Multiscale Importance Metrics for Robust Skeletonization of 2D and 3D Shapes

M. Hofstra

Supervisors:
Prof. Dr. Alexandru C. Telea
Dr. M.H.F. Wilkinson

February 17, 2015

Abstract. We present a single generalized method for computing skeletons for arbitrary objects in both 2D and 3D space. We also present a method for computing the multiscale boundary-collapse importance metric for arbitrary 2D shapes in an incremental fashion, the results of which are comparable to the AFMM Star for 2D shapes. Finally, we reveal an approach which potentially allows the same importance metric to be computed for 3D shapes as well to provide results comparable to the Reniers et al.[33] method. The resulting method retains the simple implementation of the AFMM Star while avoiding the computational cost of the 3D boundary-collapse measure as computed by Reniers et al..

Contents

List of Figures

1 Introduction

Skeletons are powerful descriptors of both 2D and 3D shapes and are commonly dealt with in the field of scientific and medical visualization, computer-aided design and others. Applications areas of skeletons include path planning and shape simplification, reconstruction, animation and compression. Skeletons represent shapes in a compact and simple manner, notable in a way that preserves many characteristics of the shape.

Shapes contain features at varying scales of detail and in varying configurations. Large-scale features convey information that allows us to understand the shape at a high level (e. g. is the shape formed like an `X` or like an `O`), while small-scale features are not required for this and can even hinder this high-level understanding. Conversely, small-scale features can be important for applications in e. g. the field of medical visualization (contour roughness may indicate features of interest) or satellite image analysis (differentiating between rough natural shapes and the generally more regular shapes of human-made structures).

When the topology of a shape yields a hierarchy of such multiscale details, we desire that the skeleton of the shape yields the same. When the skeleton does, we say it yields a multiscale importance measure; one that distinguishes between unimportant (small-scale, local) and important (large-scale, global) features. We want such a measure to be monotonically increasing from the endpoints of the skeleton, meaning that when we traverse the skeleton from the boundary towards the inside of the shape, we will encounter only equal or greater values.

Computing the multiscale importance measure for the skeleton of a shape is however a difficult problem that is still a topic of ongoing research. One has to deal with inherent sensitivity to boundary noise as well as a high computational cost, especially for 3D shapes. Accuracy limitations, and in the case of binary voxelized skeletons also discretization artifacts, often introduce instabilities in the skeleton, leading to methods that compute an approximation of the skeleton.

There is currently only one known multiscale measure for 3D skeletons [33], but the method is complex and computationally expensive. A numbering scheme is used by Telea and van Wijk[39] to compute this importance for 2D shapes in an efficient manner, but there is no known numbering scheme that works for 3D shapes.

In this thesis, we consider the boundary distance importance measure which was first introduced by Ogniewicz and Ilg[30] and present a method which computes this importance measure in an incremental fashion.

1.1 Goal

The 2D AFMM Star method [39] is seen as a state-of-the-art method for computing multiscale skeletons for 2D shapes, and the Reniers et al.[33] method is considered the same for 3D shapes. The goal of this thesis is to investigate the feasibility of an alternative importance measure. In chapter 3 we propose such a method which:

- has a chance to generalize to 3D
- computes a measure comparable to the geodesic measure of Reniers et al.
- avoids the particular computational cost associated with computing geodesics

We will compare the results of this method qualitatively and quantitatively with the results of the AFMM / AFMM Star and Reniers et al. method.

1.2 Outline

In chapter 2 we provide definitions of shapes, skeletons and various related concepts and methods. In chapter 3 we present the foundation for this thesis: an approach to compute a multiscale importance measure identical to the 2D AFMM Star. We analyze several differences with the AFMM Star and construct our method in such a way that individual components:

- provide a solid framework to produce the desired result
- readily allow a generalization to the 3D case
- are easily replaced or modified in such a way that other components are unaffected (i. e. components are highly compartimentalized)
- are simple, efficient and verifiably correct

In chapter 5 we present our preliminary findings on implementing the generalized method in 3D. We also elaborate on interpretations supporting the idea that a method computing the desired metric in the desired manner is feasible. In chapter 6 we provide results of the implemented 3D skeletonization method. We summarize and discuss our results in chapter 7 and finally elaborate on future work in section 8.2.

2 Definitions and related work

In this chapter, we list definitions and properties for several well-known concepts (shape, boundary, skeleton) and methods (DT, FT) related to our work. We also introduce several notations that serve to express our work in terms of these concepts and methods.

2.1 Shape and boundary representation

A n-dimensional shape Ω is a subset of a continuous Euclidean $\mathbb{R}^n$ space. In this thesis, we consider only 2D and 3D shapes and therefor apply the restriction $n = \{2, 3\}$. The boundary of the shape is denoted as $\partial\Omega$ and is defined as the infinitely thin layer of points that wrap around the shape.

Cavities The number of *cavities* or *tunnels* in a shape is defined as its genus g. With regard to skeletonization, shapes of genus > 0 are significantly more difficult to deal with, especially for the 3D case. This is undesirable given the exploratory focus of this thesis, and as such, shapes of genus 0 are the main focus of our work.

2.1.1 Discretization

Representing a shape can be done in analytical form or in a discretized (sampled) form. Using an analytical form has the advantage of having an exact representation, but can be difficult to implement and can become costly in terms of computational time.

A discrete form is much simpler to implement and can more readily facilitate an efficient and accurate implementation. Two main forms of discretization exist: sampling the entire shape Ω (volumetric, typically by using a grid of pixels / voxels) or sampling only the boundary $\partial\Omega$ (typically by using a point-cloud / mesh representation).

Volumetric representations are generally slower than point-cloud / mesh representations while being simple to implement and argueably easier to parallelize. Since our main focus lies on exploring a new concept, it makes sense to favor simplicity over performance. As our work is an extension of the AFMM [39] / AFMM Star [32], which uses a volumetric representation, we will use this representation as well.

A shape Ω can be represented in a discrete grid of binary elements. Each element is a pixel (2D) or voxel (3D) and yields a value of 1 if it represents the interior of the shape and a value of 0 otherwise. The boundary $\partial\Omega$ can then be represented as the points with value 1 which are adjacent to a point with a value of 0. We formalize this

in eq. (2.1):

$$\partial\Omega = \Big\{ p \in \Omega \mid q \notin \Omega \wedge q \in N_{\perp}(p) \Big\}$$
$$\text{where } \perp = \begin{cases} 4 & \text{if } \Omega \in \mathbb{R}^2 \\ 6 & \text{if } \Omega \in \mathbb{R}^3 \end{cases} \tag{2.1}$$

Here, $N_{\perp}(p)$ depicts the points that are edge-neighbours (2D shapes) or face-neighbours (3D shapes) of p. The boundary $\partial\Omega$ of Ω is of a dimensionality that is one lower than the dimensionality of Ω, i. e. if $\Omega \in \mathbb{R}^n \Rightarrow \partial\Omega \in \mathbb{R}^{n-1}$. We denote $N_{\perp}^{\partial}(p)$ for the boundary points that are adjacent to p, which translates to those points $q \in \partial\Omega$ which are vertex-neighbours (1D boundaries of 2D shapes) or edge-neighbours (2D boundaries of 3D shapes) of $p \in \partial\Omega$.

2.2 Distance transform

The distance transform (DT) is a measure that assigns to each point in the shape the minimum Euclidean distance to the boundary:

$$D(p \in \Omega) = \min_{k \in \partial\Omega} ||p - k|| \tag{2.2}$$

Where $||\cdot||$ denotes the Euclidean distance. For this thesis, we use the Fast Marching Method (FMM) to compute the distance transform.

2.2.1 Fast Marching Method

The FMM is a method that solves the Eikonal equation (eq. (2.3)) on a discrete grid by progressively propagating a virtual wavefront towards the center of the shape with a constant speed. The FMM computes for each point in the shape the wavefront arrival time T.

Similar to Dijkstra's algorithm [11], the FMM relies on the fact that information can only flow away from the "source" (hereafter *minpoint*) and never towards it. Initially, the wavefront (hereafter *narrowband*) coincides with the boundary of the shape. By iteratively removing a minpoint from the narrowband, the FMM ensures that the arrival time is never *under*estimated. The arrival time of the minpoint is then used to correct overestimated arrival times adjacent points. Since every point in the shape will eventually become a minpoint, this process ensures a correct arrival time for all points.

$$\big|\nabla T\big| = 1 \tag{2.3}$$

The arrival time T is a good estimator for the distance to the boundary, which means that effectively the FMM can be used to compute the distance transform. Initially, each point in the shape is assigned a particular flag and distance value as follows:

- `KNOWN`: points that are not part of the shape. These points lie *behind* the wavefront / narrowband, their T-value is initialized as 0

- `NARROWBAND`: points on the boundary of the shape. The T-value of these points is initialized as 1
- `UNKNOWN`: points that are part of the shape, but not on the initial boundary. The T-value of these points is initialized to some unattainably high value (e. g. an equivalent to $+\infty$)

The flagvalue $p^{\{\mathrm{K}|\mathrm{N}|\mathrm{U}\}}$ of a point p thus conveys how to interpret its current T-value:

- `(K)NOWN`: the value is final and accurate
- `(N)ARROWBAND`: the value is temporary and possibly over-estimated
- `(U)NKNOWN`: no estimation has been made yet

This interpretation is invariant during the FMM execution and formalized as:

$$T(p^{\mathrm{K}}) \leq T(p^{\mathrm{N}}) < T(p^{\mathrm{U}}) \tag{2.4}$$

Processing order

Each of the points on the initial boundary is initialized with the same arrival time $T = 1$. Formally, any one of these points qualifies as a valid minpoint which we can process next, although processing them in any particular order can lead to subtle discrepancies in the computed DT and FT. In order to separate these discrepancies from the main focus of this thesis, we present a notation in the form of a processing sequence. Effectively, we care only for *which* point to process next and we ignore *how* this point was selected as such.

For a shape Ω containing n points p_t with $t \in [0 \dots n\rangle$, the FMM will remove points from the narrowband in a sequence of n timesteps. These timesteps are atomic: all calculations for a particular timestep are to be completed in order to proceed with the next timestep. We denote $\partial^{\mathrm{N}}\Omega_t$ for the state of the narrowband at timestep t and we can express consecutive states of the narrowband as follows:

$$\partial^{\mathrm{N}}\Omega_{t+1} = \begin{cases} \partial\Omega & \text{if } t < 0 \\ \partial^{\mathrm{N}}\Omega_t \setminus p_{t+1} & \text{if } 0 < t < n \\ \emptyset & \text{if } t \geq n \end{cases} \tag{2.5}$$

We now express the flagvalue of points as a function of the current timestep t. Since we know that $0 \leq t < m < n$, we get:

$$\mathit{flag}(p_s) = \begin{cases} \texttt{KNOWN} & \text{if } s \leq t \\ \texttt{NARROWBAND} & \text{if } t < s < m \\ \texttt{UNKNOWN} & \text{if } m \leq s \end{cases} \tag{2.6}$$

which we can then use to express eq. (2.4) as a function of timesteps:

$$T(p_a) \leq T(p_b) < T(p_c) \quad \text{where} \quad 0 \leq a \leq b < c < n \tag{2.7}$$

Effectively, eqs. (2.5) to (2.7) are derived from eq. (2.4) and are in a form that is more convenient for expressing the presented approaches.

2.3 Feature transform

The feature transform (FT) can be seen as an extension of the distance transform. Whereas the DT computes the *distance* to the nearest boundary point, the FT determines the actual *position* of such a boundary point. If we uniquely label points on the boundary of a shape, then we can compute the distance transform using the FMM while also tracking the labels of the nearest boundary points (feature points). It is difficult to guarantee that *all* feature points are identified, but for the purpose of this thesis, a complete set of feature points is not essential.

In the continuous case, points a, b are the *feature points* $F(p)$ of $p \in \Omega$ and the feature transform is defined as:

$$F(p \in \Omega) = \left\{ q \in \partial\Omega \;\middle|\; ||q - p|| = D(p) \right\} \tag{2.8}$$

For the discrete case we denote $b \in B(p)$ as the closest boundary points $b \in \partial\Omega$ of $p \in \Omega$. In theory, $B(p) \equiv F(p)$. In practice, such a strict definition complicates the implementation, which has to deal with discretization artifacts and accuracy limitations. As such, for the discrete case we assume eq. (2.9) instead.

$$B(p) \rightsquigarrow F(p) \tag{2.9}$$

2.4 Skeleton

The skeleton is a well-known descriptor for the geometry of a shape. Generally, the skeleton is centered within the shape, represents the geometry and topology of the shape and is of a lower dimensionality. In the continuous case, the Blum skeleton $\mathcal{S}$ of a n-dimensional shape $\Omega \subset \mathbb{R}^n$ with boundary $\partial\Omega$ is defined as the points $p \in \Omega$ for which at least two distinct points $a, b \in \partial\Omega$ exist that lie at a minimum distance to p. The Blum skeleton [2] can be seen as a mapping of a n-dimensional shape onto a $(n-1)$-dimensional shape:

$$\mathcal{S} = \mathcal{S}^{n,n-1}(\Omega) = \left\{ p \in \Omega \;\middle|\; \exists a, b \in \partial\Omega, a \neq b, ||p - a|| = ||p - b|| = D(p) \right\} \tag{2.10}$$

where $D : \Omega \to \mathbb{R}_+$ is the distance transform (section 2.2) and a, b are the *feature points* of p, computed using the feature transform.

Effectively, a 3D shape yields a 2D skeleton, a 2D shape yields a 1D skeleton and a 1D shape yields a single point as the 0D skeleton. In this thesis, we consider only 2D and 3D shapes and their corresponding 1D respectively 2D skeletons, due to which the dimensionality of skeletons is restricted to $n \in \{0, 1, 2\}$.

2.4.1 Anatomy

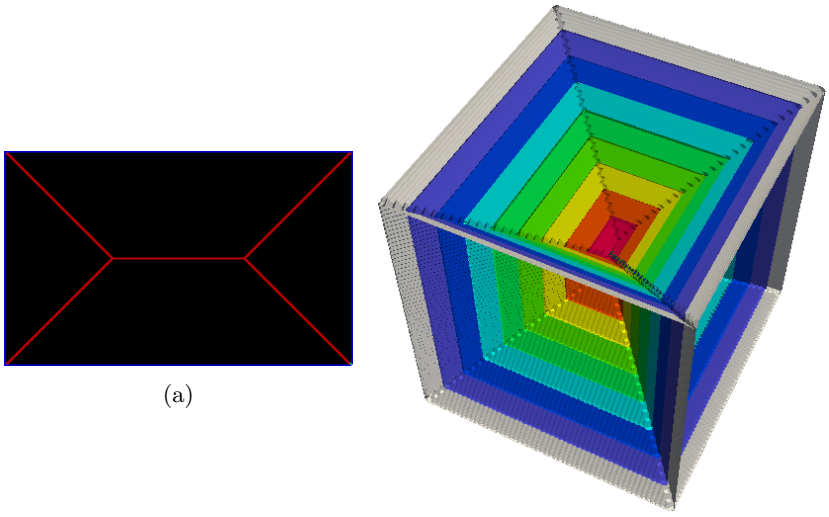
Analogous to the dimensionality of the input shape, one can distinguish between several types of skeleton points. A n-dimensional shape will yield skeleton points of the $(n-1)$-dimensional type. A subset of these $(n-1)$-dimensional skeleton points is the set of $(n-2)$-dimensional skeleton points. This pattern continues until the skeleton is a single (0-dimensional) point. Conversely, a skeleton point is thus always a subset of a relatively higher dimensional skeleton.

Regardless of the particular dimensionality, each skeleton point in a segment is exactly one of three particular subtypes. Additionally, the subtype of a n-dimensional skeleton point is related to the subtype of the $(n \pm 1)$-dimensional skeleton points at the same position.

In the following sections, we enumerate these types, subtypes and their properties. We also detail on other particular properties of types and subtypes in relation to the shape, boundary, DT and FT.

Main type Skeleton points of a particular d-dimensional type can be represented as a collection of one or more d-dimensional shapes. For example, the 1D skeleton of a 2D `X` shape consists of two crossing 1D segments (lines). When we stack many 2D `X` shapes (residing in the xy plane) in the z-direction to form a 3D `X` shape, the resulting skeleton is 2D and consists of two crossing 2D segments (planes).

Since the scope of this thesis is limited to 3D shapes, we only need to consider the skeletons of dimension $n \in \{0, 1, 2\}$. We identify the *surface skeleton* (SS, denoted by $\mathcal{S}^{\mathcal{S}}$) as the 2D skeleton, the *curve skeleton* (CS, denoted by $\mathcal{S}^{\mathcal{C}}$) as the 1D skeleton and the *skeleton root* (SR, denoted by $\mathcal{S}^{\mathcal{R}}$) as the 0D skeleton.



(a) (b)

Figure 2.1: Skeleton examples for rectangular and cubical shapes

Subtype Regardless of the particular dimensionality, each skeleton point in a segment is exactly one of three particular subtypes (ignoring corner cases). When we consider a 1D line segment, we distinguish between the two *endpoints* $\mathcal{S}_{\mathcal{E}}$ of the line and the remaining *inner points* $\mathcal{S}_{\mathcal{I}}$. When two 1D line segments meet, the points that lie at the crossing are distinguished as *junctions* $\mathcal{S}_{\mathcal{J}}$. The 0D skeleton is considered a special case and relates to the genus of the shape (section 2.1): the skeleton of a shape of genus g yields $g+1$ skeleton *root points*. Since the work in this thesis focuses only on shapes of genus 0, we only need to consider a single root skeleton point per shape.

The subtype of a skeleton point relates closely to the dimensional hierarchy of the main types, but the order of inheritance is inverted. Consider a 3D shape, its 2D skeleton and the 1D and 0D skeletons contained within. We can now formulate several relations between the different dimensionalities that detail how a skeleton is "nested" in a skeleton of higher dimensionality.

A point on the n dimensional skeleton is always a junction on the $(n+1)$ dimensional skeleton ($p \in \mathcal{S}_{\mathcal{E}}^{n,n-1} \implies p \in \mathcal{S}_{\mathcal{E}}^{n+1,n}$). Additionally, endpoints on the n dimensional skeleton are always endpoints on the $n+1$ dimensional skeleton ($p \in \mathcal{S}_{\mathcal{J},\mathcal{I}}^{n,n-1} \implies p \in \mathcal{S}_{\mathcal{J}}^{n+1,n}$).

Conversely, inner points on a $(n+1)$ dimensional skeleton are not in the n dimensional skeleton ($p \in \mathcal{S}_{\mathcal{I}}^{n,n-1} \implies p \notin \mathcal{S}^{n-1,n-2}$), and endpoints on the $(n+1)$ dimensional skeleton are not on the n dimensional skeleton unless the n dimensional skeleton point at the same position is also an endpoint ($\exists p \in \mathcal{S}_{\mathcal{E}}^{n,n-1} : \mathcal{S}_{\mathcal{E}}^{n-1,n-2}$).

2.5 Desirable properties

We summarize some well-known quality criteria [7] describing desirable properties for skeletons. We differentiate between the intrinsic properties (objective measures) that follow from the definition of the Blum skeleton (eq. (2.10)) and the extrinsic properties (subjective metrics) which relate to the requirements of applications (and/or their implementations) using the computed skeleton. Many of these definitions were taken from recent work [35, 36].

Ideally, many of these properties follow directly from the skeletonization process without requiring "parameter tweaking" or applying a post-processing step. This is however not always feasible as some properties conflict with one another by definition.

2.5.1 Intrinsic properties

The following properties relate to formal properties of the skeleton. These properties are easily verifiably and can be objectively measured.

Homotopical / topological equivalence The skeleton must be topologically equivalent to the input shape, i. e. both the skeleton and the shape yield the same number of connected components, cavities and tunnels.

Thin The skeleton should be as thin as possible given the sampling model, and it should be of a dimensionality that is one lower than the input shape.

Centered For the surface skeleton, centeredness follows from eq. (2.10). For curve skeletons, no such unique centeredness definition exists. In this thesis, we apply a useful weak form of curve-skeleton centeredness which states that since the surface skeleton is defined as the subset of points centered in the shape, the curve skeleton must be a subset of the surface skeleton [9].

Smooth The surface skeleton should have a continuous second-order derivative. Since the curve skeleton is assumed to be a subset of the surface skeleton, each individual branch of the curve skeleton should have a continuous second-order derivative as well.

Regular multiscale representation Input shapes can yield structures at varying levels of scale. For any given scale, such details should be representable by the skeleton and should map back onto the original shape. This mapping should be hierarchical and monotonic: the skeleton that represents the shape at a coarse scale should be a subset of a skeleton that represents the shape at a finer scale. Noise in the input shape should be considered as small-scale structures and should therefor map onto small-scale structures in the skeleton. This implies that pruning small-scale structures from the skeleton using some threshold value directly relates to removing the corresponding small-scale structures in the input shape.

Transformational invariance Disregarding the discontinuities that result from using a discrete representation, the skeleton should be invariant to isometric transformations of the shape. Since these transformations do not change the topology of the shape, it follows directly that the topology of the skeleton should not change as well.

2.5.2 Extrinsic properties

The following properties relate to practical properties for implementations of skeletonization methods. These properties are more abstract or high-level than intrinsic properties and are therefor somewhat subjective metrics.

Sampling robustness If we resample a shape, the difference between the skeleton of the original shape and the skeleton of the resampled shape should be proportional to the sampling differences between the input shapes.

Detail preservation The skeleton should be able to represent small details relating to discontinuities on the boundary of the input shape. This is effectively another form of robustness: the skeleton should tolerate all kinds of details (e. g. discretization artifacts).

Scalability The skeleton of large shapes (1024^3 or similar) should be computed efficiently in terms of time and memory usage and, ideally, should readily allow a parallelized implementation yielding an acceptable, preferrably near-linear speedup factor.

Implementation complexity When the implementation of a skeletonization method is perceived as a "black box" or a "big ball of yarn" or is hard to replicate, tune or improve, the usefulness of the method will be reduced significantly. Effectively, the implementation of a skeletonization method should also optimize for the time and effort of application developers.

2.6 Skeletonization methods

In this section we discuss several techniques and approaches that are commonly used in skeletonization methods.

2.6.1 Thinning

Thinning, also called *pruning*, *erosion* or *boundary peeling*, is a process which aims to retain topological properties while removing points from the shape. Points that can be removed from the shape in this manner are called a *simple points*. By iteratively removing simple points from the shape, the shape is reduced to a thin structure in which all remaining points are skeleton points. All shapes of genus 0 will be reduced to one single point however. This is generally avoided by identifying endpoints of the skeleton and ensuring these are never classified as simple points.

Thinning is typically applied to volummetric representations and is generally simple to implement. It is hard however to fine-tuning thinning methods towards producing the desired result. A comprehensive survey on skeletonization methods by thinning is provided by Lam et al.[23].

2.6.2 Geometric

Geometric methods use mesh / boundary representations and generally scale well for large shapes. Some examples are Voronoi-based methods [10], mesh contraction in normal direction [1, 38, 4, 25], mean-shift-like clustering [16] and union-of-ball approaches [28, 26, 17]. A qualitative comparison for several such methods was recently conducted by Sobiecki et al.[35].

