrusion of a (near) circular contour along a set
of 3D curves. Such objects occur in many applications, e.g. blood vessel
trees obtained from CT or MRI scanning techniques, intestine or colon
structures obtained by similar scanning procedures, or plant branches or
roots, measured by laser scanning or vision techniques. However, com-
pared to surface skeletons, curve skeletons do not admit the definition
of a MAT that would allow the perfect reconstruction of the input shape
Ω. As such, it is arguable that many surface analysis and processing op-
erations can not be fully translated to a shape representation involving
only curve skeletons, but require the richer surface skeletons. This is an
additional justification for the subsequent focus of this thesis on surface
skeletons.

2.2.2 Classification of Medial Points

As outlined earlier, 2D skeletons are formed by sets of 1D curve seg-
ments that intersect at a number of points (also called junctions); 3D me-
dial surfaces are formed by sets of 2D surface manifolds with boundaries,
that intersect along several curves (also called Y-intersection curves [23,
34, 93]; and curve skeletons are formed by sets of 3D curve segments
that intersect at a number of junction points. As such, we can say that
skeletons (of all types discussed so far) exhibit a high amount of structure.
Computing and understanding this structure and its relationships
with the input shape Ω is of great use in analyzing and processing shapes
by using their medial descriptors. We give below a brief overview of sev-
eral definitions and results in this direction. For additional information,
we refer to [23, 34, 93, 154].

In two dimensions, the medial axis $S_{\partial \Omega}$ of a shape $\partial \Omega$ consists, in
the generic case, of a collection of 1D curve segments. Each such seg-
ment has thus two end points. These can be next classified as ‘termi-
nal’ points, also called tip points or tips; and intersection points, or junc-
tions. Tips are end points which are shared by a single skeleton branch.
They form, thus, the boundary $\partial S_{\partial \Omega}$ of the medial axis. It can be next
shown that tips map, by means of the feature transform, to fragments of
the boundary $\partial \Omega$ where this boundary exhibits local curvature maxima.
These correspond to convex areas, or ‘cusps’, on the shape boundary. For
instance, consider a 2D rectangle shape: Its skeleton has four tip points,
which precisely map, via the feature transform, to the four corners of
the rectangle. If we consider the same rectangle, but slightly ‘round off’
its four corners, e.g. by replacing each boundary fragment centered at
a corner by a small quarter-circle, the resulting skeleton will still have
four tips. However, these tips will now map, via the feature transform, to
four quarter-circle fragments on the boundary $\partial \Omega$, capturing the bound-
dary regions represented by the rounded corners. Junctions are points
where three or more curve fragments of $S_{\partial \Omega}$ meet. They correspond, in-
tuitively, to points where several distinct parts of the shape Ω get joined
together. Overall, 2D skeletons $S_{\partial \Omega}$ can be thus represented as graphs
whose nodes encode the separate skeleton curve segments (or skeleton branches), and edges represent the adjacency relations of branches meeting at a junction point (or conversely). Node and/or edge weights can be added to represent geometric properties measured on the shape elements corresponding to the respective skeleton fragments or junctions, such as (average) distance-to-boundary, curvature, or length. Such graphs thus represent a compact, though not lossless, encoding of the main properties of the analyzed shape. They are frequently used to implement shape retrieval and shape matching operations in terms of graph comparison operations [42, 102, 169].

In three dimensions, understanding the structure of medial surfaces becomes a considerably more complex problem. One way to tackle this higher complexity is to classify the medial points \( x \in S_{\partial \Omega} \) based on the so-called order of contact of maximally inscribed spheres centered at these points with the boundary \( \partial \Omega \). This is, loosely speaking, equivalent to studying the values that the feature transform \( FT_{\partial \Omega}(x) \) takes at such points. Giblin and Kimia proposed such a classification, where each medial point is denoted as being of type \( A^n_k \) is introduced, where \( n \) corresponds to the number of different \( k \)-fold tangencies and \( k \) to the contact order [58]. Note that this classification is equivalent to analyzing the number \( n \) of disjoint groups that \( FT_{\partial \Omega}(x) \) consists of, and the dimensionality \( k \) of each such group respectively.

Following this classification, a medial surface \( S_{\partial \Omega} \) can be decomposed into the following point types:

**Sheets**

Medial sheets characterize the two dimensional elements of a medial surface – thus, each medial sheet is a manifold with boundaries. For typical shapes (excluding spheres and tubular shapes), the vast majority of medial points are located on medial sheets. Medial sheets contain \( A^2_1 \) medial points, i.e. are the locus of maximally inscribed spheres which touch the surface \( \partial \Omega \) at exactly two different points. In other words, these medial points have each a feature transform consisting of exactly two different points on \( \partial \Omega \).