2.6.3 Field based

The skeleton of a shape can be defined by the singularities in the Euclidean distance-to-boundary field (the distance transform). When we represent the distance field using a height map, the result looks like a mountain for which all slopes have identical steepness. Slopes run from the boundary of the shape (which is at ground level) towards the ridges on top of the mountain. These ridges are the local maxima

which represent the skeleton of the shape. Methods which operate on the Euclidean distance transform are called *distance field* methods [24, 21, 13, 40, 34, 15] and can be implemented efficiently on GPUs [37, 5].

A problem for distance field methods is that the distance field does not yield enough information to extract the skeleton from a shape. Another class of methods that work around this issue are *general field* methods. These methods use fields that are smoother than the distance transform, containing less singularities. These methods are more robust towards boundary noise as a result. By considering various additional measures, i. e. the angle between feature vectors [12, 37] or the geodesic distance between points [9, 33], undesirable consequences of boundary noise can be reduced to improve the detected skeleton.

2.7 Regularization

In the continuous case, skeletons are sensitive to small pertubations on the surface. This is also true for the discrete case, as it is only a sampled derivative of the continuous case. These pertubations are often numerous and generate many of small skeleton branches, which we generally consider as noise. One reason for this is that these "noisy" branches are unstable: they disappear entirely in response to very minimal changes on the surface. This is generally undesirable, which is why we require a method to detect and eliminate such instabilities.

The process of eliminating noisy elements is called *regularization*. In this section, we detail on several classes and instances of importance measures for the skeleton of a shape.

2.7.1 Importance measure

Importance measures enable pruning strategies to selectively prune the skeleton of a shape as a function of some pruning parameter τ. Pruning skeleton points with a importance value smaller than τ removes spurious skeleton branches that are the result from boundary noise. The importance value of a skeleton point or branch ideally expresses the relevance of the associated portion of the shape. We can distinguish two classes of measures: local and global measures, which we will discuss next.

Local measures

Local measures take into account only the points in a certain neighbourhood around the point for which the measure is computed. Examples of local measures are the Euclidean distance[8], first-order moments[6] or the angle between the feature vectors[3]. A typical property of local measures is that they cannot distinguish between configurations that are locally identical, even if these configurations differ in a global context. As a result, using a local measure to prune the skeleton using a threshold value can result in a disconnected skeleton.

Although local measures are simple to implement and generally also easy to parallelize, their usefulness is limited due to these issues. As discussed below, global measures are a more sensible choice in our case.

Global measures

The key advantage of using a global measure is that global measures can differentiate between the cases which local measures would consider identical. This enables computing a multiscale measure for the skeleton that increases monotonically from the endpoints of the skeleton towards the center. Such measures can be thresholded to yield a connected skeleton, which makes global measures preferable to local measures, despite global measures being significantly more expensive to compute.

Two methods exist in the class of global measures: The AFMM Star[39] and the Reniers et al.[33] method, which are considered as state-of-the-art methods for computing multiscale skeletons for 2D and 3D shapes respectively. We detail on these methods in the following section.

2.8 Boundary-distance measure

The boundary-distance measure was introduced by Ogniewicz and Ilg[30] and assigns to each point p in the shape the minimum distance over the boundary of the shape between feature points of p. This is equivalent to collapsing the corresponding section of the boundary onto p. The resulting measure for a given skeleton point p relates to the size of the boundary features that are related to (collapsed onto) p, translating to low values for small boundary features and larger values for large boundary features.

As described in the following sections, two methods exist which compute the boundary-distance measure for 2D and 3D shapes respectively: The AFMM Star method [39] and the Reniers et al. method. These methods represent the core of our work in which we attempt to unify both methods.

As presented by Reniers et al.[33], the boundary-distance measure can be interpreted as the result of an advection process, which we discuss in section 2.8.3.

2.8.1 AFMM Star

The AFMM Star applies a numbering scheme U to the initial boundary $\partial\Omega$ and uses the Fast Marching Method to propagate U along with the computed distance T. By exploiting the topological properties of 2D shapes, the AFMM Star is able to produce the desirable results of a global measure even though it is technically using a local measure.

The AFMM Star traverses the boundary $\partial\Omega$ of the shape Ω and assigns to the n boundary points $p \in \partial\Omega$ a monotonically increasing integer value U_i where $i \in [0 \dots n\rangle$. During the traversal, the Euclidean distance between consecutive boundary points b_i, b_{i+1} is computed $(0 < i + 1 < n)$, added to the sum of distances computed so far and consecutively assigned to the current boundary point as a floating point value U_f

(eq. (2.11)).

$$U_f(b) = \sum_{i=0}^{b<n} ||b_i - b_{i+1}|| \tag{2.11}$$

An example is shown in fig. 2.2, U_i (left, right) and U_f (middle).

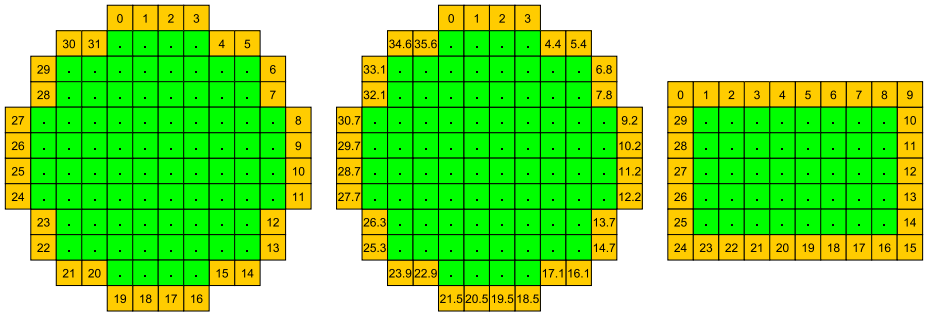


Figure 2.2: AFMM Star boundary initialization (U-values)

One can now easily verify that it is trivial to compute the minimum distance over the boundary between any pair of boundary points. For example, points 7 and 24 on the circle yield distance values 7.8 and 27.7. Since we know the length of the entire boundary, we can cheaply compute both options as $(27.7 - 7.8) = 19.9$ and $(35.6 - 27.7) + 7.8 = 15.7$, and simply choose the minimum of these two. This is exploited by the AFMM Star, effectively reducing the cost of computing a geodesic to $O(1)$.

Formally, we can say the 2D case "works" because for any skeleton point p all closest boundary points $B(p)$ lie on the same plane, and the U field is a planar "mold" which can be constructed efficiently for computing the required geodesics in $O(\Omega)$ time.

Unfortunately, no known equivalent numbering scheme exists for 3D shapes, limiting the application of the AFMM Star to the 2D case.

2.8.2 Reniers et al.

The Reniers et al. method [33] extends the boundary-distance measure to 3D by using Dijkstra's algorithm [11] to replace the AFMM Star's U field. The method first computes the FT of the shape, applying Dijkstra's algorithm to compute the shortest path between the feature points of $p \in \Omega$. This results in a computed multiscale importance metric for the entire shape, which reveals the 2D surface skeleton after thresholding with a parameter τ. This parameter also allows further simplification of the skeleton, similar to the AFMM Star.

After extending $F(p \in \Omega)$ to include all feature points found in a $2 \times 2 \times 2$ neighbourhood of p, the geodesics associated with the surface skeleton are combined to form Jordan curves, which are curves that wrap around the shape and divide the boundary

of the shape into two parts. A point on the surface skeleton which yields such a Jordan curve is then classified as a curve skeleton point.

The Reniers et al. method is currently the only known method that computes a multiscale importance metric for the skeleton of a 3D shape. Although a complete formal proof of correctness is not given, in practice the method produces results that are reliable and robust. The implementation is however complex and the computational cost of identifying all the geodesics is significant.

2.8.3 Advection model

The idea to interpret the boundary-distance measure as the result of advection model was first introduced by Reniers et al.[33] and is an interpretation that is key to this thesis. The idea behind this relation is that the distance over the boundary between the individual feature points of a certain skeleton point p corresponds with the result of an advection process. More precisely, the points on the surface of the shape that lie on the geodesic between feature points of p can be said to "flow through" p. Since the skeleton can be interpreted as a tree-like structure, a flow can be defined from the surface all the way towards the root of the tree.

Effectively, points on the surface flow towards the nearest surface skeleton point. Analogous to the skeleton anatomy in section 2.4.1, the flow of mass is directed from the surface skeleton towards the nearest curve skeleton point and finally towards the skeleton root. Such a flow yields a monotonically increasing amount of mass flowing through subsequent skeleton points.

Relation to our thesis The advection model suggests that it is possible to compute the desired importance measure in an incremental fashion. By definition, the endpoints of the skeleton relate to boundary features which lie in close proximity to each other. This means that we could use a local measure to compute the initial importance value for the endpoints of the skeleton. We can then increment these importance values by simultaneously traversing the boundary of the shape and the skeleton of the shape, provided that we can find the correct order of traversal.

3 Proposed method

In this chapter, we introduce our proposal for computing a multiscale skeleton (2D and 3D) using a volumetric representation (pixels / voxels). The main focus of our proposal is to compute a multiscale skeletal importance measure, similar to the AFMM Star (2D) [39] and the Reniers et al. method (3D) [33]. The core of our added value is computing these measures simply and efficiently, using a method that readily generalizes to the 3D case. Additionally, our method builds upon the Fast Marching Method (FMM) in a way that retains the FMM advantages of using different speed functions and / or applying early distance based termination. This enables early termination based on the computed importance measure, which would allow an efficient and reliable smoothing method for large shapes.

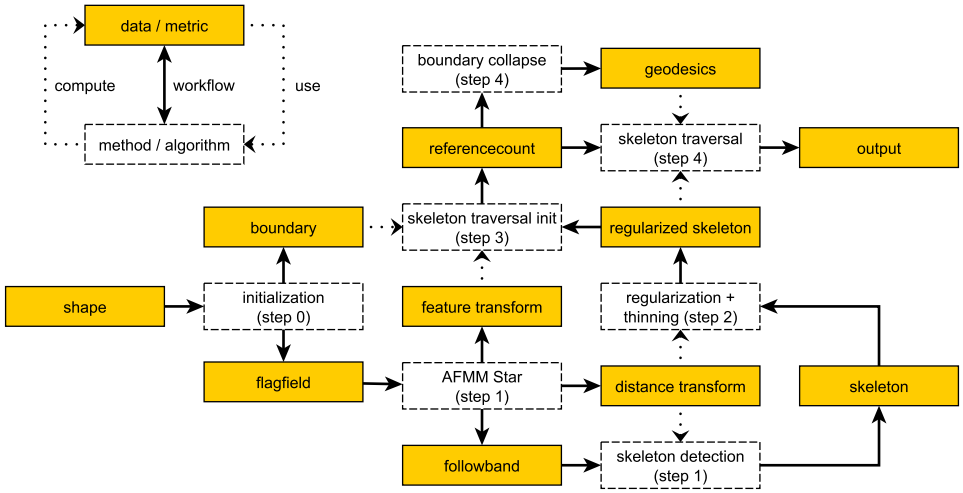


Figure 3.1: Workflow / dependency overview of our proposed method

3.1 Overview

In this section we provide an overview of our proposal in several forms. The core of this overview is a breakdown of our method into several steps, each of which describes the purpose and approach at a high level. The output of these steps and phases is visualized in figs. 3.3a to 3.3f. An overview at a lower level is given in the form of a

workflow / dependency overview and an algorithmic flowchart. The steps are grouped in two separate phases, the purpose of which will become more clear in chapter 5.

The flowchart in fig. 3.1 details the dependency chain between the methods used and the measures they compute and details how the order of computation relates to these dependencies. Starting at the input node, the solid lines detail the path towards the output node while the dashed lines convey where measures are computed and where they are used. The purpose of this flowchart is to provide a high-level overview of the manner in which the individual steps are intertwined.

The algorithmic flowchart in fig. 3.2 highlights several key steps and conditionals in the actual implementation of our method. This flowchart corresponds directly to the individual steps that together form our method.

Finally, fig. 3.4 contains an overview that contains examples for several of the following sections. This image uses a simple input shape to visualize how the methods in this chapter are intertwined.

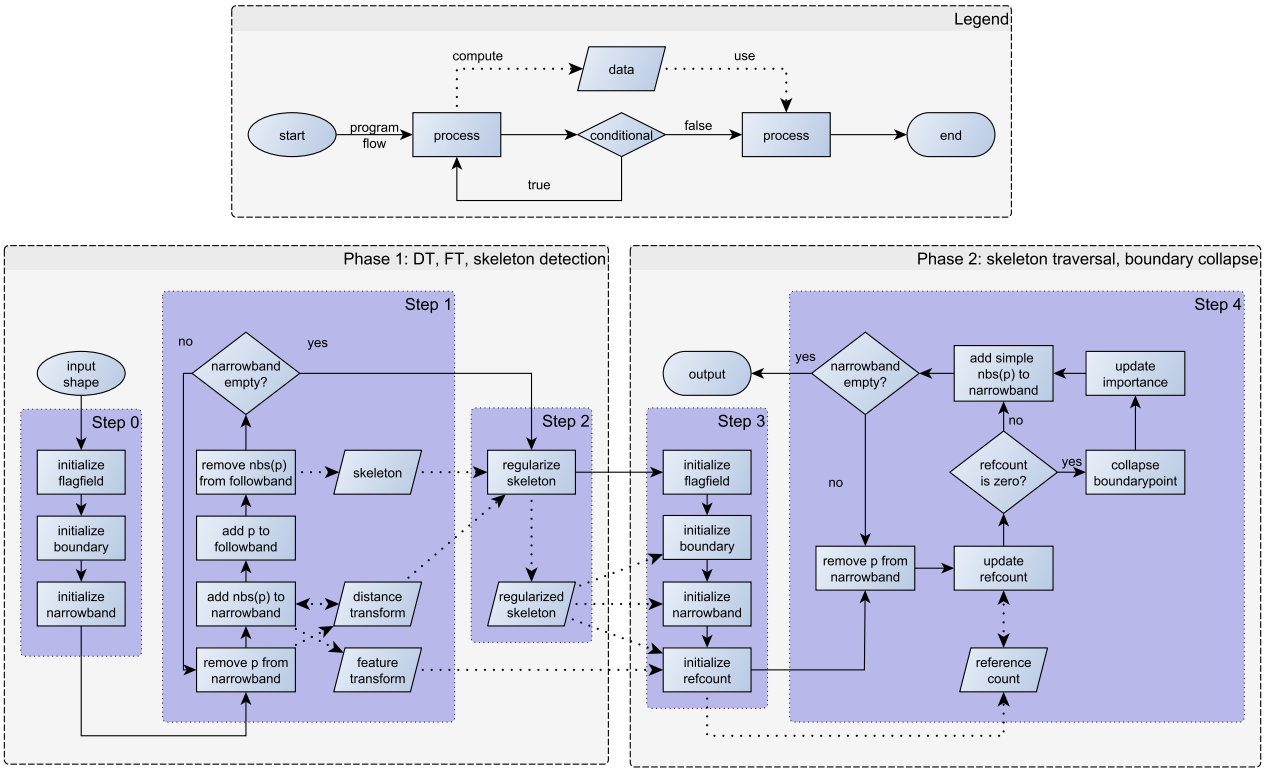
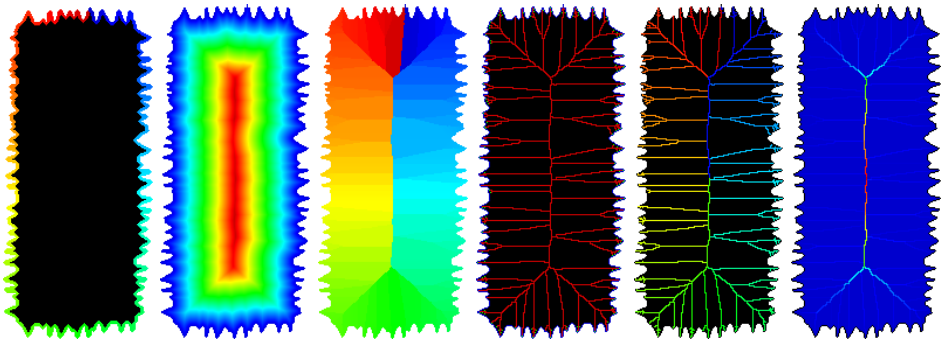


Figure 3.2: Algorithmic flowchart of our proposed method



(a) (b) (c) (d) (e) (f)

Figure 3.3: Stepwise depiction our method. Boundary labels (a), DT (b), FT (c), regularized skeleton (d), skeleton FT (e), boundary-collapse measure (f)

3.1.1 Phase 1

This phase contains the steps (0,1,2) which compute the DT, FT and identify the skeleton. This phase can be summarized as an variation on the AFMM Star without the boundary-collapse measure part.

Step 0: Initialize flags, narrowband, boundary (section 3.2) Identical to the FMM approach, we initialize the flags for the shape and boundary and add boundary points to the initial narrowband. We model the boundary of the shape by computing an adjacency mapping between boundary points, the result of which is very comparable with the AFMM U-field approach.

Step 1: Compute DT+FT, detect skeleton (section 3.4.2) The main purpose of this step is to compute a skeleton that we can readily interpret as a connected graph, i. e. the desired skeleton is minimal (as few skeleton points as possible), 4-connected and 1 point thick. Additionally, we want to ensure that every boundary point can be mapped onto at least one skeleton point.

To accomplish this, we perform the FMM DT and the AFMM Star FT and apply the followband approach described in section 3.3 to identify skeleton points. During this process, we also identify skeleton endpoints for use in steps 2 and 3. The skeleton can contain some anomalies (cycles, areas with a thickness greater than 1), which will be dealth with in the following step.

Step 2: Remove cycles, perform thinning (section 3.4.3) We detect and remove small cycles and other anomalies in the skeleton / graph and perform the topological thinning on the skeleton. This achieves the desired result: a 4-connected skeleton where the number of adjacent skeleton points is 1 for skeleton endpoints, 3 or 4 for junctions and 2 for all other skeleton points.

3.1.2 Phase 2

This phase yields the steps (3,4) which compute the boundary-collapse measure in an incremental fashion.

Step 3: Initialize skeleton traversal (section 3.6) We take the reference counting approach (section 3.7.1), which is a local measure derived from the FT, and compute this measure for each skeleton point. Effectively, this step produces the *boundary* $\rightarrow$ *skeleton* mapping which is key to triggering the boundary collapse events (section 3.7.2).

Step 4: Traverse skeleton (section 3.6), collapse boundary (section 3.7.2) The skeleton traversal approach taken in this step bears many similarities with the FMM. The skeleton is traversed from the skeleton endpoints towards the center of the skeleton. For every traversed skeleton point, the local measure that was computed in step 3 is inverted (i. e. a "undo" operation). Although the order in which this measure was initialized does not matter, the order in which we perform these "undo" operations ensures the importance is computed in the desired manner. As the skeleton is traversed, boundary collapse events are generated, adding "bits" of importance to the currently processed skeleton point and propagating it to subsequent skeleton points. Eventually, the last skeleton point that is processed in this manner will yield an importance value that is equal to the entire length of the boundary of the shape.

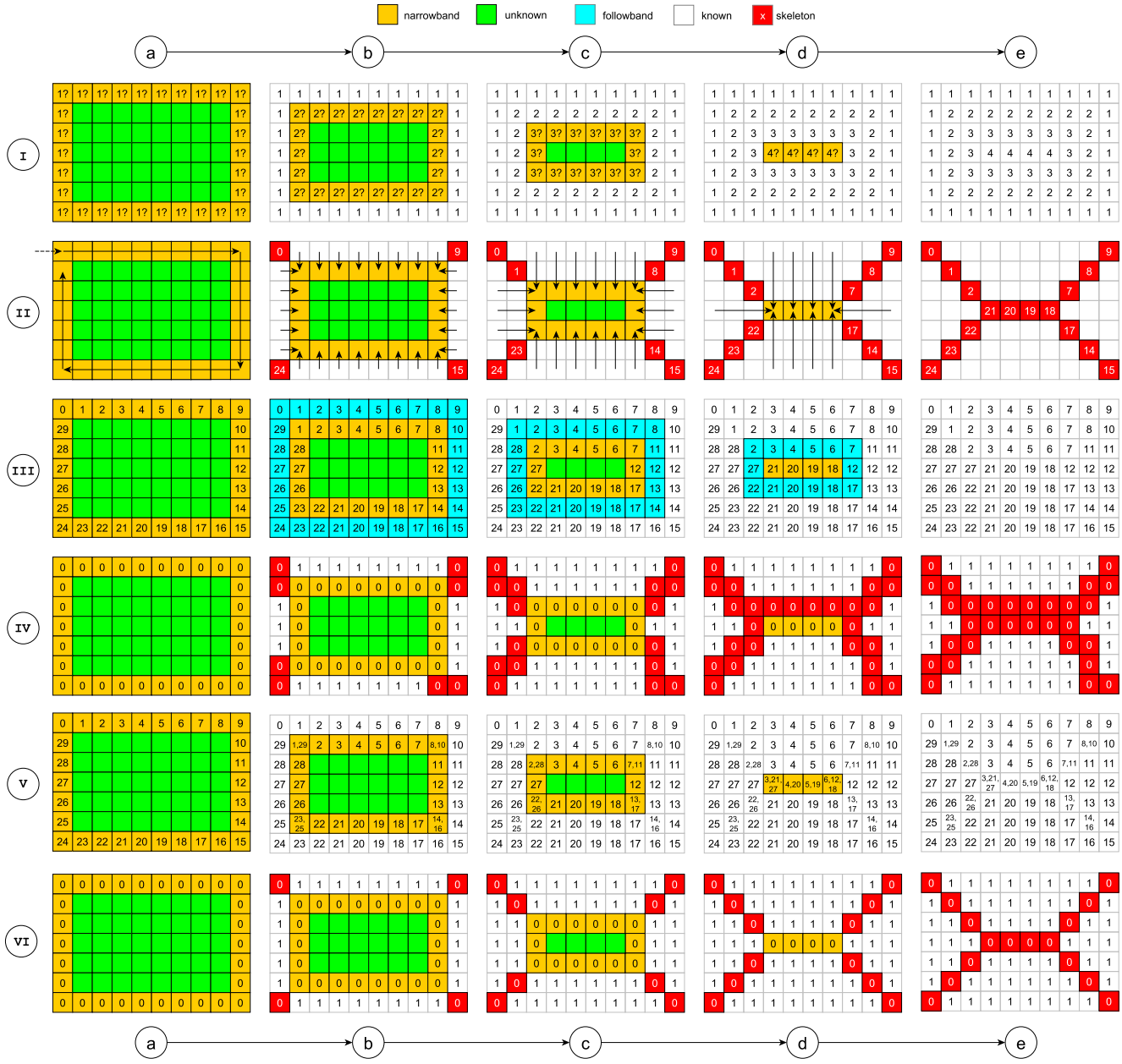


Figure 3.4: Stepwise depiction of our method. Rows: (I): FMM DT. (II): boundary initialization, information flow and skeleton. (III/V): simple FT ($\left|B(p)\right| = 1$) / full FT ($\left|B(p)\right| \geq 1$. (IV/VI): inheritcount measure for simple FT / full FT. Columns: (a): boundary initialization. ($b \ldots e$): DT / FT / inheritcount / skeleton detection progress. Pixel labels: (I): DT value, (II, III, V): FT boundary labels, (IV, VI): inheritcount for (III, V).