**Curves**

Curve points are located on the boundaries of the medial sheets. Two types of curve points can be next identified, depending on the boundary types. First, we have curves characterized by \( A^3_3 \) points. These curves are the locus of maximally inscribed spheres which have a single contact zone with \( \partial \Omega \), which represents a surface fragment of \( \partial \Omega \). Such curves represent the ‘open’ boundary of sheets. Their union creates the boundary \( \partial S_{\partial \Omega} \) of the surface skeleton. The points on \( \partial \Omega \) to which such boundary curves map correspond to the maxima of the surface curvature of \( \partial \Omega \) – or, more informally, to convex edges of \( \partial \Omega \). \( A^3_3 \) points are thus the 3D
2.2 MEDIAL REPRESENTATIONS

Figure 2.1: Medial cloud classification into different point types.

The medial curves defined above can be further characterized by studying their end points. Two types of curve end points can be defined, as follows. First, $A_4^1$ points are defined as the intersection of four $A_3^1$ curves meet. These are thus centers of maximally inscribed spheres which have four different contact points with $\partial \Omega$. Loosely put, we can think of them as the ‘internal corners’ of the medial sheets. Secondly, $A_4^1 A_3^1$ points are defined as the intersection of an $A_3^1$ and an $A_4^1$ curve – or alternatively, the end points of $A_3^1$ curves. They are the centers of maximally inscribed spheres having one regular tangency point and one higher-order tangency contact zone with $\partial \Omega$. They can be loosely seen as the ‘external corners’ of the medial sheets. These points map, by means of the feature transform, to corners of $\partial \Omega$, where several surface edges meet.

The above point classification of medial surfaces has many important uses. First, it allows us to decompose (and reason about) the complex structure of a medial surface in terms of simpler separated elements and relationships thereof. For instance, the set of $A_3^1$ curves forms the so-called Y-network of a surface skeleton, which defines how different shape parts (characterized by $A_2^1$ sheets) are joined together. More generally, the entire set of sheets, curves, and curve endpoints, and the rela-
tions between them, can be described by a graph structure, also called the medial scaffold [93]. Analyzing this graph structure supports a number of important operations such as shape segmentation [23]. In Chapter 6, we shall show how such point classifications can be efficiently computed for large surface skeletons represented as point clouds, and how they can be subsequently used to support a variety of shape analysis and processing operations.

2.3 AN OVERVIEW OF SKELETONIZATION TECHNIQUES

In the previous sections, we have provided an overview of the main definitions and concepts associated with medial descriptors. Separately, Section 1.4.2 presented several classes of applications where such descriptors are useful.

However, to make medial descriptors – that is, both the ‘raw’ skeletons and their derived properties such as feature transforms, MATs, and skeletal point classifications – useful and usable in practice, we need efficient and effective ways to compute such descriptors from a given shape representation. Methods that compute medial descriptors given an input shape are called medial representation extraction methods, or more briefly, skeletonization methods. In this section, we provide a compact overview of the main classes of skeletonization methods known in the literature, together with several example algorithms in each class.

In line with the shape representation models discussed earlier in Sections 2.1.1 and 2.1.2, we can classify skeletonization methods into volumetric ones and boundary-based ones. These two classes of methods are discussed next in Sections 2.3.1 and 2.3.2 respectively. In these sections, we focus on the extraction of skeletons which follow the canonical definition given by Equation 2.2, i.e., 2D medial axes and 3D surface skeletons. 3D curve skeletons represent a particular case, due to the strong dependence between the extraction method and the underlying curve-skeleton definition that the respective method aims to support (see Section 2.2.1.2). As such, we discuss such methods separately in Section 2.3.3. A separate important computational concern is the problem of regularization, or elimination of unwanted details and/or noise from the extracted skeletons. Since regularization can be seen as a cross-cutting concern for virtually all skeletonization methods, we discuss it separately in Section 2.3.4.

2.3.1 Volumetric Methods

As suggested by their name, volumetric (or voxel-based) methods start with a voxel representation of the input shape \( \Omega \). Typically, this comes as a binary volume, with foreground voxels describing locations inside \( \Omega \), and background voxels describing locations outside \( \Omega \) respectively. Note
that, in this discussion, we use the term ‘voxel’ mainly because our interest is in computing 3D surface skeletons. However, our characterization of volumetric methods also covers the 2D case, where the equivalent term is ‘pixel’. To represent the computed skeleton \( S_{\partial \Omega} \), we also have two options: use a voxel representation (analogously to the one capturing the input shape), or use other representations, such as boundary sampling. Most, though not all, methods that use a voxel-based representation of \( \Omega \) will subsequently also use a voxel-based representation for \( S_{\partial \Omega} \). This has the advantage of simplicity – a single shape representation is used throughout the computational pipeline. However, as we shall see, this introduces several practical and conceptual challenges. In contrast, methods that use a voxel-based representation for \( \Omega \) but a boundary-sample representation for \( S_{\partial \Omega} \) avoid such challenges. However, they are more complex to implement, as the computational pipeline has now to deal with both volumetric and boundary-sampling representations.