3.2 Boundary modeling

The core idea of our method relies on being able to traverse the boundary of the shape in any direction. Since we operate on a discrete (pixelized / voxelized) grid, this means we want to be able to step from one edge of a pixel (2D) or one face of a voxel (3D) to a edge / face of an adjacent pixel / voxel. Aiming for a method that generalizes to higher dimensional problems, it makes sense that our representation for the boundary of a shape can generalize to higher dimensional space as well.

We focus on a representation which suits our particular purpose (finding adjacent boundary points) while mapping directly onto basic digital topology. Because the Blum skeleton is defined in a similar manner, such a representation of the boundary is advantageous: mappings between skeletons, shapes and boundaries of different dimensionality are intuitive and well-defined. Although this may appear as somewhat cumbersome and overly theoretical, the result is fairly simple and efficient and alleviates complications at a later stage.

We note that this boundary model is in no way essential for our proposal and could very well be stripped out of our implementation or replaced by other models. As is, the model provides a way to relate our 3D proposal with our 2D proposal as well as the AFMM Star and is generally useful for visualizing, verifying and / or debugging the implementation.

In the following sections, we define a 2D representation that closely matches the AFMM Star representation. We then generalize this representation to the 3D case by showing how to combine multiple 2D representation into one 3D representation.

3.2.1 Adjacency mapping

In this section, we elaborate on the chosen adjacency mapping and provide a formal definition. An approach that appears straightforward at first is analyzed and the issues identified in this manner are used to convey the merits of a more elaborate solution.

Straightforward approach A straightforward approach to determine adjacency between boundary points is to simply inspect the neighbourhood around a boundary point p and declare all boundary points in that neighbourhood as adjacent to p (and vice-versa). In cases where the shape is of genus > 0 or when the shape contains a thin section, this can produce some awkward results as shown in fig. 3.5 (right) as points on opposing sides of a shape are now defined as adjacent. This is clearly not an accurate representation of the topology of the shape, as it allows topologically invalid paths.

To overcome this, we can eliminate the problematic point configurations from the input using basic morphological operators (binary opening / closing). This has the undesirable effect of potentially altering the topology of the input, especially for low-resolution inputs. Fully ruling out such topological changes requires that all 1 point thick structures become 2 points thick, which implies that we must increase (i. e. double) the resolution of the input. Since the performance of our method strongly

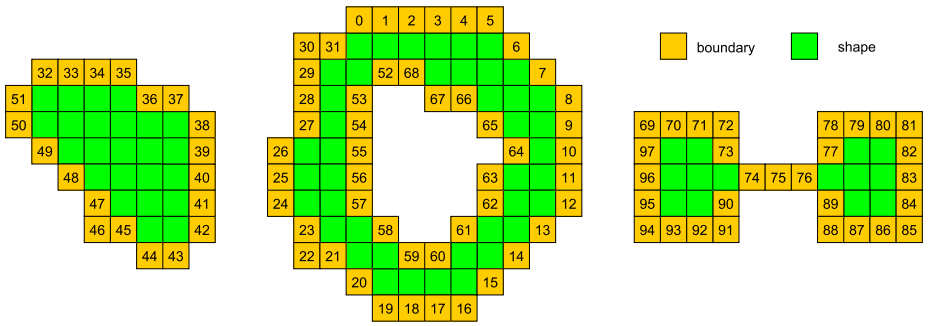


Figure 3.5: Boundary model initialization of an image containing three separate shapes. The image is scanned left-to-right, top-to-bottom. When a boundary is found, it is traversed clockwise for "outside" boundaries and counterclockwise for "inside" boundaries.

depends on the size of the input, especially for the 3D case, this is very costly and undesirable.

Another option is to handle all the problematic cases separately, which appears feasible for the 2D case. However, the number of possible configurations in the 3D case is daunting and the 3D case is hard enough to comprehend already. Clearly the approach is not as "straightforward" after all.

(a) (b)

Figure 3.6: (a) Initial boundary labels, (b) numbering scheme for pixels adjacent to p.

Chosen approach We thus opt for a more elaborate representation of boundary points and their neighbouring boundary points. For the 2D case, each pixel has up to four edges, each having two adjacent edges. If two or more adjacent edges belong to the same pixel, we compress these edges and consider them as one edge. This provides us with the desired behaviour (i. e. for each uncompressed / remaining edge, two adjacent pixels are defined) and enables us to compute the number of remaining edges by counting the number of connected components of p with the H-crossing number [22]:

$$X_H(p \in \partial\Omega) = \sum_{i=1}^{4} c_i$$

$$\text{where } c_i = \begin{cases} 1 & \text{if } x_{2i-1} \notin \Omega \ \wedge \ \big(x_{2i} \in \Omega \vee x_{2i+1} \in \Omega\big) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\text{and } x_8 = x_0$$

Here, the relative positioning of each c_i w.r.t p is as depicted in fig. 3.6b. As such, if $n = X_H(p)$ we will need to track adjacent edges for each of n remaining edges of p. For the 3D case, we can simply apply this scheme for each of the three 2D planar slices through p. The resulting approach is allows us to traverse the 1D edges of boundary pixels (2D shapes) and the 2D faces of boundary voxels (3D shapes) in an unambiguous manner.

If we now look at fig. 5.1 (right) again, it becomes clear that the chosen representation does not degrade the topology of the input, even if the input contains pixel-thin details. As such, this particular representation is reliable and accurate at low resolutions.

Implementation To implement our boundary model, two sweeping patterns are used: `sweepgrid` (left to right, top to bottom) and `sweepboundary` (follows the boundary of the shape). We start with `sweepgrid` at the top left corner. When we encounter a boundary point that has not been initialized yet, we pause `sweepgrid` and enter `sweepboundary` at the current coordinates. For the shape in fig. 3.5, the first point we encounter in this manner is point 0.

During `sweepboundary` we follow the boundary in a clockwise order and number points incrementally. We go around the shape and end up back at point 0. We denote that points $\{0 \dots 31\}$ belong to boundary B_0 and resume `sweepgrid` at point 1.

Since `sweepboundary` has found and initialized several boundary points, the next boundary point found by the `sweepgrid` is 32. Again we enter `sweepboundary` and denote boundary B_1 as the points $\{32 \dots 51\}$. We then resume `sweepgrid` at point 33.

After detecting boundary B_2 as $\{52 \dots 68\}$ we resume `sweepgrid` at point 68. Since all boundary points are already initialized, `sweepgrid` ends when it reaches the bottom right corner.

3.3 Followband

Consider the narrowband used by the FMM (section 2.2.1), which is a level-set with an initial shape identical to the input boundary. During the DT, the boundary is advected with constant unit speed in the direction of its inward normal field. The followband can then be defined as another level-set that directly "follows" the narrowband and is represented by a slightly larger shape wrapped tightly around the narrowband.

The followband allows us to react to certain events occuring during the computation of the DT and FT, as opposed to performing computations in a post-processing step. Reacting to events differs from a post-processing step in the sense that we can easily access the state of all the tracked information at a particular timestep. Performing the same in post-processing step would require an elaborate data structure for recording and retrieving the state changes. This would be a costly, complex and often error-prone endeavour which we avoid.

In section 3.3.1 we show how we define such a band of points in terms of changes to the narrowband, how it can be made to work and why it provides the desired result. We formalize the followband and we detail on various aspects of the approach in section 3.3.2.

3.3.1 General idea

As the narrowband "sinks" into the shape, the followband takes its place, and the space that was occupied by the followband is then checked for skeleton points. Since the FMM removes one point p at a time from the narrowband, updating the followband accordingly requires changes only to points nearby p. As such, the followband can be made to "follow" the narrowband via a simple local measure.

The purpose of the followband is to efficiently trigger an event for a point p_m as soon as p_m and all adjacent points have their final DT values. Clearly, points have their final DT values when the entire DT is computed. However, responding to this event gives easy access to the values of adjacent points at a particular timestep, i. e. during the computation of the DT. An example is shown in fig. 3.7, showing the manner in which narrowband, followband and skeleton detection are intertwined. The net result of this approach is that these events trigger (roughly) in order of increasing DT value, which turns out to be key to the skeleton detection approach that is discussed in the next section.

Simply put, we want to use the followband to perform computations on points as soon as a certain state is entered. For boundary points, the relevant state relates to the number of points in the shape that can still affect that boundary point. For points nearby the narrowband, the relevant state relates to whether the DT and FT has been computed for all adjacent points.

3.3.2 Definition

When a point p_t is removed from the narrowband at timestep t and its flag is set to `KNOWN`, it becomes a followband point and p_t is "added" to the followband. It is then removed from the followband at some timestep m when it no longer has any adjacent points q^{N}. To indicate that p_t is a followband point, we denote p_t^{F}, and we define the followband:

$$\partial^{\mathrm{F}}\Omega_t = \left\{ q_m \;\middle|\; q_m \in N_{\perp}^{\mathrm{K}}(p_t) \wedge p_t \in \partial^{\mathrm{N}}\Omega_t \wedge m \leq t \right\} \tag{3.2}$$

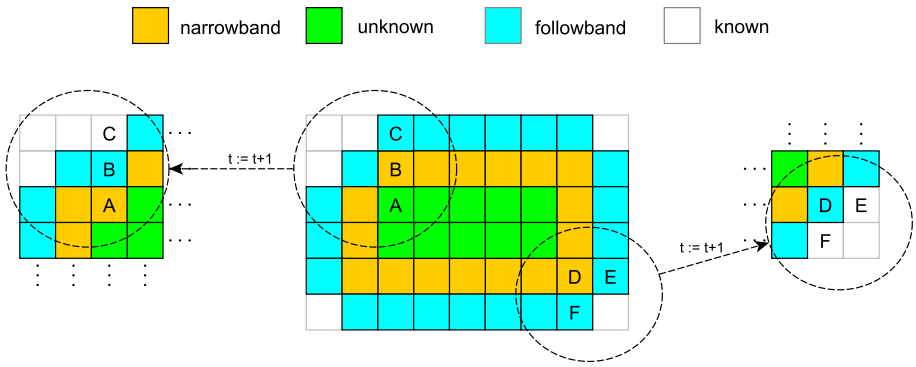


Figure 3.7: Visualization of the followband for consecutive timesteps. Middle: entire shape, left and right: changed section of the shape after removing B and D from the narrowband. B,D are removed from the narrowband and added to the followband, A is added to the narrowband and $B(A)$ is computed, C,E,F are removed from the followband and trigger the skeleton detection logic to be applied to C,E,F at this time

Note that when p_t is removed from the narrowband when there are no adjacent narrowband points, we will have no way of triggering the followband removal of p_t. This happens at extrema in the DT, for example for the very last point in the shape removed from the narrowband. In this case, we can remove p_t from the followband immediately after it is added, hence we include in eq. (3.2) the stronger term $m \leq t$ instead of $m < t$. This ensures that the desired event can always trigger for every point in terms of the removal from the narrowband of some adjacent point p_t.

Implementation When we initialize the FMM / AFMM Star, the followband does not contain any points. At every timestep t, the FMM removes a point p from the narrowband. We immediately add p to the followband and count the number of adjacent narrowband points $q^{\mathrm{N}} \in N_{\perp}(p)$ and store this number as the followcount of p.

We then inspect all adjacent followband points $q^{\mathrm{F}} \in N_8(p)$ and decrement the followcount of each such point q^{F}. If this causes the followcount of q^{F} to drop to zero, then p must be the last narrowband neighbour of q^{F} and we now trigger the removal of q^{F} from the followband.

After inspecting all the neighbouring points of p, we check the followcount of p itself. If the followcount of p is already zero, then the followband removal of p cannot be triggered as per our definition due to the absence of adjacent narrowband points. The lack of adjacent narrowband points implies that no point r adjacent to p has $T(r) > T(p)$. As such, p is an extremum in the DT. It follows from eq. (3.2) that it is now correct to remove p from the followband immediately.

Computational cost The followband uses only one byte of memory per point $\in \Omega$ and $O(|\Omega|)$ point inspections. Only integer math and cheap conditionals are used; the followband is a low-overhead approach. Additionally, since we inspect points nearby the point that was just processed by the AFMM Star, the required data is likely already in the CPU cache. Overall, the cost is minimal.

It is strictly not necessary to store the followcount for a point p, since the followcount can at all times be derived directly by inspecting the flagvalue of adjacent points. Storing the value however eliminates multiple recomputations of the followcount of a point and allows a simple and elegant implementation.

Variations One can easily come up with several variations of the followband. In our case, the followband does not overlap with the narrowband and strictly "follows" after it. If we define the followband in terms of `UNKNOWN` points instead, then the narrowband and followband can overlap and the narrowband is then a subset of the followband.

We have not exhaustively checked all the various options and instead resorted to using a variant that is intuitively clear, appears to be sufficient for the 2D case and reliably results in a correct skeleton.

3.4 Skeleton

Computing the skeleton of a shape using a method that works for both 2D and 3D shapes is a complex problem. Approaches that work well enough in 2D can easily break down in 3D, and the sheer number of possible voxel configurations makes it difficult to understand and visualize the problem at hand. Regardless of how we deal with these problems, it is desirable to represent the skeleton in a uniform, practical manner.

As detailed in section 2.4.1, the $(n-2)$-dimensional skeleton of a n-dimensional shape should be a subset of the $(n-1)$-dimensional skeleton. This effectively means that we must be able to "trim" a higher dimensional skeleton to yield a lower dimensional skeleton: the 2D surface skeleton (SS) is trimmed to yield the 1D curve skeleton (CS), and the curve skeleton is trimmed to yield the 0D skeleton root (SR).

Aside from representing lower dimensional skeletons as subsets of higher dimensional skeletons, our method also requires that we can traverse the skeleton in much the same manner as we can traverse the boundary of a shape. As such, we also need a skeleton that we can readily interpret as a topological graph. Implementation-wise, each skeleton point is represented by a node and must yield edges to adjacent skeleton points / nodes.

In the following sections, we detail on how we detect, represent and regularize skeletons of 2D shapes in a way that accommodates our desired generalized skeletonization method. The key aspect here is how the skeleton is represented, which enables the use of relatively simple and reliable detection, regularization and traversal methods.

3.4.1 Representation

We represent the skeleton as a collection of 4-connected (2D) or 6-connected (3D) points. Due to the topological properties of 4- and 6-connected shapes, skeleton endpoints and junction can be identified by a simple classification scheme: the number of 4- or 6-adjacent skeleton points n_a. For skeleton endpoints $n_a = 1$, for junctions $n_a = \{3, 4\}$ and for other skeleton points $n_a = 2$.

As will become clear in the following sections, this definition is advantageous in several ways:

- It maps directly onto digital topology, which simplifies generalizations to higher dimensional space
- It defines simple points in a way that relates directly to the different main and subtypes of skeleton points (section 2.4.1). This allows us to define effective and simple traversal and regularization (i. e. thinning) methods
- It eliminates the need to construct and maintain a separate graph structure: the pixels in an image (or voxels in a volume) unambiguously define the topological graph of the skeleton
- It allows for fewer possible pixel configurations, allowing techniques to operate on the skeleton without the burden of a tedious and error-prone case analysis
- It is easily converted to a 8-connected skeleton (if desired), reducing the number of skeleton points and allowing a smoother looking skeleton

Due to the intrinsic properties of a discrete grid, the skeleton of a shape is not always restricted to a single solution. For a shape with a even thickness, there is a 2 point thick area in which we somehow must determine a 1 point thick skeleton. An example is shown in fig. 3.8, in which the image to the right clearly cannot yield a centered skeleton due to discretization.

Although fig. 3.8 depicts a 8-connected skeleton, the problem set is the same for our 4-connected case.

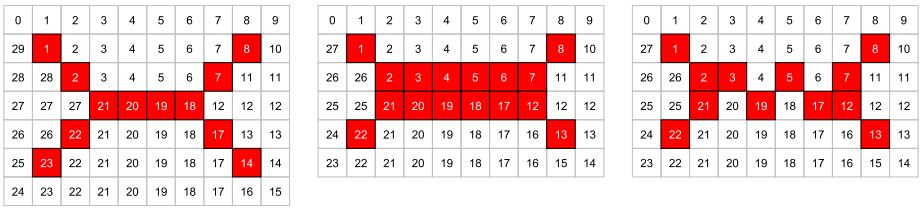


Figure 3.8: 2D shape of odd height (left) and even height (middle, right). Numbers indicate closest boundarypoints. Red (dark) points: skeletonpoints (thick), yellow (gray) points: other points.

3.4.2 Detection and classification

In this section we elaborate on the approach used by the AFMM Star to detect skeleton points and analyze it in detail, after which we present an alternative, improved method.

AFMM Star approach Our skeleton detection is based on the AFMM Star approach, which detects skeleton points via the FT and applies a directional gradient to reduce the skeleton to a thin structure. The AFMM Star compares closest boundary points $B(p)$ of p at coordinates (i, j) to the closest boundary points of q at $(i+1, j)$ and r at $(i, j+1)$. When $B(p)$ is not boundary-adjacent to $B(q)$ or $B(r)$, then p is identified as a skeleton point.

By comparing p only to adjacent points in the x_+ and y_- direction, the skeleton is be skewed towards the bottom and the right and can yield skeleton branches that visually "clump" together (fig. 3.9a). Since this hinders our goal of computing a thin, well-formed, 4-connected skeleton, we need a alternative method.

Analysis If we apply the AFMM Star approach for a point p to all other points $\in N_8(p)$ instead of just its x_+ and y_- neighbours, we get a result like fig. 3.9b. This effect happens when disturbances on the boundary lie at the minimum possible distance from one another. Inspecting all points in the 3×3 neighbourhood around p clearly identifies too many points as skeleton points, resulting in what looks like a dilated skeleton. This is exactly what the gradient approach is trying to avoid.

The key issue with the skeleton detection approach is that it is a function of the FT of two adjacent points (p, q). If p is a skeleton point w.r.t q, then q is also a skeleton point w.r.t p. The gradient approach effectively solves this issue by comparing only specific adjacent points.

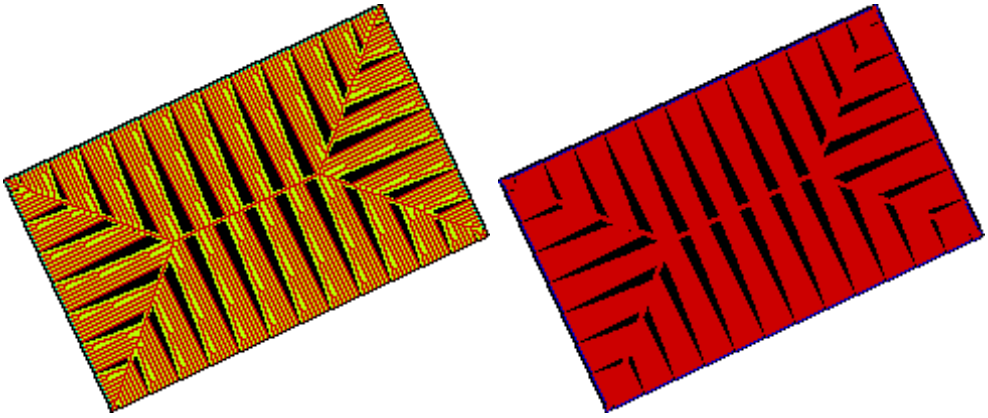
Proposed approach To avoid this skewing / clumping problem, we perform the skeleton detection while the DT and FT are still being computed. For a point p the detection is triggered at the time when p is removed from the followband. From the definition of the followband (see section 3.3), $B(q)$ and $T(q)$ are already computed for all points $q \in N_8(p) \cup p$.

This seems no different than what is available in a post-processing step. What *is* different, however, is that some of the neighbouring points of p are in the followband, some are in the narrowband and some are already flagged as `KNOWN`. We use this information to simplify our problem and "grow" the skeleton from the boundary towards the extrema in the DT.

We determine if p is a skeletonpoint by comparing it with all adjacent points q which are not already detected as a skeletonpoint at the time when p is removed from the followband. This is a crude but simple way to avoid the issue depicted in fig. 3.9b by taking advantage of the order in which points are processed to enforce an alternating sequence of skeleton- and non-skeleton points.

To formalize the skeleton detection, we define:

$$isSkeleton(p, q) = \Big\{ (p, q) \in \Omega \;\Big|\; \neg isAdjacent\big(B(p), B(q)\big) \Big\} \tag{3.3}$$



(a) (b)

Figure 3.9: Difference between detecting skeleton points inline during DT / FT computation (a, AFMM Star) or postprocessing (b, our method), as defined in eq. (3.6). Red: detected skeleton points. Yellow: points that can be excluded via the inline approach

where *isAdjacent* is defined as:

$$isAdjacent(b_1, b_2) = \left\{(b_1, b_2) \in \partial\Omega \;\middle|\; \left(b_1 = b_2 \vee b_1 \in N_\perp^\partial(b_2)\right)\right\} \tag{3.4}$$

where $N_\perp^\partial(p)$ is defined in section 2.1 as the points that are boundary-adjacent to p. We can then represent the skeleton the AFMM Star detects as:

$$\mathcal{S}(\Omega) = \left\{p \in \Omega \;\middle|\; \exists q \in \left\{(p_i, p_{j+1}), (p_{i+1}, p_j)\right\} : isSkeleton(p, q)\right\} \tag{3.5}$$

and for our method we define the skeleton as:

$$\mathcal{S}(\Omega) = \left\{p \in \Omega \;\middle|\; \exists q \in N_8(p) : isSkeleton(p, q) \wedge T(p) > T(q) \wedge q \notin \mathcal{S}(\Omega)\right\} \tag{3.6}$$

Effectively, if we leave out the term $q \notin \mathcal{S}(\Omega)$ in eq. (3.6), we get fig. 3.9b while including the term results in fig. 3.9a.

3.4.3 Thinning and regularization

Our skeleton detection approach usually produces a skeleton that is not free from artifacts. Removing these artifacts is necessary for the skeleton traversal step to work, which is discussed in detail in the following sections.

Removing cycles

Before we perform the thinning, we detect and remove cycles in the skeleton. These cycles occur as a side effect of using the followband trigger (section 3.3) to decide when

to perform the skeleton detection step on points. The order in which these triggers are executed cannot be relied on to avoid these cycles. This is similar to the issue in the AFMM that was subsequently mitigated in the AFMM Star[32]. A visualization of these cycles is shown in fig. 3.10.

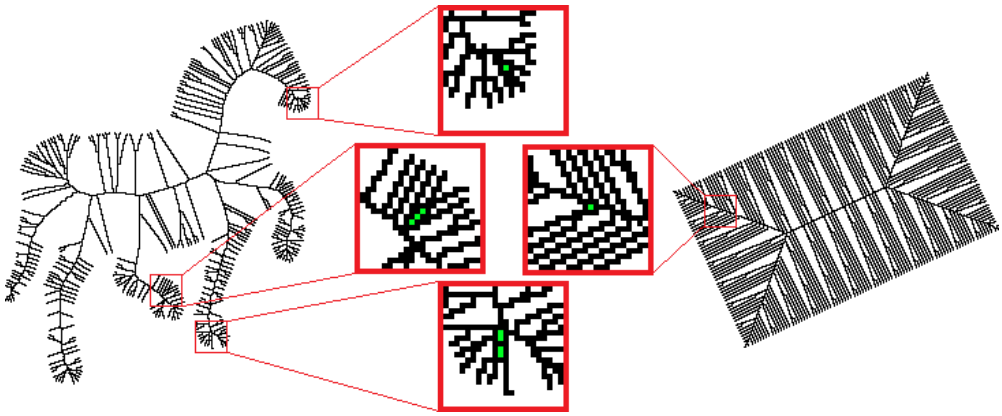


Figure 3.10: Cycles in the skeleton formed by our skeleton detection approach for the `anim4` (left) and `rect1` (right) shapes. Green pixels depict the non-skeleton points enclosed by such cycles.