Volumetric skeletonization methods can be in turn broadly divided into two categories: Morphological thinning and distance field based methods. These two categories are discussed next. For a recent overview of volumetric skeletonization methods, we refer to [159].

2.3.1.1 **Morphological thinning**

Morphological thinning approaches are among the first known skeletonization methods. Their idea is simple: Given a shape \( \Omega \) represented by the set of foreground voxels, the method iteratively removes so-called boundary voxels (foreground ones which have at least one 4-connected (in 2D) or 6-connected (in 3D) background voxel). Thereby, the shape \( \Omega \) is effectively ‘thinned’, until we obtain its skeleton \( S_{\partial \Omega} \). During this process, two elements are essential: First, voxels should be removed in an order which ensures that the resulting skeleton is centered within the shape \( \Omega \), thereby satisfying the definition given by Equation 2.2. Secondly, voxel removal should stop at some point – otherwise, we obtain \( S_{\partial \Omega} = \emptyset \). The stopping criterion has to accommodate further constraints:

- the skeleton is a ‘thin’ \( n - 1 \) dimensional subspace of the space \( \mathbb{R}^n \) in which the input shape \( \Omega \) is embedded (see Section 2.2.1). If we use a volumetric representation for \( S_{\partial \Omega} \), the skeleton should be as thin as possibly allowed by this representation, e.g., a one-pixel-thick curve for \( n = 2 \), respectively a one-voxel-thick surface for \( n = 3 \). Thus, thinning should not stop too early, otherwise we obtain a ‘thick’ skeleton.

- the skeleton is homotopic to the input shape \( \Omega \) (see Section 2.2.1). Thus, thinning should stop early enough, otherwise we may disconnect the skeleton, thereby altering the homotopy property.

Many thinning methods exist which enforce the above desirable criteria in various ways. Tools from mathematical morphology [145] were
among the first used to compute curve skeletons by thinning. The residue of openings, based on Lantuéjoul’s formula [92], usually leads to disconnected skeleton branches. Methods based on homotopic thinning transformations yield thin and connected skeletons [13, 107, 119, 120]. This is typically done by using suitable local filters, or templates, that check whether the removal of a voxel does not change the thinned shape’s topology. Such templates are typically very small in size, e.g. 3^t up to 5^n voxels. Centeredness can be helped by removing boundary voxels in the order given by their distance-to-boundary value, i.e., in terms of the distance transform DT∂Ω [4, 126, 171].

Thinning approaches present several advantages based on their algorithmic simplicity: They are relatively simple to implement and are relatively fast when compared to alternative methods (further explored below): Although naive thinning algorithms visit the volume at each iteration, resulting in expensive implementations, approaches based on sorted queues cut the timings to O(N) where N represents the number of voxels. Also, they can readily accommodate the computation of all three skeleton types discussed so far (2D medial axes, 3D surface skeletons, and 3D curve skeletons), by suitable choices of the local thinning filters being used. However, they have a key challenge: The skeleton S∂Ω is represented on the same volumetric grid as the input shape Ω. As such, there are situations when this grid is unable to capture locations that are at equal distance from two boundary points. For instance, consider a simple example of a 2D axis-aligned rectangle of width equal to an even pixel count: There is no pixel-grid location which is at equal distance from both the left and right vertical rectangle edges. As such, thinning methods typically have to somehow relax the skeleton centeredness (thus, relax the definition in Equation 2.2), or alternatively compute skeletons which are not pixel or voxel thin. As a consequence of this, skeletons computed by thinning can be sensitive to isometric transformations such as rotation, translation or scaling.

2.3.1.2 Field-based skeletonization

Field-based skeletonization methods find S∂Ω as the singularities of a given scalar or vector field f : R^n defined over the shape Ω. As such, they work by first computing a (typically volumetrically-sampled) representation of f over Ω, and next detecting the desired singular points to form the skeleton. The main differences between sub-methods in this class amount to the specific choice of field and type of singularity to compute.

The arguably simplest field-based method detects S∂Ω as the ridge points of the distance transform scalar field DT∂Ω. These correspond, indeed, to centers of maximally inscribed disks in the shape. Intuitively, if we consider the n = 2 case, and visualize the graph of DT∂Ω as a height plot, then S∂Ω corresponds to the ridges of this graph; these are precisely the locations where fire fronts originating at different bound-
ary points would meet, following the grassfire analogy. One advantage of this method is that computing $DT_{\partial \Omega}$ can be done very efficiently in both 2D or 3D. For this, we can use e.g. the fast marching method (FMM) \cite{146}, which finds $DT_{\partial \Omega}$ as the solution of the Eikonal equation $\|\nabla DT_{\partial \Omega}\| = 1$ with boundary conditions $DT_{\partial \Omega}(x \in \partial \Omega) = 0$. The complexity of this method is roughly $O(\|\Omega\| \log \|\partial \Omega\|)$, where $\|\cdot\|$ indicates number of voxels in the respective structure. Distance transforms can be also computed with $O(\|\Omega\|)$ cost \cite{106}. Such algorithms can be efficiently parallelized on graphics processing units (GPUs) \cite{20, 40, 40, 166}. However, detecting ridges of $DT_{\partial \Omega}$ can be a very sensitive process. In the best case, fronts meeting at such ridges come from opposite directions, so the ridge angle (90 degrees) is easy to detect, e.g. by using an edge detector on the distance image. In a similar setting, Stolpner et al. find skeleton voxels as the points where the gradient of the shape’s distance transform is multi-valued \cite{164, 165}. However, in all the above approaches, we can see that fronts can meet at arbitrarily low angles along a so-called ligature skeleton branch, yielding thus very low ridge angles which are hard to find \cite{57, 78}. Missing such points will thus disconnect the detected skeleton into multiple fragments.

One solution to this problem is to explicitly find the skeleton as the locus of points where the feature transform (Equation 2.3) has more than a single boundary-point as value. As discussed above for thinning methods, this should be done with care, since we are essentially testing distance-to-boundary equality to capture a singularity (the skeleton) on a fixed grid. One way to relax the issues induced by the fixed grid is to compute so-called tolerance-based distance and/or feature transforms \cite{128}. These amount to replacing the strict equality relations present in the right hand sides of Equations 2.2 and 2.3 by distance comparisons within the range of a (small) user-given tolerance value $\tau > 0$. Tolerance-based feature transforms are used to compute surface skeletons as voxels in $\Omega$ that have at least two different feature-points on $\partial \Omega$ \cite{135}. A different approach that detects skeletons by using interval arithmetic to evaluate Equation 2.2 by the so-called integer medial axis (IMA) \cite{66}. The IMA can be efficiently computed with $O(\|\Omega\|)$ cost. A third solution is to use a different field than the distance transform, so that singularities of that field can be found more robustly. Many such fields have been proposed, e.g. second-order moments \cite{140} or divergence-based detectors \cite{139, 155}. While such methods can alleviate some of the difficulties of using the distance transform as ‘raw’ detector, they also use relatively smoother fields, and introduce the question of finding the right threshold that detects skeleton points, which make voxel-precise skeleton localization harder. To tackle this issue, one can detect a conservative skeleton detection \cite{168}: first, we find a superset of points $S_{\partial \Omega} \subset X$ that contains the skeleton and in the same time is reasonably close to the exact skeleton location; next, the exact skeleton is found by homotopical thinning of $X$, using one of the thinning methods discussed earlier.
Overall, field-based methods are less sensitive to local decisions than comparable thinning methods, and have a comparable computational complexity. Also, depending on the field being used, they are less sensitive to isometric transformations than thinning methods. However, they suffer from the same centeredness issues implied by the fixed discretization grid. They also introduce issues related to detection sensitiveness to noise and/or small details. The latter type of issues is discussed separately in the context of regularization (Section 2.3.4).

2.3.2 Boundary sampling methods

In contrast to the volumetric approaches discussed in Section 2.3.1, boundary-sampling approaches represent the input shape \( \Omega \) by a sampling of its boundary \( \partial \Omega \). This follows the techniques discussed earlier in Section 2.1.2 for representing contours in 2D and surfaces in 3D: The boundary \( \partial \Omega \) is represented either as a point cloud (possibly including per-point surface-normal information), or, more generally, as a 3D unstructured mesh. The resulting skeleton \( S_{\partial \Omega} \) can be represented, just as in the case of volumetric methods discussed earlier, by a boundary sampling (thus, using the same representation as for the input shape), or by a volumetric model (thus, using a different representation than the input shape). If both the input shape and its skeleton are represented as a boundary sampling, this leads to two important advantages (as opposed to volumetric-only or volumetric-boundary mixed representations): (1) The memory requirements of such a representation are significantly lower than for any other representation (see [159]; (2) Space can be freely sampled, i.e., we can place sample points for both the input surface and resulting skeleton at any desired location and/or with any desired density (up to the machine precision of floating-point numbers).

We overview next several subclasses of skeletonization methods based on boundary-sampling representations.

2.3.2.1 Voronoi-related methods

The methods in this class are based on the key observation that, given a set of points \( x_i \in \mathbb{R}^n \), which are supposed to (densely) sample a shape boundary \( \partial \Omega \subset \mathbb{R}^n \), the skeleton \( S_{\partial \Omega} \) is a subset of the \( n \)-dimensional Voronoi diagram of the points \( \{x_i\} \). Thus, to compute skeletons, we need to (a) compute a Voronoi diagram, and (b) have a procedure to select relevant subsets of this Voronoi diagram.

For (a), many methods exist for both the 2D and 3D cases, see e.g. [11, 153]. These methods require a mesh version of the input shape and deliver the Voronoi diagram thereof also as a mesh, thus, fall within the class of boundary-sampling methods discussed here. For (b), one of the best known approaches is the power crust method [2], which finds skeletons as the subset of Voronoi-diagram elements which correspond to
balls located (deeply) inside the surface described by \( \{ x_i \} \). In 2D, a method to compute medial axes from Voronoi diagrams was presented by Ogniewicz and Kubler [118] – this method is discussed in more detail in the context of skeleton regularization in Section 2.3.4. Other related methods use edge collapses [95], starting from a mesh segmentation [76], or sphere sweeping [112].