If we can identify the "empty" points which form the gap inside each cycle, we can simply flag them as skeleton points and let the topological thinning step deal with the thickened skeleton sections created in this manner. Fortunately, using a 4-connected skeleton implies that an empty point inside a cycle cannot be 8-adjacent to an empty point not inside a cycle.

We utilize this property and perform a floodfill to find all empty points which are 8-connected to the boundary of the shape. Here we use the points in $\partial\Omega$ as the seed points. Points not reached by the floodfill must then be the empty points that form the gaps.

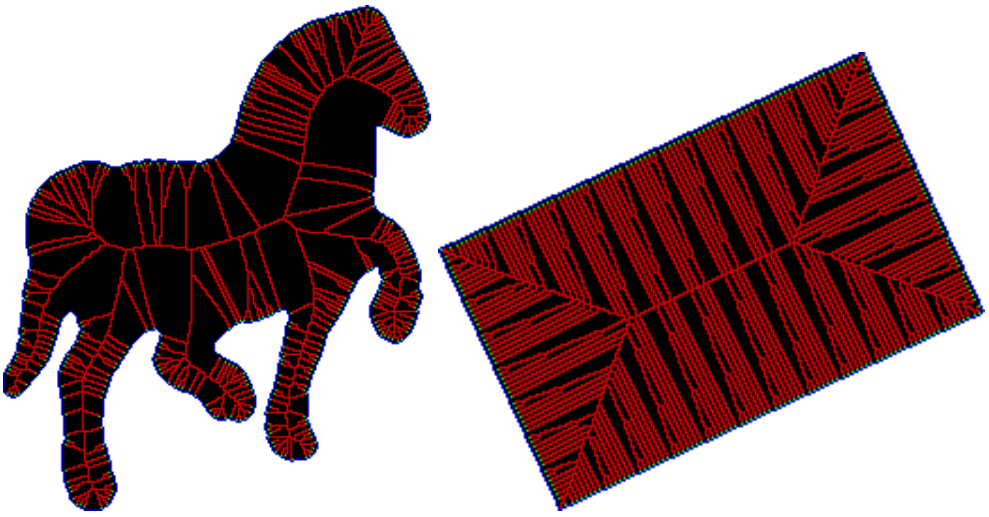
The computational cost of the cycle removal step is $O(|\Omega|)$ for the floodfilling and $O(|\Omega|)$ for finding the points that were not reached by the floodfill. Memory requirements depend on the floodfill implementation and / or the shape topology, a straightforward approach typically requires $O(|\Omega|)$ memory in the worst-case scenario.

Thinning process

We perform a topological thinning on the skeleton using the connectivity number X_B [22] to identify simple points eq. (3.7). A point is simple iff $X_B = 1$. To ensure that we do not remove the entire skeleton, we first identify the endpoints of the skeleton, serving as anchor points which we will not trim away. A skeleton point p is an endpoint iff p is simple and $T(p) < 4$. Here, skeleton points are foreground points (1), other points are background (0) and x_i are positioned relative to p as defined in our boundary model (fig. 3.6b on page 28).

$$
\begin{aligned}
X_B\big(p \in \mathcal{S}(\Omega)\big) \;&=\; \sum_{i=1}^{4} x_{2i-1} - x_{2i-1} * x_{2i} * x_{2i+1} \\
\text{where} \quad x_8 = x_0 \text{ and } x_i &= \begin{cases} 1 & \text{if } x_i \in \mathcal{S}(\Omega) \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{3.7}
$$

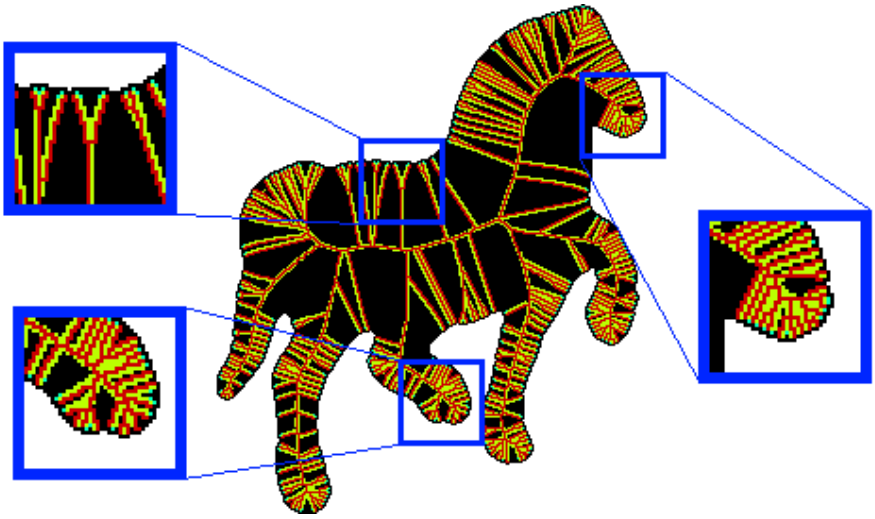
We iteratively remove all simple points from the skeleton until no removeable simple points remain. We do this by adding all points p for which $p \in \mathcal{S}(\Omega) \wedge p \notin \mathcal{S}_{\mathcal{E}}(\Omega)$ to a sorted map, processing points in order of ascending $T(p)$ value. This ensures that we end up with a complete, well-formed and 4-connected skeleton. The result of the thinning and classification is shown in figs. 3.11a and 3.11b.



(a) (b)

Figure 3.11: Skeleton classification for `anim4` (left) and `rect1` (right) shapes. Red: skeleton $S(\Omega)$, green: endpoints $S_{\mathcal{E}}(\Omega)$, blue: boundary $\partial\Omega$ / initial narrowband $\partial^{\mathrm{N}}\Omega$

Some limitations of this unsupervised thinning method are visible in fig. 3.12a where the unthinned skeleton is shown together with the thinned skeleton. Clearly there is opportunity for improving the smoothness, but a method to do so is not provided here.



(a)

Figure 3.12: Example of thinned + unthinned skeleton for `anim4` shape, revealing areas where the smoothness of the skeleton is suboptimal

3.5 Skeleton DT

The skeleton DT is defined analogous to the FMM DT in section 2.2.1 by applying several substitions. The Euclidean distance is replaced by the boundary collapse measure $\rho_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}_+$, the shape Ω is replaced by the skeleton $\mathcal{S}(\Omega)$ and finally the boundary points of the shape $\partial\Omega$ are replaced by the endpoints of the skeleton $\mathcal{S}_{\mathcal{E}}(\Omega)$. The definition for $\rho_{\mathcal{S}}$ is given in section 3.7, and in this section we simply assume that $\rho_{\mathcal{S}}$ can be computed.

For a skeleton $\mathcal{S}(\Omega)$ containing n points p_t with $t \in [0 \dots n\rangle$, we will remove points from the narrowband in a sequence of n timesteps. We rewrite eq. (2.4) as

$$\rho_{\mathcal{S}}(p^{\mathrm{K}}) \leq \rho_{\mathcal{S}}(p^{\mathrm{N}}) < \rho_{\mathcal{S}}(p^{\mathrm{U}}) \tag{3.8}$$

and eq. (2.5) as

$$\partial^{\mathrm{N}}\mathcal{S}(\Omega)_{t+1} = \begin{cases} \partial\mathcal{S}(\Omega) & \text{if } t < 0 \\ \partial^{\mathrm{N}}\mathcal{S}(\Omega)_t \setminus p_{t+1} & \text{if } 0 < t < n \\ \emptyset & \text{if } t \geq n \end{cases} \tag{3.9}$$

and eq. (2.7) as

$$\rho(p_a) \leq \rho(p_b) < \rho(p_c) \quad \text{where} \quad 0 \leq a \leq b < c < n \tag{3.10}$$

and finally eq. (2.6), which applies to the skeleton DT if we copy it verbatim:

$$\mathit{flag}(p_s) = \begin{cases} \texttt{KNOWN} & \text{if } s \leq t \\ \texttt{NARROWBAND} & \text{if } t < s < m \\ \texttt{UNKNOWN} & \text{if } m \leq s \end{cases} \tag{3.11}$$

Stopping criterion The FMM DT has a built-in stopping criterion: the lack of adjacent `UNKNOWN` points around a point p_t, which occurs around extrema in the DT. Together with the manner in which the flagvalue of a point is updated, this ensures that points are processed only once.

For the skeleton DT, the extrema in the DT correspond with junction points (and ultimately the skeleton root). In order to provide a comparable stopping criterion for the skeleton DT, we take advantage of our 4-connected skeleton representation in which junctions are non-simple points. Effectively, a point $q^{\mathrm{U}} \in N_8(p_t)$ is only added to the narrowband iff $X_B^{\mathcal{S}}(q^{\mathrm{U}}) = 1$, where $X_B^{\mathcal{S}}$ represents the number of connected components and is defined as

$$\begin{aligned} X_B^{\mathcal{S}}\big(p \in \mathcal{S}(\Omega)\big) \;&=\; \sum_{i=1}^{4} x_{2i-1} - x_{2i-1} * x_{2i} * x_{2i+1} \quad \text{where} \quad x_8 = x_0 \\ \text{and } x_i &= \begin{cases} 1 & \text{if } x_i \in \mathcal{S}(\Omega) \wedge \mathit{flag}(x_i) \neq \texttt{NARROWBAND} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.12}$$

Intuitively, this means that the skeleton DT prunes a skeleton branch until a junction point is encountered. This junction point $\mathcal{S}_{\mathcal{J}}$ is then only added to the narrowband if it is a simple skeleton point at that particular timestep. Due to the inherent properties of our 4-connected skeleton, this is guaranteed to happen, since pruning skeleton branches eventually reduces all junctions to inner skeleton points $\mathcal{S}_{\mathcal{I}}$ for which eq. (3.12) = 1.

3.6 Skeleton FT

The initial boundary of the skeleton, denoted by $\partial\mathcal{S}(\Omega)$, consists of skeleton endpoints, i. e. $\partial\mathcal{S}(\Omega) = \mathcal{S}_{\mathcal{E}}(\Omega)$. We assign to each of these endpoints p the value of $B(p)$ as the boundary label for the 1D FT, which makes a visual representation of the skeleton FT conveniently similar to the FT of the entire shape. In terms of the FMM, we can say that $\partial\mathcal{S}(\Omega) \rightsquigarrow \partial\Omega$.

We denote $b_{\mathcal{S}} \in F_{\mathcal{S}}(p)$ as the feature points of p, where $b_{\mathcal{S}} \in \mathcal{S}_{\mathcal{E}}(\Omega)$. A single feature point is assigned to skeleton points in the order in which points are visited by the skeleton DT. When more than one feature point can be assigned to a skeleton point, we choose the feature point with the highest computed importance value.

The skeleton FT is defined analogous to the "regular" FT. In the continuous case, a point $b_{\mathcal{S}}$ is a feature point $F_{\mathcal{S}}(p)$ of $p \in \partial\mathcal{S}(\Omega)$ and the feature transform is defined as:

$$F_{\mathcal{S}}\big(p \in \mathcal{S}(\Omega)\big) = \Big\{ q \in \partial\mathcal{S}(\Omega) \;\big|\; ||q - p|| = \rho_{\mathcal{S}}(p) \Big\} \tag{3.13}$$

Similar as for the "regular" FT, we denote for the discrete case $b_{\mathcal{S}} \in B_{\mathcal{S}}(p)$ analogous to $B(p)$ and assume eq. (3.14).

$$B_{\mathcal{S}}(p) \rightsquigarrow F_{\mathcal{S}}(p) \tag{3.14}$$

Note on discrete vs continuous In the continuous case, $p \in \mathcal{S}_{\mathcal{E}}(\Omega) \Rightarrow p \notin \partial\Omega$. In the discrete case, this is not necessarily so, e. g. an axis-aligned shape of size $2 \times 2 \times 5$ cannot yield a skeleton point that does not lie on the initial boundary. As such, we do not consider the above case as undesirable or incorrect given the use of a discrete representation.

3.7 Importance measure

The importance measure can be seen as the result of an interaction between the boundary, skeleton and a particular order in which interactions occur. In the following sections, we detail on how the importance measure is computed and how we can implement it in terms of the AFMM. We show why the resulting measure is a multiscale importance measure similar to the geodesic idea introduced in [33].

We define a *boundary collapse* event (section 3.7.2) of some boundary point b by using the referencecount of b (section 3.7.1). The skeleton is traversed (section 3.6) one point at a time, updating the referencecount at every point. The referencecount identifies collapsing boundary points, which in turn contribute to the importance value of the currently processed skeleton point.

Advection model

The manner in which the importance measure is computed is inspired by interpreting it as the result of an advection process (section 2.8.3). In terms of the 2D problem at hand, this means that we will need to start at the endpoints of the skeleton, compute the importance of such an endpoint and then ensure that we always propagate this importance value onto the "next" skeleton point. Eventually, all the importance values are aggregated in the skeleton root, which should then be equal to $|\partial\Omega|$. During this process, we also need to ensure that the computed importance values are monotonically increasing towards the skeleton root.

3.7.1 Referencecount

The referencecount is a simple counting scheme that utilizes the FT to tie the boundary to the skeleton. The general idea is to track the number of mappings between skeleton points and boundary points during the skeleton traversal step (section 3.6). This mapping allows us to define a triggered event during which we collapse the boundary (section 3.7.2) at particular points. The referencecount determines *when* a boundary point collapses during skeleton traversal, and the particular skeleton point which

triggers the collapse event defines onto *which* skeleton point we should assign the importance contributed by the collapsed boundary point.

Definition

Whenever a boundary point b is a closest boundary point of $p \neq b$, we say that p references b. When we count the number of distinct points $p \neq b$ that reference b, we call this the *referencecount* of b. For our method, we compute the referencecount only using skeleton points. Since the skeleton is a 1D representation of a 2D shape, we can expect from the feature transform that all feature points of skeleton points reference every boundary point at least once.

In discrete space however, computing or representing a perfect feature transform is not always directly possible. One way to compensate for the discretization artifacts is to consider the feature set of adjacent points as *feature set candidates* for p. We enumerate for each skeleton point p the feature points b found in the neighbourhood of p, and increment the referencecount once for each of these feature points. The result is shown in eq. (3.15).

$$R(b \in \partial\Omega) = \#\Big\{p \in \mathcal{S}(\Omega) \;\Big|\; q \in \big(N_8(p) \cup p\big) \wedge q \neq b \wedge b \in B(q)\Big\} \tag{3.15}$$

Example

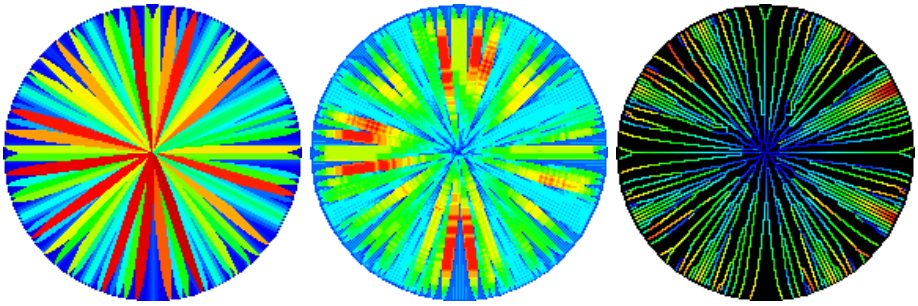
In order to visualize and understand the referencecount, we first apply eq. (3.15) to the entire shape by replacing the term $p \in \mathcal{S}(\Omega)$ in eq. (3.15) by $p \in \Omega$. We first compute the DT and FT. The result is a *shape* $\rightarrow$ *boundary* mapping: for every boundary point b we count the number of points p in the shape for which $b \in B(p)$. After we have performed this mapping for all points in the shape, we assign to each point in the shape the referencecount of its closest boundarypoint, i. e. $\forall p \in \Omega : \mathit{value}(p) = R\big(B(p)\big)$.

As can be seen in the resulting image in figs. 3.13a and 3.14a, high values correspond to points on concave boundarysegments. Another visualization can be made if we compute values for this image *during* the computation of the DT / FT, as opposed to constructing it after the entire DT / FT have been computed. The result of this is seen in figs. 3.13b and 3.14b, showing how the referencecount of points changes during the DT / FT.

When we generate figs. 3.13a and 3.14a using $\mathcal{S}(\Omega)$ instead of Ω, we get the result as depicted in figs. 3.13c and 3.14c. These images visualize the decreasing referencecount as the skeleton is traversed from the endpoint towards the center of the skeleton.

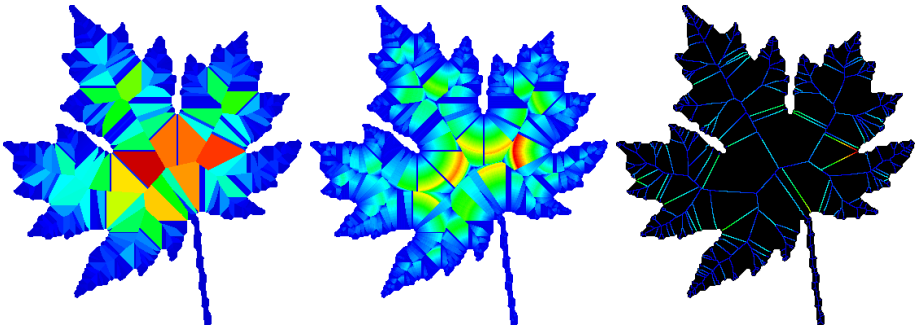
3.7.2 Boundary collapse

The collapse of a point on the boundary is a triggered event which fires at a particular timestep during the skeleton traversal step (section 3.6). This event is triggered whenever the referencecount of a boundary point b is reduced to zero after traversing onto a skeleton point p. When this event occurs, we say that b has *collapsed* onto p and that the importance associated with b contributes directly to p. In this section,



(a) (b) (c)

Figure 3.13: Referencecount example for `circle` shape



(a) (b) (c)

Figure 3.14: Referencecount example for `leaf1` shape

we discuss *how* these collapse events are defined and *where* the associated importance is assigned to. Details about how the actual importance is computed are discussed in section 3.7.3.

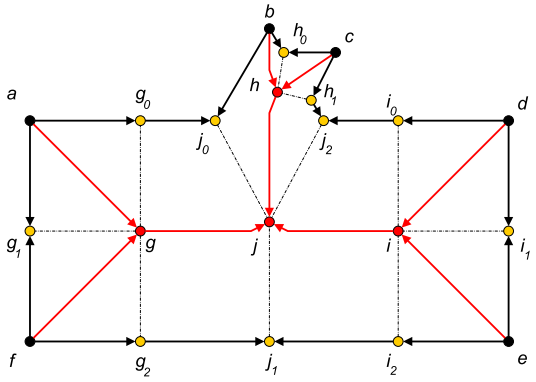
Example

The connection between the boundary collapse and referencecount is effectively an application of the pigeonhole principle. If we view the boundary points as the pigeonholes, then every skeleton point will add one pigeon to a pigeonhole if the boundary point b is *reachable* via p, i. e. when b is included in the union of all feature sets $B(q)$ of points $q \in (N_8(p) \cup p)$.

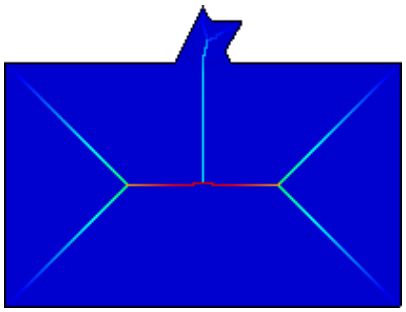
After the skeleton has been detected, we initialize the referencecount by adding pigeons to "reachable pigeonholes" for each skeleton point. For now, assume that this means that all pigeonholes have at least one pigeon, i. e. every boundary point is reachable by at least one skeleton point.

During the skeleton traversal step, we process a skeleton point p by removing a

pigeon from the reachable pigeonholes of p. When this removes the last pigeon from a pigeonhole, we know that the pigeonhole is no longer reachable by any other skeleton point that we have not visited yet and we can thus attribute the pigeonhole to p.



(a) Flow of importance. Importance increases along the skeleton (red arrows). ρ is computed in the direction of the black arrows. Blue points: skeleton endpoints $\mathcal{S}_\mathcal{E}$, red points: junctions $\mathcal{S}_\mathcal{J}$, yellow points: feature points $F(p)$ of junctions



(b) Actual output for a shape corresponding to the diagram in fig. 3.15a. Importance increases along the skeleton from blue (0) to red ($|\partial\Omega|$).

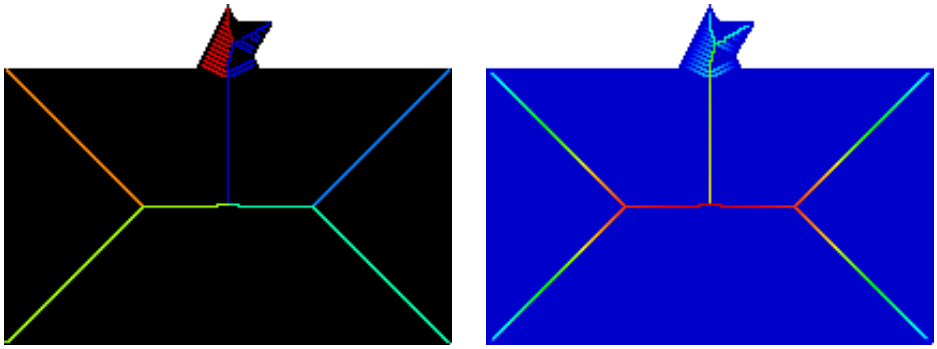
Figure 3.15

3.7.3 Incremental computation

As we traverse the skeleton (section 3.6), the referencecount of boundary points is decremented (section 3.7.1) and triggers the collapse of boundary points (section 3.7.2). In this section, we explain how the boundary distance corresponding with collapsed boundary points is computed.

Example

When closest boundary points $B(p)$ are determined for a point p, the neighbouring points $q \in N_8(p)$ are used to gather the list of boundary points that are potential candidates for $B(p)$. This implies that if a boundary point b is not a closest boundary point of any point q in the neighbourhood of any such point p, then b is "out of reach" and b can not directly contribute to the importance of p. An indirect contribution is possible however when p is not a endpoint of the skeleton, i. e. when it lies halfway a skeleton branch. If b contributes to any of the points between p and the endpoint of the branch, then p will receive this contribution via the skeleton.



(a) (b)

Figure 3.16: Transported endpoint label (left) and order in which the skeleton is traversed (right) for `rect_bump` shape

Definition

Collapsed boundary points only contribute to the importance value of a skeleton point if at least one adjacent boundary point has also collapsed into the same skeleton point. If this is the case, the importance contribution is computed as the Euclidean distance between the two adjacent boundary points. For the 2D case, this results in contributions of either 1 or $\sqrt{2}$.

This process is shown in a series of steps in fig. 3.18 for a rectangular partial shape (truncated on the right). The importance of the skeleton increases from the boundary into the direction of its inward normal. Each section of the boundary that corresponds with such an importance value is marked.

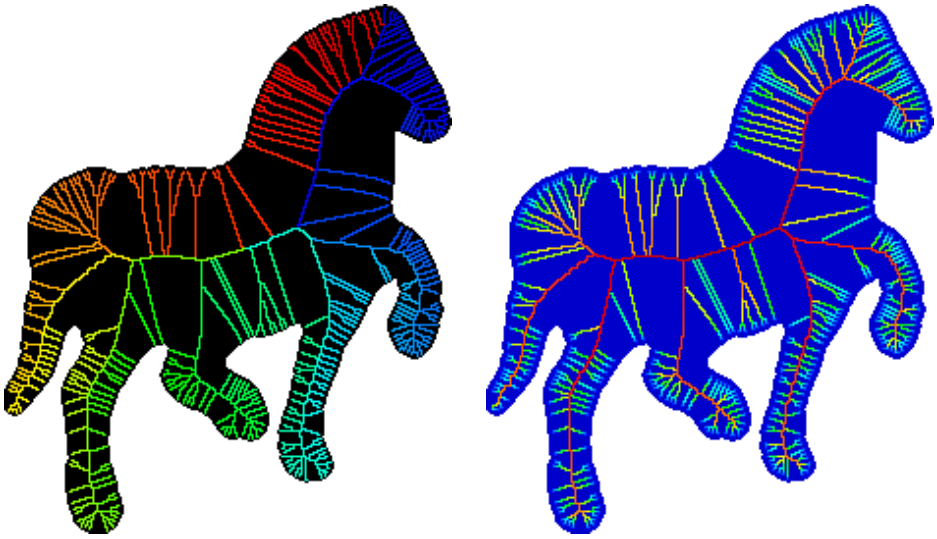
We define the importance $\rho_{\mathcal{S}}(p_t)$ of a point p_t, which is removed from the skeleton FT narrowband at timestep t, as:

$$\begin{aligned} \rho_{\mathcal{S}}\big(p \in \mathcal{S}(\Omega)\big) &= \sum \rho_{\partial}(b_t) + \rho_{\mathcal{S}}(q_r) \\ \text{where } b_t &\in B\big(p_t \cup N_8(p_t)\big) \\ \text{and } q_r &\in N_{\perp}(p) \wedge r < t \end{aligned} \tag{3.16}$$

where the importance contribution $\rho_{\partial}(b_t)$ of a boundary point b_t collapsing into p_t at timestep t is defined as:

$$\begin{aligned} \rho_{\partial}(b_t \in \partial\Omega) &= \sum ||b_t - c_{\mathcal{S}}|| \\ \text{where } c_s &\in N^{\partial}(b_t) \wedge isAdjacent(b_t, c_s) \\ \text{having } R(c_s) = 0 &\wedge R(b_t) = 0 \wedge R(b_{t-1}) > 0 \end{aligned} \tag{3.17}$$

Here, eq. (3.16) can be interpreted as the sum of boundary points collapsing into p_t plus the sum of importance values of adjacent skeleton points q_r which have been removed from the narrowband at a previous timestep $r < t$.



(a) (b)

Figure 3.17: Transported endpoint label (left) and order in which the skeleton is traversed (right) for `anim4` shape

Note that the importance measure is defined in terms of previously computed importance values of adjacent skeleton points, and that the order in which points are processed by the the skeleton FT ensures that the importance measure is monotonic.

Note on correctness This approach may result in partial skeleton branches having an importance of 0, which happens in cases where a collapsing boundary point does not have an adjacent collapsed boundary point. This only affects the extremities of the skeleton and is a side-effect of various minor inaccuracies that are inherent to the FMM DT, the AFMM Star FT and the discrete grid.

We can also compute the importance contribution of $b \in \partial\Omega$ as $\frac{1}{2}(||a-b|| + ||b-c||)$, where a and c are boundary points adjacent to b. Another potentially more accurate approach is to use a curve length estimator to to compute an estimated importance contribution, in which case a Newton polynomial intuitively seems to be a good fit.

However, when we apply a pruning threshold to the skeleton on the importance value, these skeleton branches are the first to be pruned. As such, these inaccuracies are unlikely to be relevant in practice.

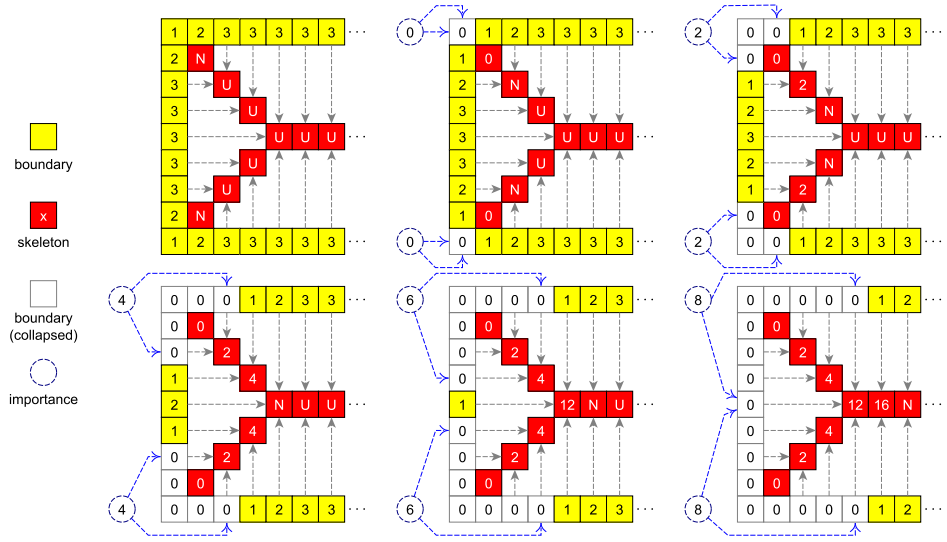


Figure 3.18: Computation of $R(b \in \partial\Omega)$ (eq. (3.15)) and $\rho\big(p \in \mathcal{S}(\Omega)\big)$ of a rectangular partial shape (truncated on the right) during 6 timesteps (left to right, top to bottom). For points $p \in \partial\Omega$, the value of $R(p)$ is shown, points $q \in \mathcal{S}(\Omega)$ show $\rho(q)$ when $\mathit{flag}(q) =$ `KNOWN`, `N` when $\mathit{flag}(q) =$ `NARROWBAND` or `U` when $\mathit{flag}(q) =$ `UNKNOWN`. See section 3.2.

3.8 Inheritcount

The inheritcount is a measure which extends the DT and FT by extracting information about *how* the FT is computed. The flow of information from the boundary towards the center of the shape is modeled as a *source* $\rightarrow$ *destination* relation where we say the destination "inherits" from the source. If *destination* is a point p and we compute $B(p)$, we require information stored in one or more adjacent *source* points q for which $B(q)$ is already computed.

The purpose of the inheritcount is primarily to provide a way to visualize and understand the FT as computed by the AFMM Star and provide insight in the differences between using a simple FT or the full FT. As such, the inheritcount is not part of our proposed method, but rather a supporting tool that yields some practical and conceptual benefits.

In the following section, we explain what the inheritcount is and why such a measure makes sense. We provide a formal definition in section 3.8.2 and in section 3.8.3 we analyze the measure and elaborate on how we can integrate the measure into our measure.

3.8.1 General idea

To clarify the reason for constructing the inheritcount measure, we apply the *source* → *destination* idea above to the skeleton points in a shape. Since skeleton points are by definition *extrema* in the DT, skeleton points should generally never be a *source* for any adjacent point. In relation to the FT, this is equivalent to tracking *how* a closest boundary point is found in addition to determining *which* boundary point is closest.

Consider the boundary collapse measure introduced as the AFMM Star in [39]. This measure is based on the idea that skeleton points relate to compact boundary segments which collapse at some timestep. If we express this idea in terms of the *source* → *destination* approach, we can conclude that a collapsing boundary segment is the result of a particular *source* for which no *destination* can be found. This relates closely to the skeleton detection approach taken by the AFMM Star: the lack of a *destination* implies that the *source* or some point adjacent to the *source* should find at least one pair of closest boundary points b_1, b_2 which are not adjacent on the initial boundary.

In different terms, when we observe a particular point b on the initial boundary, we can identify a "trail of breadcrumbs" towards the skeleton of the shape, where the trail consists of points which have b as a closest origin. When the end of the trail is reached, we will find a *source* for which no *destination* is found, and we can identify the *source* and / or adjacent points as skeleton point.

The idea behind this measure is consistent with the advection model (section 3.7) introduced in [39]. Information flows from the boundary towards the skeleton and across the skeleton away from the skeleton endpoints and can never flow from a skeleton point into a non-skeleton point. Our method considers the DT / FT only as a *boundary* → *skeleton* mapping and computes the information flow across the skeleton (a *skeleton* → *skeleton* mapping) in a subsequent phase. As such, the inheritcount measure relates only to the first phase in which the skeleton is detected.

3.8.2 Definition

We first define the points that are a *potential source* of p:

$$S_p(p \in \Omega \setminus \partial\Omega) = \Big\{(b, q) \;\Big|\; q \in N_{\perp}^{\{\mathrm{K}|\mathrm{N}\}}(p) \wedge b \in \big(B(p) \cap B(q)\big)\Big\} \tag{3.18}$$

from which we determine the actual sources of p for which the distance to the boundary point is minimal:

$$\begin{aligned} S_a(p \in \Omega \setminus \partial\Omega) = &\Big\{q \;\Big|\; (b, q) \in S_p(p) \wedge ||q - b|| = d_{min}\Big\} \\ &\text{where } d_{min} = \min_{(c,r)\in S_p(p)} ||r - c|| \end{aligned} \tag{3.19}$$

Effectively, q is a potential source of p if q share at least one closest boundary point with p, i. e. p could have obtained b via q. Consecutively, q is an actual source of p if $||q - b||$ yields the minimum distance over all potential sources of p, i. e. q is the source that is closest to b.

We can now define the inheritcount by simply counting the number of *destinations* for which a point p is the *source*:

$$I(p \in \Omega) = \#\Big\{ q \in \Omega \;\Big|\; p \in S_a(q) \Big\} \tag{3.20}$$

Since skeleton points should never be a *source*, we can interpret eq. (3.20) as a measure that yields low values for skeleton points and higher values for other points. This corresponds with the example depiction in fig. 3.19.

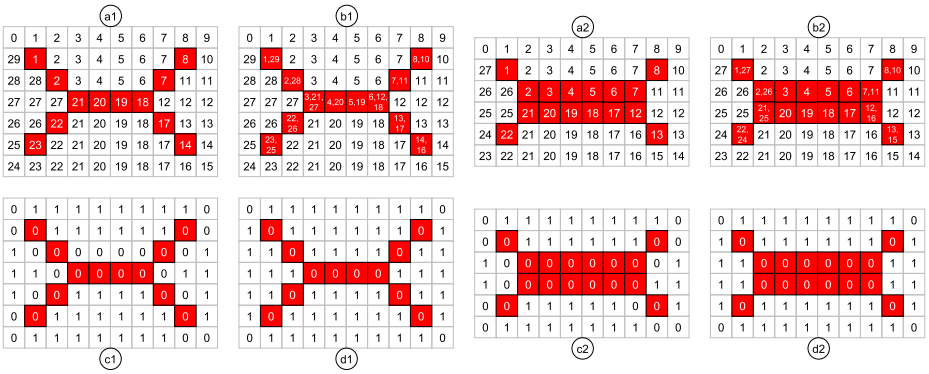


Figure 3.19: Visualization of inheritcount for two shapes (left: odd height, right: even height). For both shapes: (a) simple FT (one closest boundary point per pixel), (b) multiple closest boundary points per pixel, (c) = inheritcount for (a), (d) = inheritcount for (b)

3.8.3 Analysis

In fig. 3.19 we show the inheritcount for the same shapes as fig. 3.8. For both rectangles (one of even height, one of odd height), the closest boundary points are shown in the top images. For the images on the left, only one closest boundary point is stored per point and for the images on the right we store all closest boundary points. The images on the bottom depict the inheritcount for the image directly above.

For the images on the left, we see that the inheritcount is 0 for all skeleton points, but also for some non-skeleton points. For the images on the right, we see that only skeleton points yield a inheritcount of 0. Some points on the boundary also yield a value of 0; whether or not we these classify as skeleton points is a matter of convention.

The example figures in figs. 3.20a and 3.20b and figs. 3.21a and 3.21b show the actual inheritcount metric for two shapes for both the simple FT and the full FT. We can see that the inheritcount metric by itself does not always reveal all skeleton points, even when using the full FT.

We observed that these missing skeleton points relate to regions where “soft” collisions occur, i. e. the region around the stem of the leaf in figs. 3.21a and 3.21b. It is

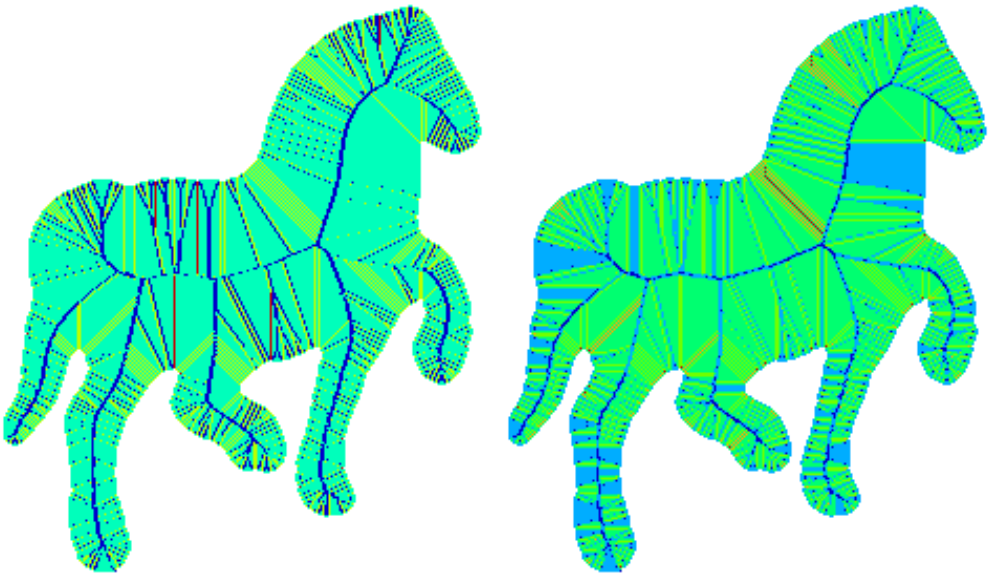
clear that these regions yield points having non-adjacent feature points, which is how both the AFMM Star and our method detects skeleton points.

When we consider the difference between $T(p)$ and $T(q)$ where $q \in N_8(p)$ for the cases where p obtains ("inherits") a closest boundary point from q, we can expect these differences to be 1 when we move in the direction of the inward surface normal. For points on or near the skeleton of the shape, a difference that is less than 1 can be found. This exception does not always apply, which is clearly visible, but if we define:

$$\delta T(p \in \Omega \setminus \partial\Omega) = \sum T(p) - T(q) \quad \text{where } q \in S^a(p) \tag{3.21}$$

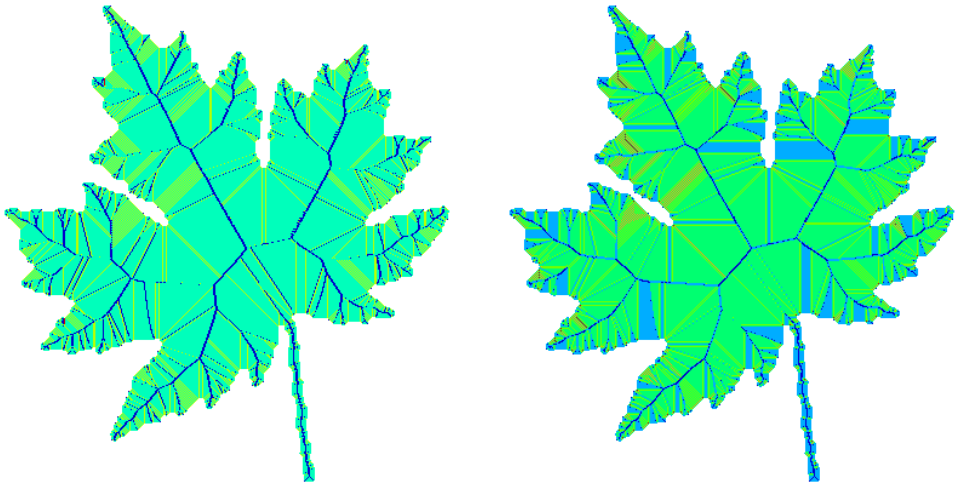
where $S^a(p)$ is defined in eq. (3.19), we get an interesting field in which low values appear to be highly correlated to positions where a skeleton would logically be expected. More interestingly, these figures figs. 3.22a and 3.22b are visually very similar to figures that depict the difference between the approximate FMM distance and the exact Euclidean distance (fig. 3.22c).

A particular use for this field might be to improve the accuracy of the FMM DT, as the field seems to identify regions where large errors occur.



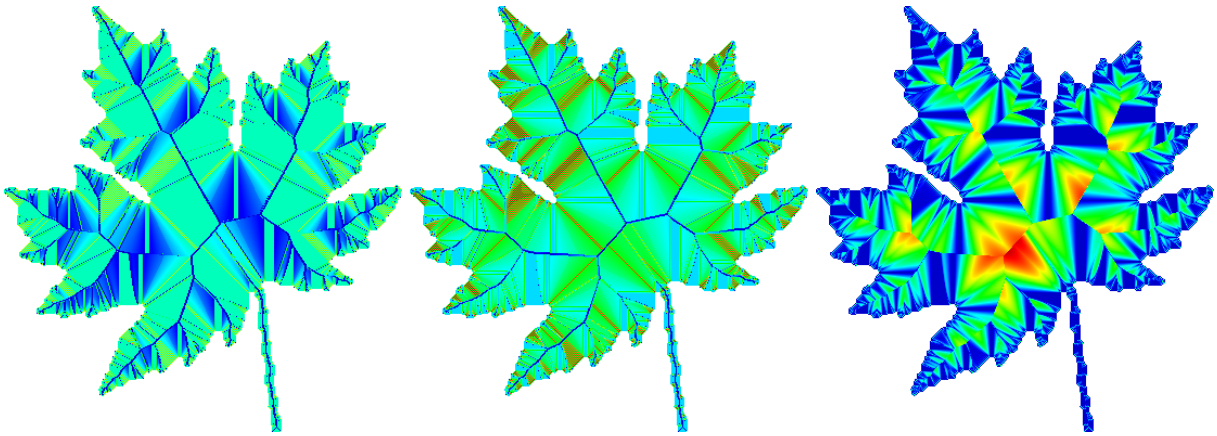
(a) (b)

Figure 3.20: Inheritcount example for `anim4` shape using the simple FT (left) and the full FT (right)



(a) (b)

Figure 3.21: Inheritcount example for `leaf1` shape using the simple FT (left) and the full FT (right)



(a) (b) (c)

Figure 3.22: Result of eq. (3.21) for simple FT (left) and full FT (center), difference between exact DT and FMM DT (right)

4 Results (2D shapes)

In this section, we compare the output of our method with the output of the AFMM Star in order to establish a measure of similarity. We propose an approach to compute such a measure with a reasonable accuracy, we detail on the resulting differences and provide several visual comparisons between the outputs of both methods.

4.1 Comparison to AFMM Star

In this section, we compare our method with the AFMM Star in terms of several algorithmic properties.

Skeleton The AFMM Star produces a skeleton that is centered, 8-connected, smooth, generally 1 point thick and computed reliably. The skeleton produced by our method is roughly the same, but is 4-connected and not as smooth. As a side-effect of computing a 4-connected skeleton, our method also eliminates some skeleton branches in areas where many skeleton branches meet. The impact of this is illustrated in fig. 4.1.

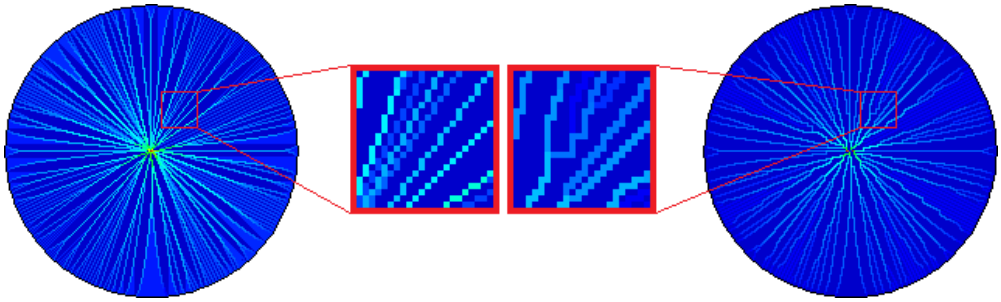


Figure 4.1: Left: AFMM Star, right: our method. $\sqrt{\rho}$ is used instead of ρ to improve contrast for low values of ρ.

Importance measure Both methods compute roughly the same importance measure, small differences being mostly accounted for by local variations in the skeleton, slightly different boundary numberings, inaccuracy of the DT and/or discretization artifacts. The most notable difference is the manner in which the importance measure is computed. The AFMM Star uses a boundary numbering scheme for which a 3D equivalent is not known, but the incremental approach of our method has the potential to generalize to 3D and even higher dimensions.

Computational complexity Our 2D method is a modified implementation of the AFMM Star, using a slightly different boundary numbering scheme and replacing the boundary distance trick with a incremental approach. The overall complexity of our method is $O(\Omega \log \partial\Omega)$, which is the same as the AFMM Star.

In more detail, one could argue that the AFMM Star uses the boundary distance trick to reduce the cost of computing a geodesic to $O\big(\partial\Omega + \mathcal{S}(\Omega)\big)$. Our method traverses the skeleton in order of increasing importance, resulting in a complexity of $O\big(\partial\Omega + \mathcal{S}(\Omega) \log \mathcal{S}_{\mathcal{E}}(\Omega)\big)$ for computing the geodesics, where $\mathcal{S}_{\mathcal{E}}(\Omega)$ is the number of endpoints of the skeleton.

4.2 Result comparison

In this section, we compare the output of our method versus the output of the AFMM Star and establish a measure of similarity between the results of the two methods.