Voronoi-based methods have the large appeal of being able to use existing formalisms (and technology) to compute skeletons atop of Voronoi diagrams. Several works have shown that, as the sampling density of \( \partial \Omega \) increases, the resulting skeletons converge to the true skeletons of \( \partial \Omega \) [2, 144]. Separately, the centeredness problems discussed in the context of volumetric methods are largely inexistent for Voronoi-based methods, since we can place both input-surface points and output-skeleton points at any location in \( \mathbb{R}^3 \) up to machine precision. However, Voronoi-based methods also come with several challenges. First, computing an accurate Voronoi diagram for a large and complex 3D shape is challenging both from computational and implementation viewpoints. Computational geometry algorithms involved are far from trivial. To give just a simple example, the C code of the 2D Voronoi algorithm in [153] is about 14K lines, whereas the code required to implement 2D IMA [66] is under 500 lines. Limit cases, e.g. involving nearly-identical or nearly collinear points can cause significant algorithmic trouble. Separately, such algorithms are far harder to parallelize than volumetric algorithms discussed earlier [167].

2.3.2.2 Point-cloud approaches

An alternative proposed to tackle the computational and implementation complexity of Voronoi-based approaches is to directly apply the skeleton definition (Equation 2.2) to the point cloud representing the sampling of the input surface \( \partial \Omega \). Putting it simply: If we are able to determine the centers \( x_i \) of maximally inscribed balls in the volume \( \Omega \), we have been able to compute a point-sampling of the skeleton \( S_{\partial \Omega} \). If this can be done (a) avoiding the heavy implementation costs of computing complex Voronoi diagrams and next selecting suitable subsets thereof, and (b) keeping the sampling advantages of boundary-based representations, then we have simple, efficient, and accurate methods to compute skeletons. Additionally, if we represent \( \partial \Omega \) by a simple point cloud (rather than a mesh), then many subsequent implementation complexities would arguably disappear.

Interestingly, there are very few methods that we are aware of in this class. One salient method in this class was presented by Ma et al. [101]. Given an oriented point-cloud sampling of \( \partial \Omega \), they compute, for each input point, a maximally inscribed ball. Key to this computation is the efficient search of a maximally inscribed ball tangent at the search point,
which translates to an efficient nearest-neighbor search on the input point cloud.

One major advantage of this computational scheme is the per-point independence of each medial ball computation, meaning that medial points can be computed in parallel. This makes the proposed method highly parallelizable. A parallelization solution involving GPUs is presented. Yet, the per-input-point costs for determining maximally inscribed balls can largely differ. This in turn causes serious performance issues in a SIMD (single instruction, multiple datastream) parallelism model, such as the one offered by typical modern GPU programming platforms (OpenCL or CUDA). These issues are well known under the name of ‘thread divergence’ in GPU programming.

A second issue with point-cloud approaches is that they provide only an unstructured point sampling of \( S_{\partial \Omega} \). While this representation may be sufficient for performing certain tasks, such as shape metrology or visual reconstruction (as we shall next see in Chapter 3), performing more complex analyses or operations that involve reasoning about the skeleton-point connectivities is not possible using just a point cloud. We examine the issue of recovering such missing information in Chapters 3 and 6.

### 2.3.3 Curve skeleton methods

As outlined in Section 2.2.1.2, 3D curve skeletons do not have an universally accepted definition. Therefore, many algorithms exist for computing them. Interestingly, each such algorithm implicitly provides its own definition of what a curve skeleton is. Therefore, the task of deciding what is the ‘right’ curve skeleton for a given 3D shape (and next, how to compute it) is far from trivial.

Early methods compute curve skeletons by thinning, or eroding, a voxel representation of the input shape \( \Omega \) in the order of its distance transform, until a connected 1D voxel curve is left [10, 119]. Thinning can also be used to compute so-called meso-skeletons, i.e. a mix of surface skeletons and curve skeletons [98]. Curve skeletons can also be computed as an intersection of 2D medial axes computed from axis-aligned slices of a 3D voxel shape – thereby reducing the problem of curve skeletonization to a 2D mexial axis extraction [182]. In a volumetric setting, other methods involve finding and connecting the local maxima of the input shape’s distance transform, with explicit restrictions that the resulting object should be a curve [14, 192].