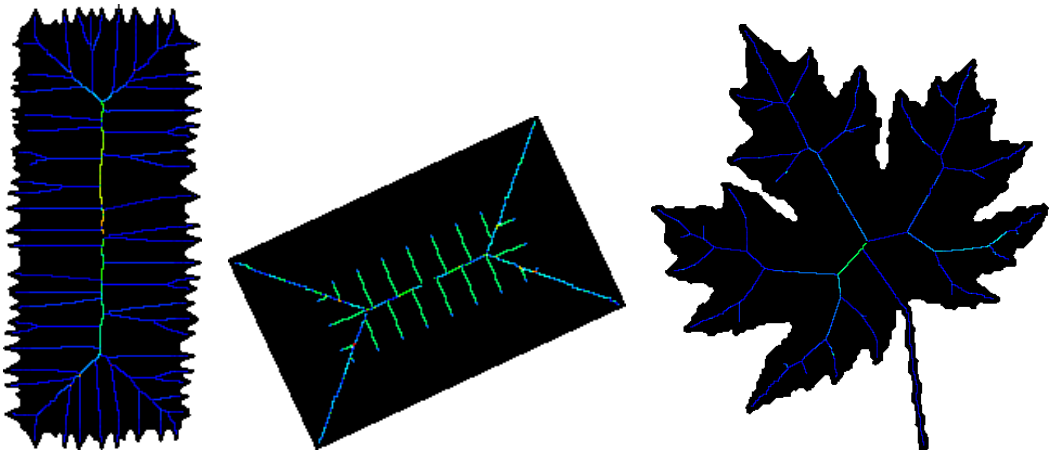
Visual comparison The AFMM Star uses a directional gradient to detect the skeleton, which skews the skeleton towards the bottom and the right. This results in a skeleton that consists of both 4-connected and 8-connected segments. Our method does not use such a gradient and uses 4-connectivity everywhere, which complicates comparing the skeletons in an exact manner.

For the skeleton itself, we simply visually confirmed the similarity. For all images for which we performed this comparison, it is apparent that the skeleton of one method closely resembles a skewed version of the other. To compare the importance measures of the skeletons $\mathcal{S}^a$ and $\mathcal{S}^b$, we use eq. (4.1). The idea behind this comparison is that for each skeleton point in $\mathcal{S}^a$ we should find a skeleton point in $\mathcal{S}^b$ at roughly the same coordinates having a roughly equivalent importance value. Here we denote $\mathcal{S}^a$ for the skeleton produced by our method and $\mathcal{S}^b$ for the AFMM Star skeleton.

$$\begin{gathered}\delta\rho(\mathcal{S}^a, \mathcal{S}^b) = \Big\{ (p, q, \delta_{pq}) \;\Big|\; r \in (N_8(p) \cup p) \wedge q = \operatorname*{argmin}_r \delta(p, r) \wedge \delta_{pq} = \min_r \delta(p, r) \Big\} \\ \text{where } p \in \mathcal{S}^a(\Omega) \wedge q \in \mathcal{S}^b(\Omega) \wedge \delta(p, r) = \big|\rho(p) - \rho(r)\big|\end{gathered} \tag{4.1}$$

As an artifact caused by discretization, it can happen that a skeleton yields two adjacent center skeleton points p, q for which $\rho(p) \simeq \rho(q) \simeq \frac{1}{2}|\partial\Omega|$ as opposed to having one point r for which $\rho(r) \simeq |\partial\Omega|$. We choose to avoid these cases by simply excluding points $p \in \mathcal{S}^a \cup \mathcal{S}^b$ from the comparison when $\rho(p) > 0.99\rho_{max}$ where $\rho_{max} = \max\big(p \in \mathcal{S}^a(\Omega) : \rho(p)\big)$. Finally, we compute both $\delta\rho(\mathcal{S}^a, \mathcal{S}^b)$ and $\delta\rho(\mathcal{S}^b, \mathcal{S}^a)$ and take for each point the minimum of the two delta values. The results for the `jaggedrect`, `rect1` and `leaf1` shapes are shown in fig. 4.2.

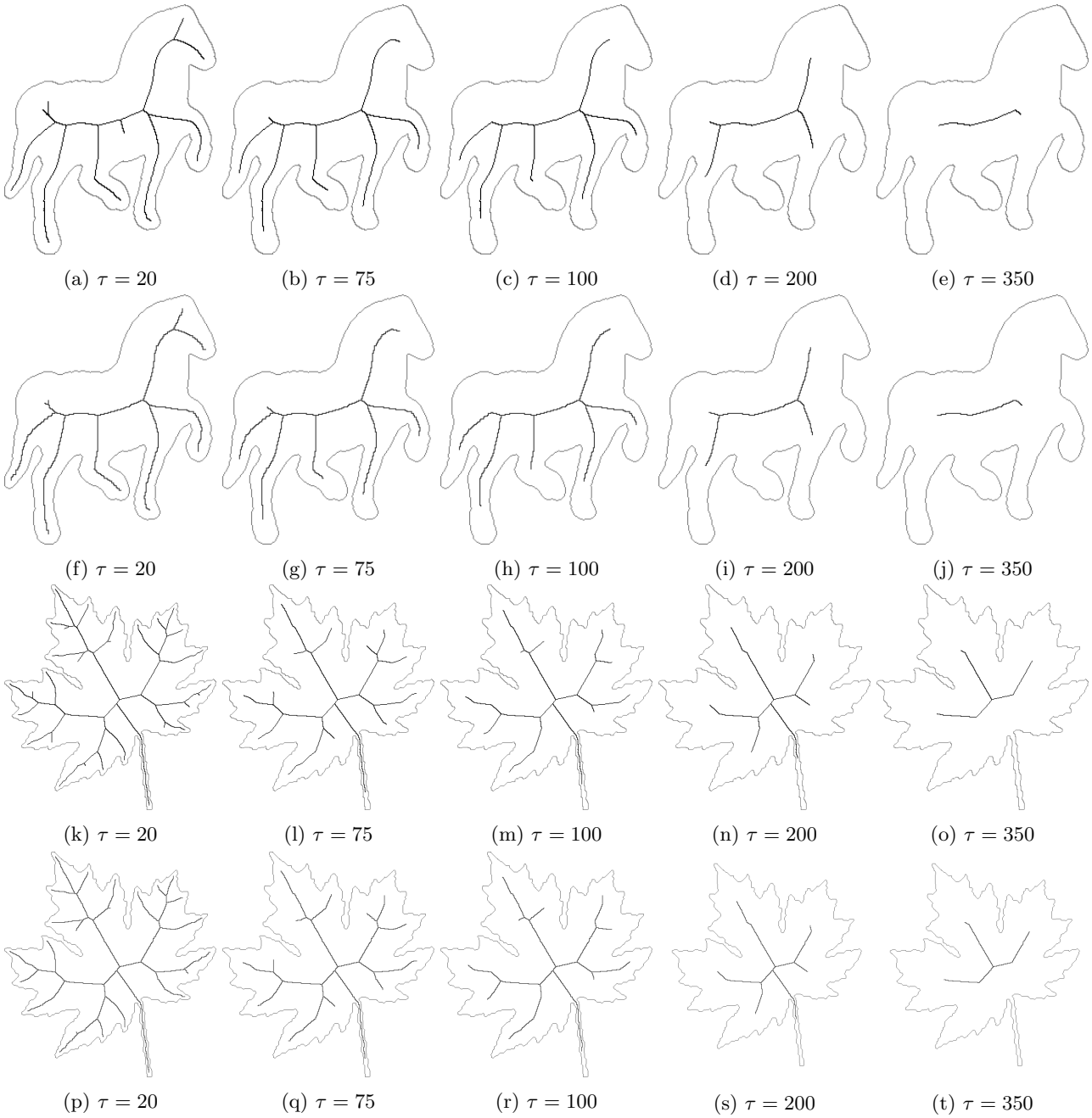
Generally, the greatest difference is no more than 2% of the total boundary length $|\partial\Omega|$ of the shape and only very few points yield such a great difference. Points with a high difference value are always located near the center of the skeleton or near a junction, which is where we would expect them. The average difference is generally less



(a) `jaggedrect` (b) `rect1` (c) `leaf1`

Figure 4.2: Differential images between our method and AFMM. Colors depict values $\delta\rho \in$ [blue...green...yellow...red]. (a) $\delta\rho \in [0 \dots 30], |\partial\Omega| = 1447$. (b) $\delta\rho \in [0 \dots 5], |\partial\Omega| = 721$. (c) $\delta\rho \in [0 \dots 26], |\partial\Omega| = 2485$.

than 0.2% in terms of $|\partial\Omega|$. In section 4.2 we compare for two shapes the thresholded skeleton as produced by the AFMM Star and our method. Both methods yield a nearly identical result. As such, we can conclude that the skeletons are very similar and compute roughly the same importance measure.



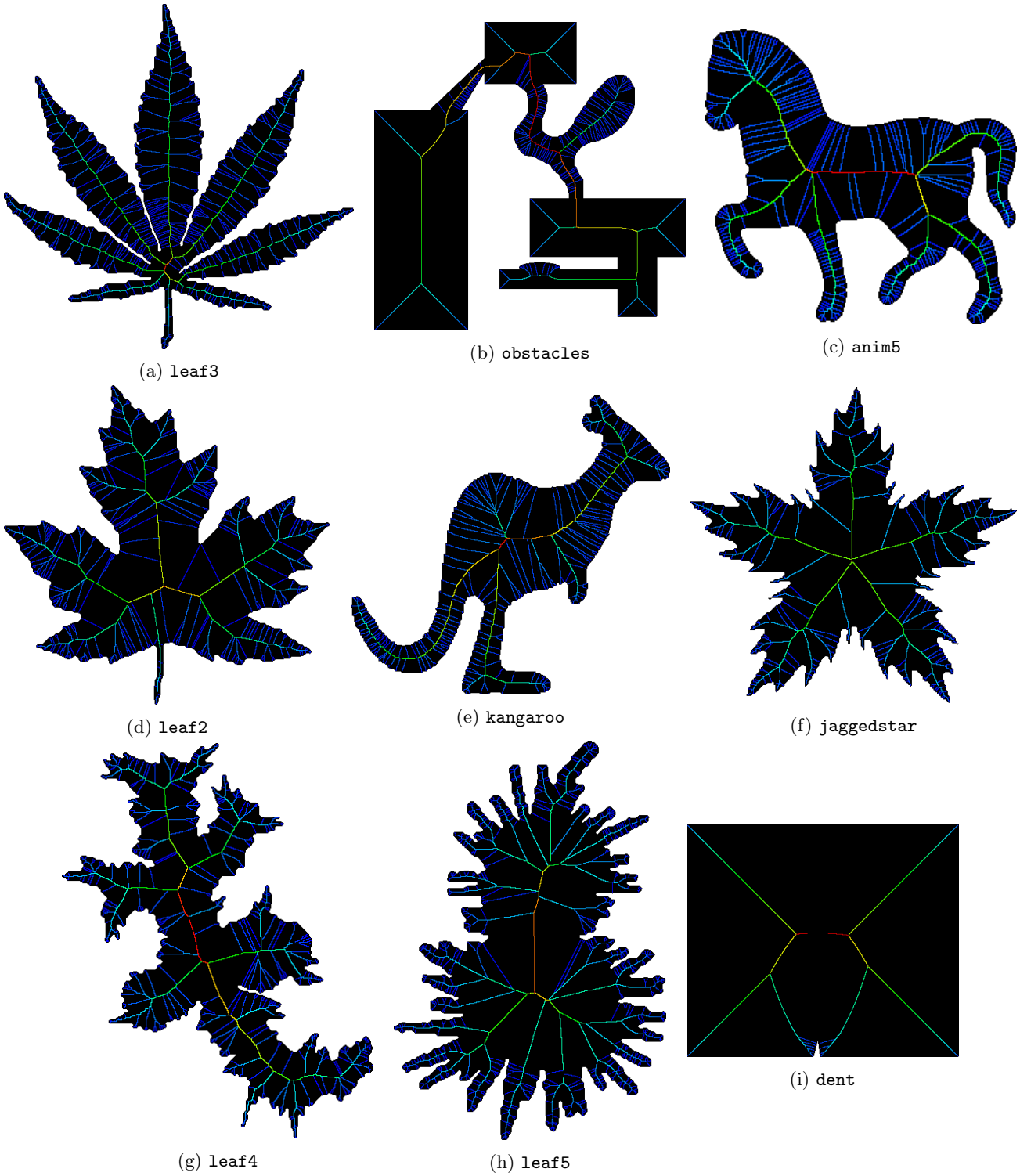
(a) $\tau = 20$ (b) $\tau = 75$ (c) $\tau = 100$ (d) $\tau = 200$ (e) $\tau = 350$

(f) $\tau = 20$ (g) $\tau = 75$ (h) $\tau = 100$ (i) $\tau = 200$ (j) $\tau = 350$

(k) $\tau = 20$ (l) $\tau = 75$ (m) $\tau = 100$ (n) $\tau = 200$ (o) $\tau = 350$

(p) $\tau = 20$ (q) $\tau = 75$ (r) $\tau = 100$ (s) $\tau = 200$ (t) $\tau = 350$

Figure 4.3: Result comparison for thresholded skeleton of `leaf1` and `anim4` shapes. First and third row: AFMM Star, second and fourth row: our method

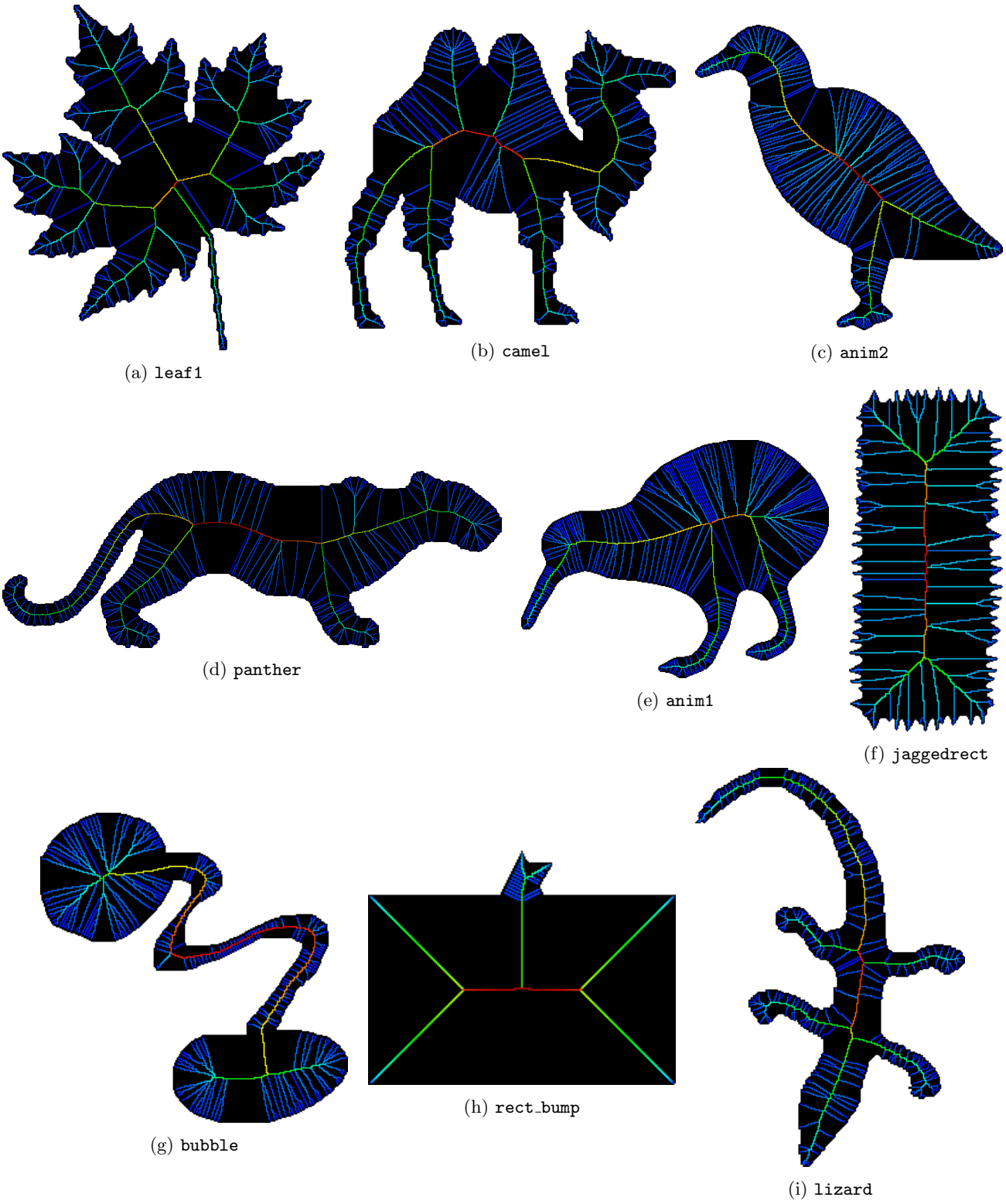


(a) `leaf3` (b) `obstacles` (c) `anim5`

(d) `leaf2` (e) `kangaroo` (f) `jaggedstar`

(g) `leaf4` (h) `leaf5` (i) `dent`

Figure 4.4: Example figures

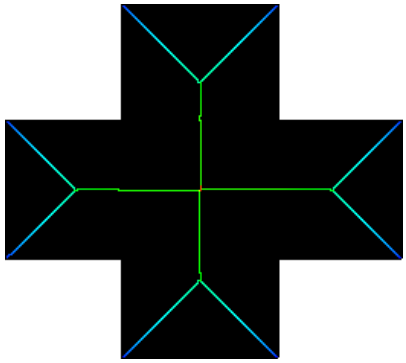


(a) `leaf1` (b) `camel` (c) `anim2`

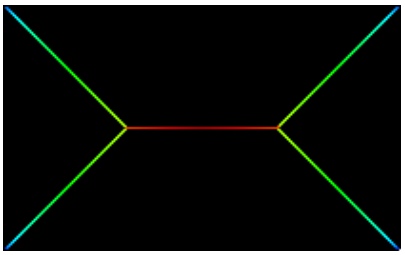
(d) `panther` (e) `anim1` (f) `jaggedrect`

(g) `bubble` (h) `rect_bump` (i) `lizard`

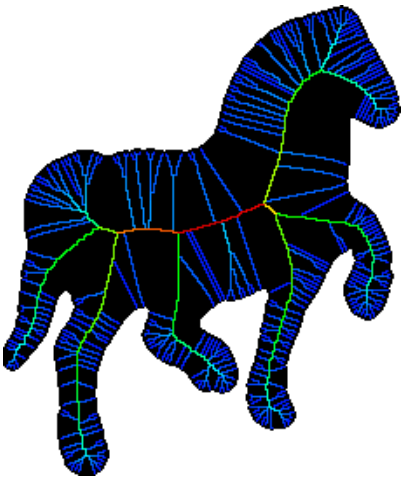
Figure 4.5: Example figures



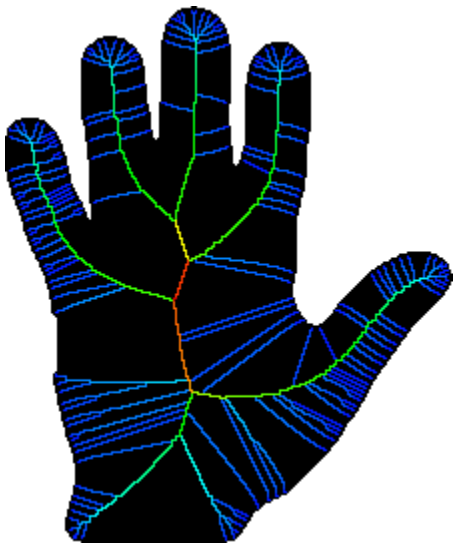
(a) `cross`



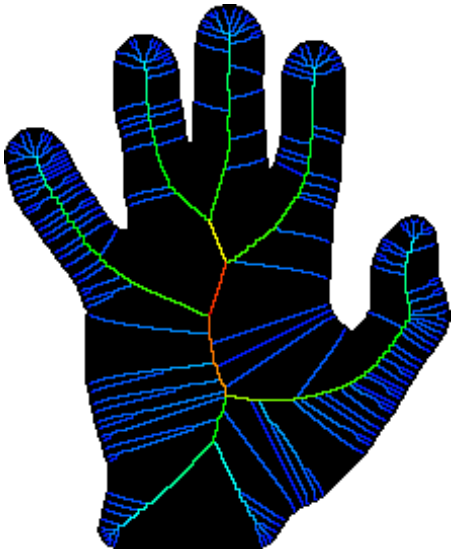
(b) `rect`



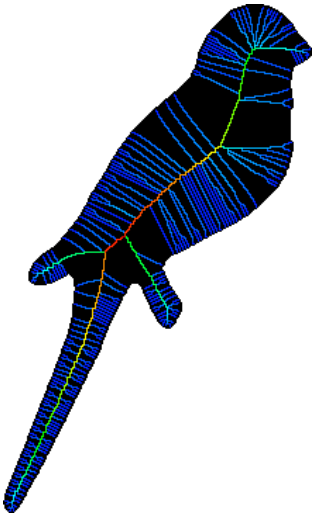
(c) `anim4`



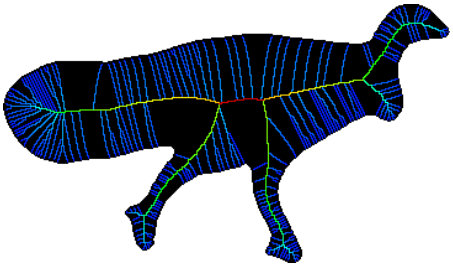
(d) `anim6`



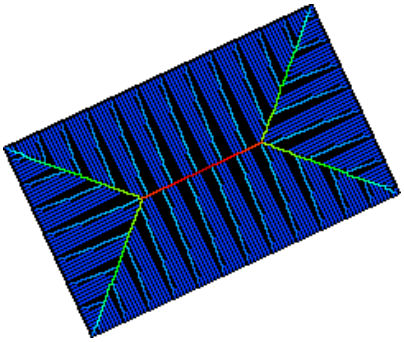
(e) `anim7`



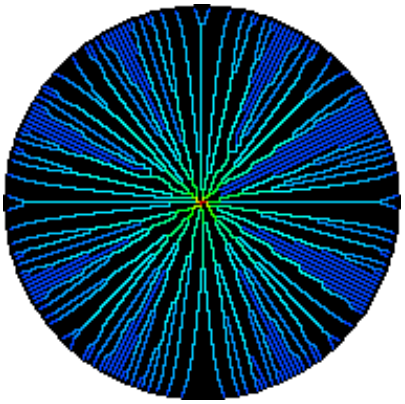
(f) `anim3`



(g) `anim0`



(h) `rect1`



(i) `circle`

Figure 4.6: Example figures

5 3D Proposal

In this chapter, we present a method that represents the 3D generalization of the method presented in chapter chapter 3. The main focus of this chapter is to convey the manner in which our 2D proposal can be used to provide an equivalent solution in 3D. In section 5.1 we provide an overview of the 3D method as well as differences and similarities with respect to our 2D proposal. In section 5.2 we detail on these similarities and provide a working implementation that computes the 3D equivalent of the 2D AFMM Star skeleton. In section 5.3 we detail on the 3D importance measure and propose a conceptual method to compute such a measure in the desired manner.

5.1 Overview

In this section, we present an overview of several difficulties and considerations relating to generalizing our 2D approach to 3D. We focus on the intuitive explanation of our approach and present a conceptual model / framework for the desired implementation. We express components in this model / framework in terms of our 2D proposal by either copying the 2D approach directly or providing a 3D equivalent replacement. For the contents detailed in this section, a working implementation is provided that resembles the 3D equivalent of the 2D skeleton as presented in chapter 3.

5.1.1 Phase 1

This phase represents the 3D generalization of phase 1 in our 2D proposal and we provide a working implementation of this phase.