For mesh-based models, alternative techniques collapse the input mesh describing \( \partial \Omega \) along its surface normals under various constraints required to maintain its quality [7]. The result captures a point-sampling of the curve skeleton of \( \Omega \). Hassouna et al. present a variational technique which extracts the curve skeleton by tracking salient nodes on the input shape \( \Omega \) in a volumetric cost field that encodes centrality [64]. As
such, this method can be seen as a hybrid between volumetric methods (used to compute the centrality metric) and boundary-representation methods (used to describe the structure of the resulting curve skeleton). Tagliasacchi et al. compute curve skeletons as centers of point-cloud projections on a cut plane found by optimizing for circularity [172]. Similarly, the ROSA method finds curve-skeleton points as the centers of local point-cloud projections of \( \partial \Omega \) under a constraint for circularity optimization [172]. An extensive comparison of curve skeletonization methods is presented in [158].

Curve skeletons can also be extracted by collapsing a previously computed surface-skeleton towards its ‘center’ using different variants of mean curvature flow [19, 174, 181]. This class of methods builds upon the assumptions that (a) the curve skeleton is a subset of the surface skeleton; and (b) the curve skeleton can be seen as a ‘local center’ of the surface skeleton with respect to a distance metric between a surface-skeleton point and the surface-skeleton boundary \( \partial S_{\partial \Omega} \). As such, these approaches see the computation of curve skeletons as a ‘recursive’ skeletonization operation: First, we compute the surface skeleton from the input shape \( \Omega \); next, we compute the curve skeleton as the skeleton of the surface-skeleton (under the same distance metrics, albeit defined over a different space). The main strength of these approaches is precisely their recursive approach: Given a boundary, space, and distance metric, skeleton definitions are the same (for surface and curve skeletons). The main practical problem of these approaches is implementation complexity: To reason about e.g. the curve skeleton of a shape, we need to (a) have a robust and efficient way to compute the boundary \( \partial S_{\partial \Omega} \) of a surface skeleton (i.e., the union of curves of \( A_3 \) points described in Section 2.2.2); and (b) have a robust and efficient way to compute shortest paths (geodesics) over \( S_{\partial \Omega} \) from any internal point to its boundary.

Curve skeletons can also be defined directly based on a representation of \( \partial \Omega \). A pioneering method in this direction is presented in [37]. As outlined in Section 2.2.1.2, they define the curve skeleton of a shape \( \partial \Omega \) as the locus of points in \( S_{\partial \Omega} \) which maximize the length of the medial geodesic function (MGF), defined as the length of the longest shortest-path between any two feature points. The MGF metric is next evaluated using a mixed volumetric and boundary-sampling approach to yield curve skeletons. Based on this idea, an improvement is proposed in [135], in a pure volumetric setting, to compute curve skeletons as the locus of surface-skeleton points that admit two equal-length longest shortest-paths between all pairs of their feature points. Both above approaches are remarkable as they (a) compute curve skeletons which visually agree with curve-skeletons computed by completely different methods; (b) provide a formal definition of curve skeletons, rather than saying that curve skeletons are the output of a given (iterative) algorithm; but (b) cannot provide a formal reason of why the commonly used geodesic metric is a ‘good’ criterion for detecting curve skeletons.
2.3.4 **Regularization Methods**

We have seen that both volumetric and boundary-representation methods (for both the input shape $\Omega$ or its skeleton $S_{\partial \Omega}$) are inherently sensitive, up to various degrees, to sampling resolution. As such, the practical question of skeleton-computation stability arises.

We can refine this question on both theoretical and practical grounds. On the theoretical ground, assume we have to shapes $\partial \Omega_1$ and $\partial \Omega_2$ which are very similar. More formally, we can quantify the shape similarity using e.g. the Hausdorff distance between the two shapes [11]. The question is then: Would similar shapes (under the above-defined distance metric) yield similar skeletons (under the same distance metric)?

The answer is, unfortunately, negative. Consider the operator, or function, $\mathcal{S}$ that, given a shape $\partial \Omega$ computes its skeleton $S_{\partial \Omega}$. We can easily argue that the function $\mathcal{S}$ is not continuous in the definition of Cauchy (or Weierstrass) continuity, also called the $\varepsilon - \delta$ criterion. Indeed, if $\mathcal{S}$ were continuous at any point $c$ in the definition domain (thus, any possible shape), then we would have that for any real value $\varepsilon > 0$, however small, there exists some real value $\delta > 0$ such that for all $x$ in the domain of $\mathcal{S}$ (thus, shapes) with $\|x - c\| < \delta$, the value of $\mathcal{S}(x)$ satisfies $\|\mathcal{S}(x) - \mathcal{S}(c)\| < \varepsilon$. Here, $\| \cdot \|$ denotes distance between two shapes embedded in the same space, e.g., the Hausdorff distance. The above $\varepsilon - \delta$ assumption is, however, not happening over the space of considered shapes. Consider, for instance, a perfectly 2D rectangle $c$. Consider now an infinitesimally small ‘bump’ added to any of the rectangle edges — that is, shifting the edge outwards with a small distance. Obviously, the Hausdorff distance $\varepsilon = \|x - c'\|$ between the ‘pure’ rectangle $c$ and ‘perturbed’ rectangle $c'$ can be made as small as desired. However, such a (small) bump on $c'$ will cause the appearance of a skeleton branch in $\mathcal{S}(c')$ whose length is *not* a function of the bump size $\varepsilon$ — indeed, the length of this branch, or in other words the value $\delta = \|\mathcal{S}(c') - \mathcal{S}(c)\|$, is *not* in any way bounded by $\varepsilon$. Thus, the function $\mathcal{S}$ is not Cauchy continuous. Note that this fact is not dependent on sampling issue related to the shape or the skeleton, such as sampling density or sampling model (volumetric or boundary-based).

This inherent discontinuity of the skeletonization function, or operator $\mathcal{S}$, is often seen in the literature under various (empirical) names such as sensitivity to noise, generation of ‘spurious’ artifacts, or instability. Beyond the theoretical discontinuity issue pointed above, this phenomenon causes relevant practical problems: For instance, we cannot guarantee that we obtain ‘similar’ skeletons (under a Hausdorff distance metric or similar) even if our input shapes are similar. Therefore, subsequent uses of skeletons, e.g. in shape processing or matching applications, are difficult at best.

This long-known issue in skeletonization has been addressed by a so-called *regularization* process. In detail, the aim is to *either* pre-process
the shapes to be skeletonized, or post-process their skeletons, in a way that Cauchy-Weierstrass continuity is ensured – thus, practically, we can ensure that small changes of a shape result to small changes of its computed skeleton.

Several such regularization method exist, as follows:

### 2.3.4.1 Input preprocessing

The intuition behind these methods is that skeleton branches (details) are inherently created by curvature maxima on the input shape boundary $\partial \Omega$ (see Section 2.2.2. As such, if we are able to decrease the curvature range of $\partial \Omega$, we will in turn decrease the variability (and in the end, discontinuity) of the skeletons of $\Omega$. This can be done by applying a wide range of smoothing filters on $\partial \Omega$, see e.g [176]. The problem with this approach is that we need to carefully control the filter parameters to make sure that ‘relevant’ shape details are kept, whereas ‘irrelevant’ details are removed. While we could do this from the strict perspective of the shape $\partial \Omega$, is is not evident how smoothing operations on this shape will affect its skeleton.

A different approach to input preprocessing is to apply a global operation to all points of $\partial \Omega$, in such a way that this operation is guaranteed to remove unwanted skeleton details. Such an operation is proposed by the so-called scale axis transform [59, 108]. Intuitively, this operation inflates the boundary $\partial \Omega$ isotropically in normal-to-boundary directions, thereby making small-scale boundary perturbations either smooth out or disappear. This is indeed expected, since this inflation is very similar to computing level-sets (isocontours) of the boundary’s distance transform $DT_{\partial \Omega}$ at increasingly higher isovalues. As the isovalue increases, such contours are increasingly closer to a ball, thus to a shape of constant curvature, or a shape whose skeleton is simpler (in the limit, the skeleton of a perfect ball is a single point). While this method works very well in practice (see e.g. [108]), the setting of the inflation factor is not evident for an end-user. That is, it is not obvious how much to inflate a given shape if we want a certain amount of its skeletal detail to be removed. Moreover, by changing the input shape this way, we cannot guarantee that certain important shape properties, such as topology, are preserved.

### 2.3.4.2 Local Metrics

In contrast to input preprocessing, local metrics take a different approach: The input shape $\Omega$ is kept as is, and its skeleton $S_{\partial \Omega}$ is computed by any desired skeletonization method. Next, a regularized, or simplified, skeleton $S_{\partial \Omega}^{\tau}$ is computed, subject to a user-supplied simplification parameter $\tau$.

The key aspect of local simplification metrics, i.e. functions that compute $S_{\partial \Omega}^{\tau}$ from $S_{\partial \Omega}$ subject to a given $\tau$, is that they act locally. That is, the value of $S_{\partial \Omega}^{\tau}(x)$ depends only on the immediate neighborhood of the
point \( x \) of the skeleton. Local measures include the angle between the feature points and distance-to-boundary [2, 50], divergence-based [16, 155] and first-order moments [140]. Leymarie and Kimia topologically simplify point-cloud skeletons to capture Y-intersection curves and skeleton sheet boundaries in medial scaffolds, and next apply similar local skeleton detectors [93].

The main advantage of such regularization approaches is their simplicity and computational efficiency: Indeed, if the cost of regularizing a point \( x \) depends only on a fixed-size spatial neighborhood of \( x \), it means that the cost of regularizing an entire skeleton \( S_{\partial \Omega} \) is linear in the size (e.g., number of sample points) of this skeleton. Moreover, the regularization procedure can work, ideally, in parallel, since every point \( x \in S_{\partial \Omega} \) is treated independently.

However, such metrics have the fundamental problem of not being able to distinguish between locally-identical, yet globally-different, contexts (see e.g. [135], Figure 1). As such, thresholding local metrics can disconnect skeletons; reconnection needs extra work [103, 122, 155, 168]. This also makes skeleton simplification, also called skeleton pruning, and makes pruning, less intuitive [148].