Step 0: Initialize flags, narrowband, boundary (section 5.2.1) This step is a direct generalization to 3D of step 0 in our 2D proposal.

Step 1: Compute DT+FT, detect skeleton Like the previous step, this step is also a direct generalization to 3D of step 1 in our 2D proposal.

Step 2: Remove cycles, perform thinning Since we detect the skeleton as a 6-connected structure, the 2D regularization step generalizes directly to 3D. The result is that regardless of the manner in which the skeleton is computed, a 6-connected set of points can always be reduced to a 1 point thick skeleton.

5.1.2 Phase 2

This phase is a more general form of phase 2 in our 2D proposal. In this phase, the goal is to provide a collection of methods and viewpoints for which we can argue that they compute the desired metric with the desired efficiency. As we do not have a working implementation for this phase, we mainly focus on conveying the details and feasibility of the suggested approach.

Step 3: Initialize skeleton FT, boundary FT (section 5.4) The boundary is now a 2D structure (versus being 1D in our 2D proposal), complicating the handling of boundary collapse events due to the different topological properties of 2D and 1D surfaces. We propose to solve this by introducing an additional skeleton FT and boundary FT step that reduces the 2D boundary and 2D skeleton to a 1D subset of these boundary- and skeleton points, which in turn can then be processed again as in our 2D proposal. This process is depicted in the following substeps.

Step 3.0: Initialize flags, narrowband, boundary For both the skeleton FT and the boundary FT, the purpose and result of the initialization step is the same as for the initialization step in our 2D proposal. The skeleton FT initialization can be performed similar to the skeleton traversal step in our 2D proposal. For the boundary FT initialization, several approaches seem feasible and are discussed in section 5.4.

Step 3.1: Compute skeleton FT, boundary FT Computing the FT of the (surface) skeleton yields the curve skeleton, whereas computing the FT of the boundary can yield a skeleton of the boundary of the shape. The skeleton of the boundary, which is a subset of $\partial\Omega$, may not be a measure that is required for computing the multiscale importance measure. Regardless of whether this is the case, it is useful for the purpose of providing a more intuitive explanation of our 3D method.

Step 3.2: Regularization step Given the necessity of a regularization step when computing the 2D and 3D skeleton, it is likely that some kind of regularization step is needed here as well. After this step is completed, the skeleton of the boundary of the shape as well as the curve skeleton will be 6-connected structures.

5.1.3 Phase 3

This phase is similar to phase 2, but operates on the curve skeleton instead of the surface skeleton. A straightforward approach is to construct the same Jordan curves that are used by Reniers et al.[33], which would be complicated but feasible given a successful implementation of phase 2. However, due to the inherent topological properties of the curve skeleton and its feature points, we may be able to apply several simplifications ("tricks") which we already use in our 2D proposal.

Since our goal is to reproduce the results of Reniers et al. while reducing the computational complexity, it makes sense to employ these simplifications where possible. In

order to convey the similarity between the simplified approach and the [33] approach, additional focus is put on comparing these approaches in an informal manner.

Step 3.3: Initialize skeleton traversal We take the 2D reference counting approach and we apply it to the curve skeleton of the shape in exactly the same manner.

Step 4: Traverse skeleton, collapse boundary The result of step 3.3 is that points on the boundary of the shape will yield higher referencecount values if these points coincide with the skeleton of the boundary. If we have computed the skeleton of the boundary, we can implement this step analogous to the same step in our 2D proposal. If we have not computed the skeleton of the boundary (or for some reason can not readily do so), we can apply the followband approach to our skeleton traversal step, which should allow us to compute the skeleton of the boundary during the skeleton traversal step.

5.2 Generalized 2D methods

In this section, we list the methods that are straightforward generalizations to 3D of the methods used in our 2D proposal.

5.2.1 Boundary modeling

We model the boundary of the 3D shape using multiple instances of our 2D boundary model. We consider each of the 3 axis-parallel slices through a boundary point. We treat each slice as a 2D image and apply the 2D numbering scheme to each. For a slice through the XY plane, we "look" at the slice in the Z_+ direction so as to have a consistent definition of clockwise and counterclockwise (see fig. 5.1).

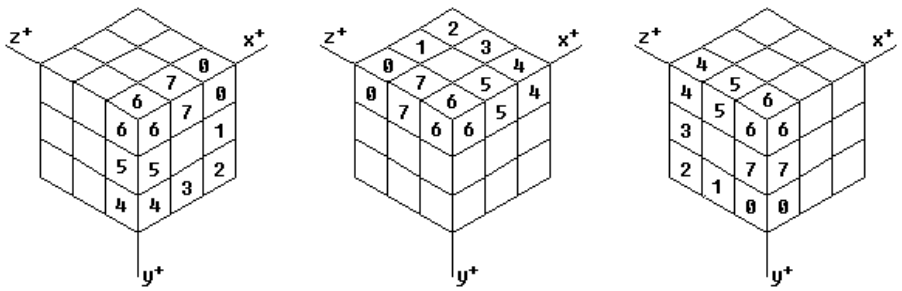


Figure 5.1: Order in which we sweep adjacent points. Left to right: XY, ZX, YZ plane.

The result of this is that every voxel has either two or three labels, depending on the topology of the boundary. The reason that points can have only two labels is visible in fig. 5.1: for the XY plane, the point in the middle of the Z+ face (as well as the Z- face) does not lie on the boundary of the 2D slice.

The purpose of these individual labels is to designate two particular adjacent boundary points for a given planar slice direction. Each voxel is also assigned a unique label to link together the planar slice labels. This unique label can then be used to visualize the FT as done in the 2D proposal (fig. 3.3a,b on page 24).

5.2.2 Followband

For the 3D case, the implementation for the followband follows directly from the 2D case (section 3.3) by using 6-connectivity instead of 4-connectivity.

5.2.3 Skeleton

For the 3D case, the skeleton is represented as a collection of face(6)-connected voxels, which is the 3D equivalent of the edge(4)-connected pixels used for the 2D case. Detecting the skeleton is done by generalizing the 2D detection method to 3D, which is done by replacing the 8-connectivity (edge- and vertex connected neighbours) in eq. (3.6) on page 35 by 18-connectivity (face- and edge-connected neighbours).

Since we use 6-connectivity to represent the skeleton, the 2D cycle detection section 3.4.3 approach also applies to the 3D representation. Thinning the skeleton is done by applying the 2D thinning measure to each of the 3 planar slices through a skeleton point p, and p is simple iff p is a simple point w.r.t each of these planar slices. An example of these slices and their corresponding numerings is shown in fig. 5.1.

5.2.4 Inheritcount

The inheritcount measure is a direct generalization of its 2D counterpart. As in our 2D proposal, the inheritcount measure is mostly a visualization aid and debugging tool and yields results that are visually very similar.

5.2.5 Skeleton DT

The 3D skeleton DT is similar to the 2D skeleton DT, and differs mainly in operating on a 2D skeleton in 3D space versus operating on a 1D skeleton in 2D space. Like the 2D skeleton DT, the 3D skeleton DT differs from the FMM DT in the sense that there is no ordered upwind method to compensate for temporary values. The reason for this is that the skeleton DT relies on a boundary DT (which is discussed in the next section) to determine the processing order.

For the 2D skeleton DT, we do not need to explicitly precompute this boundary DT and instead use the same topological properties which are also utilized in the AFMM Star. For the 3D skeleton DT, we encounter the same kind of issue that disallows the AFMM Star to generalize to 3D: the topological properties of 3D shapes do not allow us to get away with the 2D-only "trick". It is thus necessary to explicitly compute the boundary DT in order to perform the 3D skeleton DT.

If we assume that such a boundary DT exists and can be computed, then we can see that the shape FT readily allows the required $skeleton \rightarrow boundary$ mapping. If

we denote $T_\partial(b)$ as the DT-value for $b \in \partial\Omega$, then the order in which $p \in \mathcal{S}^{\mathcal{S}}(\Omega)$ are visited can be expressed as a function of $T_\partial(b)$ where $b \in F(p)$.

The above implies that the skeleton DT itself is the same for 2D and 3D shapes, differing only in the manner in which the importance value for a point is computed obtained. Unsurprisingly, this process is more involved for the 3D case, but the basic approach is the same.

5.2.6 Skeleton FT

For the 3D skeleton FT we can copy the 2D approach and use the boundary labels supplied by the boundary model. All we require is a way to identify points that are adjacent on the boundary, which the boundary model is able to supply. As such, the 3D skeleton FT is similar to the 2D skeleton FT.

5.3 Importance measure

In this section we present several ideas and observations regarding a key aspect of our proposal: reducing the computational complexity of the boundary distance computation. We do not have a working implementation, as this is beyond the scope and available time for this thesis. We base our approach on our 2D proposal, which provides the framework which reliably computes skeletons for both 2D and 3D shapes.

In section 5.3.1 we analyze our 2D solution and identify the changes that are required in order to provide a equivalent solution in 3D. In section 5.3.2 we present our suggested approach, and in section 5.3.3 we detail on the feasibility and implications of this approach.

5.3.1 Problem description

For the 2D case, the boundary of the shape yields a geodesic for every skeleton endpoint. Since the boundary of a 2D shape is 1D, multiple 1D geodesics are easily concatenated into one geodesic by adding up the lengths of the individual geodesics. As a result, there is no need for our 2D approach to ensure that boundary points collapse in a particular order. For example, consider a geodesic spanning boundary points $[n \dots m]$ and consider that boundary points $m+1$, $m+2$, $m+3$, $m+4$ and $m+5$ are the next boundary points to collapse. Regardless of the order in which we collapse these points, concatenating the geodesics $[n \dots m]$, $[m+1 \dots m+2]$ and $[m+3 \dots m+5]$ into one geodesic $[n \dots m+5]$ is trivial.

As such, we do not need to concern ourselves with a optimal order of collapsing these boundary points (append $m+1$ to $[n \dots m]$, then append $m+2$, and so on), which is key to keeping our 2D approach simple.

When we try to generalize the above approach to the 2D surface of a 3D shape, this simple approach is no longer feasible. Collapsing boundary points out-of-order no longer guarantees that we can concatenate them in the same care-free manner due to the different topological properties of 2D surfaces.

We effectively walk into a kind of "trap" that relates closely to why the AFMM Star cannot readily generalize to 3D. We simplify our 2D implementation by relying on the topological properties of 1D surfaces, and unsurprisingly we find it insufficient for the 3D case: concatenating two geodesics in the desired manner is no longer a trivial task.

5.3.2 Suggested approach

The key to mapping our 2D method to 3D is the realization that our boundary distance computation is a DT over the boundary of the shape with a very particular initial wavefront. If we consider the cubical shape in fig. 5.4a, we know that the surface skeleton of the shape consists of points on the 2D planes defined by two diagonally opposing edges of the cube. The curve skeleton is known to consist of points on the 1D lines defined by each pair of diagonally opposing vertices of the cube.

If we traverse the surface skeleton from the endpoints towards the center of the cube, we can see that the feature points of skeleton points lie at a monotonically increasing distance to the edges of the cube. This process is visualized in fig. 5.2. Similarly, if we traverse the curve skeleton from the endpoints towards the center, we see that the same holds true for the distance towards the vertices of the cube. This process is visualized in fig. 5.3.

For now, assume that this idea holds not only for simple shapes such as the depicted cube, but for any shape in general. We now claim that intuitively there must be a 1:1 correspondance between the order in which points are visited by the boundary DT. An argument supporting this claim is found in the definition for the FT of the 3D shape, which suggests that not having such a 1:1 correspondance conflicts with the definition for the FT.

In the form of an example, we can imagine drawing lines from skeleton points to their feature points as we traverse the skeleton of the shape. Barring minor discrepancies (due to discretization and / or approximation), we must be able to traverse the skeleton from the endpoints towards the center of the skeleton without drawing lines that intersect previously drawn lines. In other words, during the traversal process, there should be a unobstructed "line of sight" between a skeleton point and its feature points, where the previously drawn lines represent obstacles.

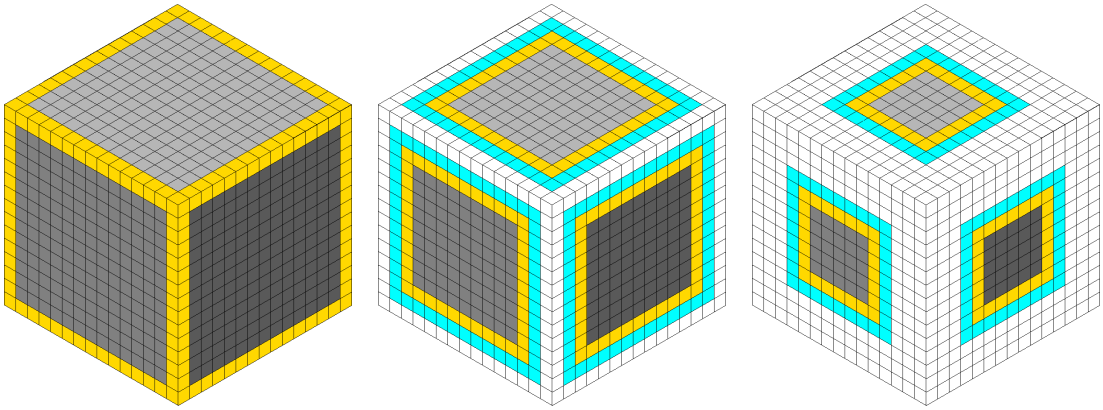
In more formal terms, this "no intersecting" principle can be expressed in terms of the relative timesteps of skeleton points and their feature points. The m points b_v on $\partial\Omega$ and the n points p_t on $\mathcal{S}(\Omega)$ are traversed in a series of $v \in [0 \dots m\rangle$ and $t \in [0 \dots n\rangle$ timesteps respectively. We can now formulate:

$$\forall b_v \in F(p_t) : b_u \in F\big(q_{s<t} \in N_{\perp}^{\mathcal{S}}(p_t)\big) \Rightarrow u < v \vee b_v = b_u \vee F(p_t) \cup F(q_s) = \emptyset \quad (5.1)$$

which translates to saying that the order in which skeleton points are visited must also apply to the relative order in which their feature points are visited. Note that the last term in eq. (5.1) covers the cases where two skeleton points yield the same feature point, which would technically imply intersecting lines. Although this is unnecessary for the continuous case, we clearly need such an exception for the discrete case.

In terms of implementation steps, the goal is now to define and compute a boundary DT / FT which we can then use to traverse the skeleton in the desired order, using the

FT of the shape as the mapping between the skeleton and the boundary. Assuming that we can express such a boundary DT / FT as a variant of the AFMM Star that operates only on $\partial\Omega$, we can then use the "no intersecting" principle to argue that the AFMM Star invariants (and thus also the FMM invariants) for the boundary DT / FT can also be made to apply to the 3D skeleton DT / FT.



(a) (b) (c)

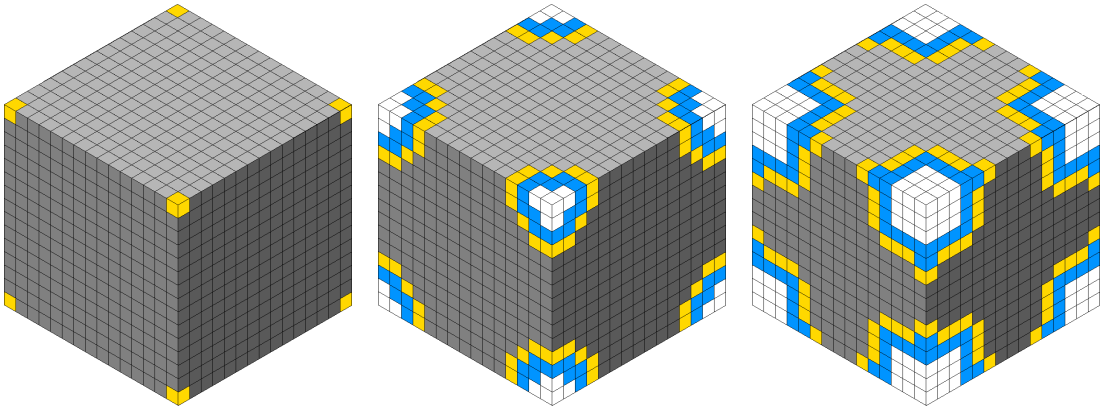
Figure 5.2: Depiction of a boundary FT for several timesteps, using $\partial\partial\Omega$ for the initial wavefront (first image). `NARROWBAND` (yellow), followband (blue), `KNOWN` points (white), `UNKNOWN` points (grey).

5.3.3 Implications of suggested approach

In this section we summarize the implications of our suggested approach in the form of several assertions. These assertions form the basis for claiming that our method is plausibly feasible, can produce the correct results and can plausibly do so with the claimed computational complexity.

Decomposition of the problem set We decompose our problem into subproblems in such a way that we can apply the idea behind the advection model to both the main problem as well as each of the subproblems. We argue that each of these subproblems reduces the computational complexity of the problem by reducing its dimensionality by 1 and we argue that this approach must yield a optimal complexity due to the formal definition of the skeleton and due to the manner in which distances are computed.

Complexity of the resulting method We argue that a method with the desired complexity is theoretically feasible by not requiring the computation of all possible shortest paths, but merely computing the length of such a shortest path. Since it follows from the definition of the Blum skeleton that such an initialization must exist,



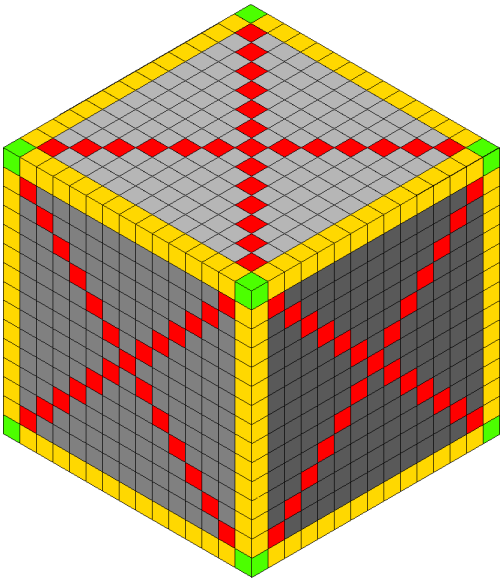
(a) (b) (c)

Figure 5.3: Depiction of a boundary FT for several timesteps, using $\partial\partial\partial\Omega$ for the initial wavefront (first image). `NARROWBAND` (yellow), followband (blue), `KNOWN` points (white), `UNKNOWN` points (grey).

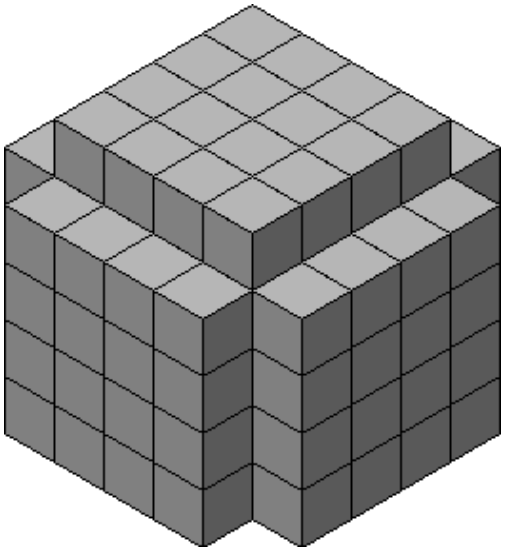
we only need to find one particular instance out of all possible initializations of $\partial\partial\Omega$ that is "close enough" to the optimal case.

This reduces the problem to a initial value problem, where "initial value" translates to identifying the position of the desired collection of points. We assert that this initial value problem is a form that can readily be solved by our (or some other variation) of the AFMM Star method. We thus assert that the feasibility of this initial value problem depends only on our ability to correctly and efficiently identify these points.

Feasibility of computing the initial points and their initial values We argue that since our method can identify skeleton points reliably and accurately, initial values can be computed reliably and accurately as well due to the correspondence between the initial-value points and the endpoints of the skeleton.



(a) Visualization of 3D point set interpretations. Yellow + green: $\partial\partial\Omega$. Grey + red: $\partial\Omega \setminus \partial\partial\Omega \equiv \text{proj}(\mathcal{S}(\Omega), \partial\Omega)$. Red: $\mathcal{S}(\partial\partial\Omega) \equiv \text{proj}\left(\mathcal{S}\big(\mathcal{S}(\Omega)\big), \partial\Omega\right)$.



(b) Cubical shape where points on the vertices and edges of the cube have been pruned.

5.4 Boundary FT

The boundary FT is an approach that works similar to the FT method that is used to compute the skeleton. The boundary FT operates on $\partial\Omega$, has a initial wavefront $\partial\partial\Omega \subset \partial\Omega$ and computes for each $b \in \partial\Omega$ the feature points $b_\partial \in F_\partial(b \in \partial\Omega)$.

Details on how to compute $\partial\partial\Omega$ are given in the next section, in this section we simply assume it is valid and can be computed. The purpose of the boundary FT is to define and compute a metric on the boundary of the shape which, via the FT of the shape, allows us to traverse the skeleton and compute the boundary-collapse measure in an incremental fashion.

We use this particular initial wavefront to compute the boundary FT in the same way as our 2D and 3D DT / FT method. The idea is that each point b on $\partial\Omega$ now has a feature point on $\partial\partial\Omega$, which we denote by $b_\partial \in F_\partial(b)$. We consider $p \in \mathcal{S}^{\mathcal{S}}(\Omega)$ and we now see that the geodesic between $b, c \in F(p)$ must always contain some point $b_\partial \in \partial\partial\Omega$. The reason for this is that b and c are by definition not adjacent on the boundary, which follows from p being a skeleton point. As such, some b_∂ must then always exist for any $p \in \mathcal{S}(\Omega)$.

More precisely, if we consider $b_\partial \in F_\partial(b)$, we see that $\partial\partial\Omega$ should lie (approximately) halfway such a geodesic. Hence when b_∂ is a feature point (over the boundary) of b, c and when b, c are feature points of a skeleton point p, then b, c should be equidistant to b_∂.

We thus use this idea to express the geodesic as two individual components which are of equal length in the continuous case (or approximately of equal length in the discrete case). If we now define the Euclidean distance over the surface, analogous to eq. (2.2) on page 11, as

$$D_\partial(b \in \partial\Omega) = \min_{b_\partial \in \partial\partial\Omega} ||b - b_\partial|| \tag{5.2}$$

where $||\cdot||$ denotes the Euclidean distance over the surface, then we can form the expression

$$\rho(p \in \mathcal{S}(\Omega)) \approx D_\partial(b) + D_\partial(c) \tag{5.3}$$

which reflects the idea that $\rho(p \in \mathcal{S}(\Omega))$ is the sum of at least two partial geodesics. Each of these partial geodesics can be computed using the boundary FT, given that we can identify $\partial\partial\Omega$. This would mean that computing the length of all these partial geodesics can be done in $O(\partial\Omega \log \partial\partial\Omega)$ time.

Identifying which of these partial geodesics combine into a "minimal" geodesic for a given skeleton point p can then be done in the same manner as the AFMM Star at a cost of $O(\mathcal{S}(\Omega))$. Note that there is a hidden constant: since $\#\{F(p)\} \geq 1$ and $\#\{F_\partial(b_\partial)\} \geq 1$, all possible combinations of partial geodesics will need to be computed. Fortunately, the number of feature points is generally low and will thus have a limited performance impact.