2.3.4.3 **Global Metrics**

In contrast to the local regularization metrics discussed earlier, global regularization metrics pose the problem differently: Given a point \( x \in S_{\partial \Omega} \) on an unsimplified, raw, skeleton, how can we measure the overall importance of \( x \) to the shape \( \Omega \)?

A key development in this direction has been the work of Dey et al. [37], similar to earlier work of [125]. Although the key motivation behind this work has been the exact definition of a curve skeleton in three dimensions, their results can (and have been) used to globally regularize skeletons. In detail, given a ‘raw’ skeleton point \( x \in S_{\partial \Omega} \), the MGF metric assigns to \( x \) the longest shortest-path distance, over \( \partial \Omega \), between the any two feature points of \( x \). As shown by numerous works [37, 129–131, 135], the MGF metric assigns a ‘natural’ importance which is lower for skeleton points \( x \) created by minute local variations of the shape surface \( \partial \Omega \), and larger for more important variations. More interestingly, for the vast majority of shapes, the MGF measure seems to increase monotonically from the skeleton boundary \( \partial S_{\partial \Omega} \) to the surface-skeleton center. The observation of this phenomenon has been consistently verified in practice on a large number of different shapes (e.g. [37, 131, 134, 135]. As such, the MGF metric allows an easy way to compute a so-called multiscale skeleton representation \( S_{\tau \partial \Omega} \); simply by upper-thresholding all points of the full skeleton \( S_{\partial \Omega} \) whose MGF value exceeds \( \tau \). If the MGF metric is indeed monotonically increasing from \( \partial S_{\partial \Omega} \) to the center of \( S_{\partial \Omega} \), for any shape \( \Omega \), this means that the above thresholding will deliver increasingly smaller and simpler, and in the same time connected, skele-
tons. Interestingly, the same MGF metric has been also used to extract multiscale curve skeletons [135].

An important analogy with the MGF metric can be found in the 2D case. Here, much earlier than studying the 3D case, a very similar regularization principle has been proposed for 2D medial axes. Interestingly, the same 2D regularization principle has been proposed independently by several researchers for both volumetric-sampled or boundary-sampled shape descriptions [33, 118, 183]. This principle can be compactly described as follows: Consider a 2D skeleton point \( x \). By definition, \( x \) will have at least two feature points on the boundary \( \partial \Omega \) of its corresponding input shape. The shortest distance (along the boundary \( \partial \Omega \)) between such feature points gives a natural and intuitive ‘importance metric’ or value for the skeleton point \( x \). By upper-thresholding this importance metric by some user define value \( \tau \), we can obtain a 2D skeleton where all boundary details longer than \( \tau \) are captured. As such, reconstructing the input shape from the simplified skeleton \( \Sigma^{\tau}_{\partial \Omega} \) guarantees that we obtain a shape \( \Omega^{\tau} \) where all details of \( \Omega \) whose length was smaller than \( \tau \) have been replaced by circle arcs. For the 2D case, the monotonicity of the importance metric (from the 2D skeleton tips to its center) can be easily proven. As such, thresholding this metric delivers nested and connected skeletons. Intuitively, the above metric can be thought as assigning to each point \( x \in \Omega \) the amount of mass, or boundary length, that would collapse to \( x \) if advected in a flow field determined by \( \nabla DT_{\partial \Omega} \). The same intuition is mentioned – though not formally argued – behind the 3D MGF metric by Reniers [135]. Variations of this so-called ‘collapsed boundary length’ metric are known, e.g. [177]. These variations ensure that simplified skeletons accurately capture spatially important 2D boundary corners, while neglecting unimportant ones. Using the shape-from-skeleton reconstruction process outlined in Section 2.2.1, these techniques are competitive alternatives to anisotropic smoothing methods which keep ‘salient’ shape features sharp, while fully removing less salient features. While this so-called saliency metric (essentially the 2D MGF metric divided by the value of distance-to-boundary) works convincingly well in 2D, no equivalent is known for 3D shapes.

2.4 CONCLUSIONS

In this chapter, we have presented a (necessarily limited) overview of the theory and current practice regarding computing 2D and 3D medial descriptors. As we have seen, computing (and using) such descriptors in practice is challenged by both theoretical and practical issues. On the theoretical side, we have the major issue of global (monotonic) regularization, i.e. the computation of a multiscale of medial descriptors that can guarantee the incremental capture of details of an input shape subject to a user-defined metric and metric threshold value. On the practical side, we see challenges in the fast and robust computation of medial sur-
faces (and, partially, the computation of curve skeletons) for large and complex 3D shapes. Additionally, we see challenges, but also interesting possibilities, in computing derived skeletal properties that help us to easily and robustly analyze and process such 3D shapes. Having now set the context, and having outlined the above-mentioned issues, the remainder of this thesis will focus on addressing these issues.