5.4.1 Interpreting and defining $\partial\partial\Omega$

Factors such as discretization artifacts, processing order and DT inaccuracies make it difficult to formulate an exact definition for $\partial\partial\Omega$. Given that we have a skeletonization method that appears accurate and reliable for both 2D and 3D shapes, it is reasonable to define $\partial\partial\Omega$ as a function of the computed skeleton:

$$\begin{gathered}\partial\partial\Omega = \left\{ b \in F(p) \;\middle|\; isAdjacent(b, b_\partial) \right\} \\ \text{where } p \in \mathcal{S}_\mathcal{E}(\Omega) \wedge b_\partial \in F_\partial(p)\end{gathered} \tag{5.4}$$

To better understand which collection of points are described by eq. (5.4), we now provide several formal and semi-formal interpretations. Aside from conveying what is being computed here and whether it is feasible to do so, these interpretations also serve to explain a key component in this thesis: why it is at all possible to achieve the desired computational complexity for our importance measure.

Note that these interpretations are expressed using what is probably the most simple shape to do so in order to best convey the underlying ideas. Naturally, more complex shapes are likely to introduce cases for which further analysis is required. This is one of the main reasons that we do not have an implementation for the contents of this section.

(a) Interpreting $\partial\partial\Omega$ as boundary features For the simple shape in fig. 5.2a it is easily verified that $\partial\partial\Omega$ consists of points on the edges and vertices of the cube. In practice, various artifacts and inaccuracies require that we also consider the points on $\partial\Omega$ that are adjacent to these edges and vertices.

(b) Interpreting $\partial\partial\Omega$ as a projection We can also express $\partial\partial\Omega$ in terms of projecting the surface skeleton of the shape onto $\partial\Omega$, which is readily available in the form of the FT. We represent the projection of a point p onto $\partial\Omega$ by

$$\text{proj}\left(p \in \mathcal{S}^{\mathcal{S}}(\Omega), \partial\Omega\right) = F(p) \tag{5.5}$$

we can then formulate

$$\partial\partial\Omega = \partial\Omega \setminus \left\{b \in \text{proj}(p, \partial\Omega) \mid p \in \mathcal{S}(\Omega)\right\} \tag{5.6}$$

which is valid for the continuous case. For the discrete case, discretization artifacts still need to be accounted for. For example, consider fig. 5.4b (page 66) where we strip away the points on the edges and vertices of a cube. For this example, we see that $\partial\partial\Omega = \emptyset$ because the boundary points adjacent to the stripped-away points are feature points of $\mathcal{S}_{\mathcal{E}}(\Omega)$.

We observe that feature points of $\mathcal{S}_{\mathcal{E}}(\Omega)$ should always be adjacent to at least one point in $\partial\partial\Omega$, which follows from the definition of a skeleton point in eq. (3.6). The points in $\partial\partial\Omega$ are the reason that $\mathcal{S}_{\mathcal{E}}(\Omega)$ was detected in the first place. More precisely, at least two feature points of $\mathcal{S}_{\mathcal{E}}(\Omega)$ are not adjacent to one another because a point in $\partial\partial\Omega$ lies in between.

Even though the above description appears to be plausible, it is difficult to directly define $\partial\partial\Omega$ for the discrete case.

(c) Interpreting $\partial\partial\Omega$ as a skeleton If we can compute the boundary FT, we can also assume that we can compute $\mathcal{S}(\partial\Omega)$ as the skeleton on the surface of the shape, using a variation of our skeletonization method. The result is that similar to projecting $\mathcal{S}(\Omega)$ onto $\partial\Omega$, we can now also project $\mathcal{S}^{\mathcal{C}}(\Omega)$ onto $\partial\Omega$. The result would be $\mathcal{S}(\partial\Omega) = \text{proj}(p, \partial\Omega)$ where $p \in \mathcal{S}^{\mathcal{S}}(\Omega)$.

(d) Interpreting $\partial\partial\Omega$ as an opportunity for a numbering scheme Recall that for the AFMM Star method in section 2.8.1, no equivalent for its numbering scheme is known to exist for 3D shapes. Consider a skeleton point $p \in \mathcal{S}(\Omega)$, its feature points $b \in F(p)$ and the feature points $b_\partial \in F_\partial(b)$ computed via the surface DT. Since b_∂ is adjacent to $q \in F_{\mathcal{S}}(p)$, we can define a plane through this collection of points, which by the definitions of $F(p), F_\partial(b \in F(p))$ and $F_{\mathcal{S}}(p)$ should then intersect $\partial\Omega$ along the geodesic of p.

Recall that the numbering scheme used in our 2D proposal effectively provides a binary choice where we can decide between a clockwise and counterclockwise direction. Note that the number of distinct possible directions on the 2D boundary is limited to four due to using a discrete grid. Also note that points on $\partial\partial\Omega$ should by definition

be aligned in a direction that is perpendicular to the direction of geodesics that cross $\partial\partial\Omega$.

We can now see that, barring small discrepancies due to discretization, geodesics do not cross between two points in $\partial\partial\Omega$. As such, we can assign to points on $\partial\partial\Omega$ a unique label which can serve as initial boundary features for the boundary FT.

(e) Interpreting $\partial\partial\Omega$ via a reference counting scheme In our 2D proposal, $\partial\partial\Omega$ are the boundary points that are the first to collapse during the skeleton traversal step (section 3.6. This interpretation is, based on our current understanding, the most reliable and also the most viable interpretation due to its simple implementation and due to having a working proof of concept in our 2D proposal.

(f) Interpreting $\partial\partial\Omega$ as the center of the geodesics Relating closely to interpretation (d), we can consider a geodesic over the boundary and consider a boundary point b which splits this geodesic in half (or if we consider the discrete case, roughly in half due to discretization artifacts).

(g) Interpreting $\partial\partial\Omega$ as the edges of segments of $\partial\Omega$ When we consider a cube, the faces of the cube are the segments and points on the edges and vertices of the cube are the edges of the segments. This relates closely to interpretation (b), where the projection of $\mathcal{S}^{\mathcal{S}}(\Omega)$ onto $\partial\Omega$ yields the segments of $\partial\Omega$. Like (b), no solution is readily available for the simple problem case depicted in fig. 5.4b (page 66).

(h) Interpreting $\partial\partial\Omega$ as the dual of $\mathcal{S}_{\mathcal{E}}(\Omega)$ The AFMM Star computes for every $p \in \Omega$ a closest feature point $b \in \partial\Omega$. These feature points are found in order of monotonically increasing DT value. Since our method detects skeleton points in roughly the same order, we can opt to also compute a reverse FT $\overline{F} : \mathcal{S}(\Omega) \to \partial\Omega$ which yields for every point $b \in \partial\Omega$ a closest skeleton point $p \in \mathcal{S}(\Omega)$. For the example in fig. 5.4a, we can see that $b \in \partial\partial\Omega$ whenever $p \in \overline{F}(b) \wedge b \notin F(p)$ where $b \in \partial\Omega$ and $p \in \mathcal{S}(\Omega)$.

This suggests that $\partial\partial\Omega$ is a kind of dual of $\mathcal{S}_{\mathcal{E}}(\Omega)$, which we might be able to compute by using a variant of the AFMM Star method. If we use $\overline{F}$ for the FT and D_{∂} (eq. (5.2) on page 67) as the DT, we may only need a variation of eq. (3.3) on page 34 to define $\partial\partial\Omega$ as a function of $\mathcal{S}_{\mathcal{E}}(\Omega)$. In other words, $\mathcal{S}_{\mathcal{E}}(\Omega)$ relates to $\mathcal{S}(\Omega)$ in the same way as $\partial\partial\Omega$ relates to $\partial\Omega$, and a certain AFMM Star-like method should be able to detect $\partial\partial\Omega$ in $O(\partial\Omega \log \partial\partial\Omega)$ time.

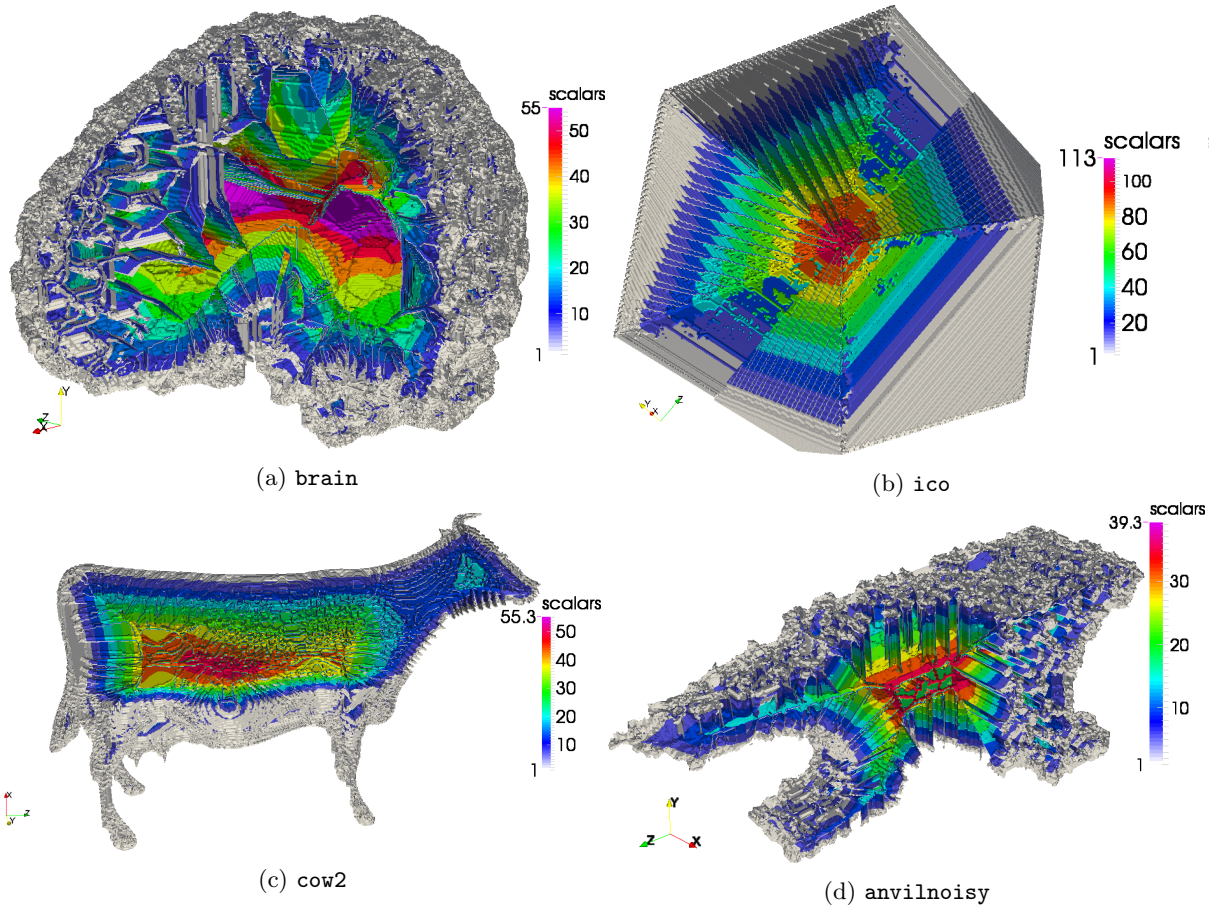
5.4.2 Interpreting and defining $\partial\partial\partial\Omega$

The problem set for the curve skeleton can be implemented in a recursive fashion in correspondance with the anatomy of the skeleton (section 2.4.1). We can define $\partial\partial\partial\Omega$ when we consider $\mathcal{S}^{\mathcal{S}}(\Omega)$ as the input shape and $\partial\partial\Omega$ as its initial boundary. Since $\partial\partial\partial\Omega$ corresponds to $\mathcal{S}^{\mathcal{C}}_{\mathcal{E}}(\Omega)$ in the same way that $\partial\partial\Omega$ corresponds to $\mathcal{S}^{\mathcal{S}}_{\mathcal{E}}(\Omega)$, identifying $\mathcal{S}^{\mathcal{C}}_{\mathcal{E}}(\Omega)$ directly allows such a recursive approach.

When we compute the skeleton of this shape using our skeletonization method, we will find $\mathcal{S}(\mathcal{S}(\Omega)) = \mathcal{S}^{\mathcal{C}}(\Omega)$ as the skeleton. Similarly we also find $\partial\partial\partial\Omega \subset \partial\partial\Omega$ and see that the interpretations for $\partial\partial\Omega$ in the previous subsection are also applicable to $\partial\partial\partial\Omega$. Effectively, if our method works for the surface skeleton, it must also work for the curve skeleton; the differences between the two problems reduce to being merely a matter of interpretation.

6 Results (3D shapes)

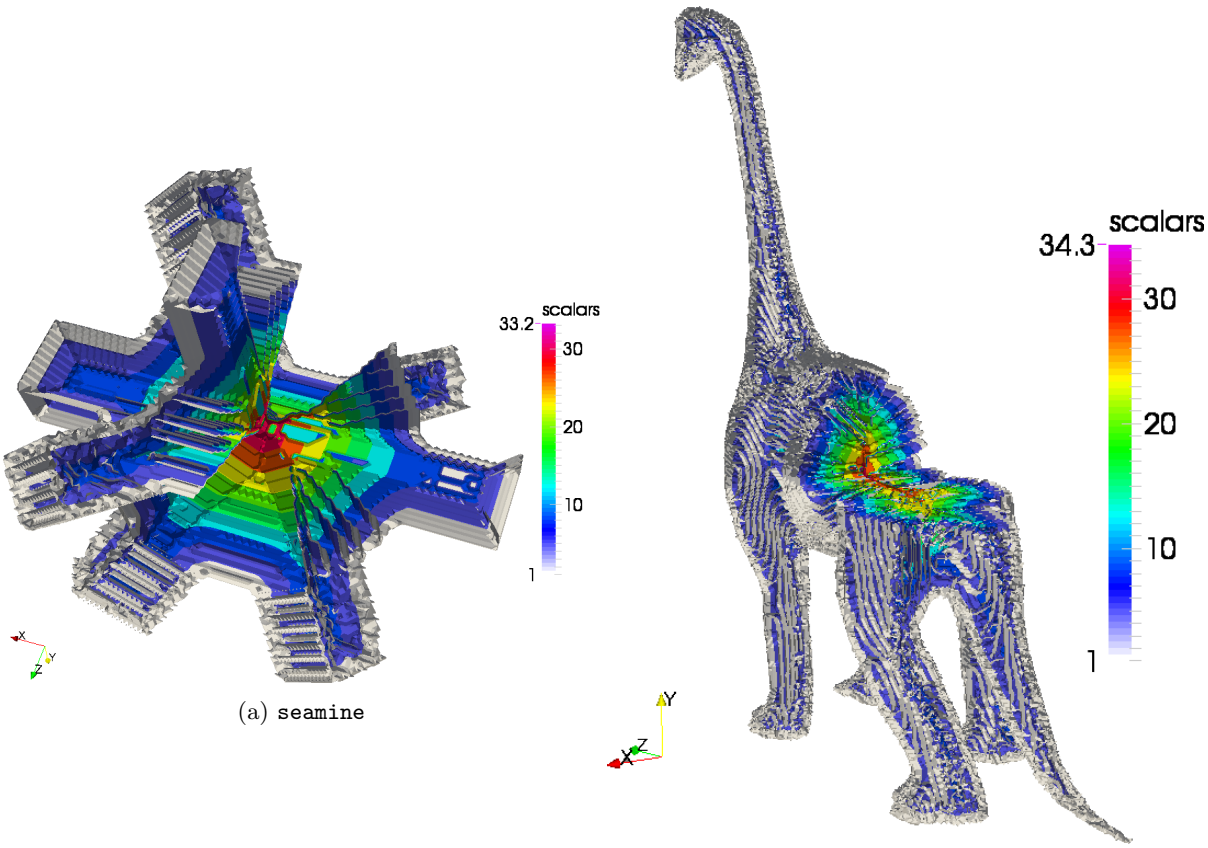
6.1 Example figures



(a) `brain` (b) `ico`

(c) `cow2` (d) `anvilnoisy`

Figure 6.1: Example figures depicting the skeleton. Several clipping planes are used to reveal the inner skeleton. Colors (restricted to 10 distinct color levels) depict shape DT values



(a) `seamine`

(b) `dinoflat`

Figure 6.2: Example figures depicting the skeleton. Several clipping planes are used to reveal the inner skeleton. Colors (restricted to 10 distinct color levels) depict shape DT values

6.2 Computational complexity

Our method is a variation of the AFMM Star and contains several instances thereof. We denote such an instance by $afmm(p, \partial p)$ where p denotes the input and ∂p denotes the initial wavefront on p, and the complexity can then be denoted as $O(p \log \partial p)$. For each of these instances we also denote whether it applies to the 2D or 3D case. The instances are thus denoted as follows:

- Shape DT / FT (2D, 3D): $afmm(\Omega, \partial\Omega)$
- Boundary DT / FT (3D) : $afmm(\partial\Omega, \partial\partial\Omega)$
- Surface skeleton DT / FT (3D) : $afmm\big(\mathcal{S}^{\mathcal{S}}(\Omega), \partial\mathcal{S}^{\mathcal{S}}(\Omega)\big)$
- Curve skeleton DT / FT (2D, 3D): $afmm\big(\mathcal{S}^{\mathcal{C}}(\Omega), \partial\mathcal{S}^{\mathcal{C}}(\Omega)\big)$

and the sum of the complexities of these instances then yields the total complexity of our method:

$$O\Big(\Omega \log \partial\Omega + \partial\Omega \log \partial\partial\Omega + \partial\mathcal{S}^{\mathcal{S}}(\Omega) \log \partial\mathcal{S}^{\mathcal{S}}(\Omega) + \mathcal{S}^{\mathcal{C}}(\Omega) \log \partial\mathcal{S}^{\mathcal{C}}(\Omega)\Big) \tag{6.1}$$

Note that $O\big(\mathcal{S}_{\mathcal{E}}(\Omega)\big) \approx O(\partial\partial\Omega)$ and $O\big(\mathcal{S}^{\mathcal{C}}(\Omega)\big) \approx (\partial\partial\partial\Omega)$, which effectively says that the amount of "contact surface" between the skeleton and the boundary of the shape determines the cost of the skeleton traversal step. From the definition of $\partial\partial\Omega$ it follows that $\partial\partial\Omega$ is the minimal set of points that must be considered to compute the importance metric. Put differently, $\partial\partial\Omega$ can be interpreted as a optimal solution for a Dijkstra-like boundary traversal method, in particular the approach taken in Reniers et al.[33].

Generally, the initial wavefront of a set of points represents is between 5% and 15% of the entire shape, although this can strongly vary depending on the input shape and its resolution. For the skeleton of a shape, this percentage can go up to 50% (for the shapes we have tested) in cases where the boundary is littered with small pertubations. This causes a relatively dense skeleton consisting of many points, the impact of which is amplified even further due to our use of a 4-connected (2D) and 6-connected (3D) skeleton. Example figures that depict the cases where this happens are given as fig. 3.11b on page 37 for the 2D case and as fig. 6.1b on page 71 for the 3D case.

6.3 Memory requirements

For each instance of the AFMM Star method, the amount of memory depends on the number of points in the input shape, the number of points on the initial wavefront and the number of points in the resulting skeleton. For the first instance $afmm(\Omega, \partial\Omega)$, we store:

- Flagfield: 0.5 bytes / point

- Shape DT: 4 bytes / point
- Shape FT: 4 bytes / point
- Sorted map pointers: 4 bytes / point
- Followband: 1 byte / point
- Boundary model: 22 bytes / boundary point

where we re-use the flagfield, followband and boundary model for the subsequent instances. For each subsequent instance we then only need to store an additional 8 bytes per point for the DT / FT values. Additionally, memory is needed for the sorted map used by the FMM and a field of pointers to reduce the number of binary searches when updating previously computed values. Given the use of `std::multimap` in a 32-bit application, this uses 28 bytes for each item in the sorted map. Generally the maximum number of points in the sorted map is of the same order as the initial wavefront, which thus gives an estimate for the required amount of memory.

7 Discussion

Some of the methods we use this thesis are limited in their accuracy by several factors. Even though some of these factors have predictable or well-known consequences, subtle variations can lead to significantly different results. Even if the results appear similar, small differences can seem insignificant or go unnoticed and introduce or amplify issues in a subsequent step. This is especially true when generalizing a method to a higher dimensional space.

Predicting or correcting issues is often difficult, if not impossible. In this section, we detail on several accuracy issues as well as the manner in which we deal with these.

7.1 Discretization

Using a discrete grid can be problematic in the sense that there is not always a single center point. fig. 3.8 depicts how the skeleton of a 2D shape can be determined in several ways. Clearly the discrete grid cannot possibly always yield a skeleton that is both centered and one point thick.

7.2 Accuracy of distance transform

The DT computed by the FMM (section 2.2.1) is numerically inexact, which means that the order in which the FMM visits points is not strictly in order of increasing distance to the boundary. Since the errors appear to diffuse away smoothly from boundary concavities [32], these errors only become significant on or near the skeleton. We crudely assume that this implies that if $T(p) > T(q)$ for $q \in N_8(p)$, then the exact distance $D(p) > D(q)$ unless p and / or q are skeleton points.

In case $T(p) > T(q)$, we can simply rely on relatively accurate values. In other cases, erroneous distances can lead to erroneously detected skeleton points. We already assumed that the skeleton can be potentially two points thick due using a discrete grid. This means that if we deal with the discretization artifacts, we automatically cover most of the areas affected by inaccuracies in the DT.

7.3 Sorting method and initialization

Consider a heap container that sorts tuples $(point, value)$ in order of ascending $value$. If we add $(a, 1)$ and $(b, 1)$ to the container, then both a and b qualify as a point with the smallest value. If the heap container uses a stable sorting method, retrieving a

tuple with the minimum value from the container will return the tuples in the order in which they were initially added.

The implication of this is that the choice of initialization method can affect the result of a method that uses a heap container in this manner. A clear (but absurd) example is a initialization method that randomizes the order of equally valued items in the container. A less obvious case is the order in which we inspect the adjacent points q of some gridpoint p. An example is the AFMM Star (section 2.8.1), which assigns to p one optimal value derived from the values stored at points $q \in N_8(p)$. If two distinct values are found that are equally optimal, the value that is encountered first is chosen.

Although this does not appear to hinder the AFMM Star (which operates in 2D space), we found that the impact of these subtle differences can easily grow to significant proportions when operating in 3D space. We considered several approaches to mitigate these problems:

7.4 Topological properties of neighbourhood sizes

Consider the 2D FMM which uses 4-adjacency when adding points to or removing points from the temporary narrowband. If a `UNKNOWN` point p is 8-adjacent to a point q in the narrowband, then we know that p will be added to the narrowband at relative timestep t and q will be added no sooner than $t + 2$ (the manhattan distance between p and q). In this regard, if we are inspecting all 8-adjacent neighbours of a point, it would make sense to inspect the 4-adjacent neighbours first and only then inspect the remaining neighbours; they are "closer".

These properties relate closely to the general idea behind the followband (section 3.3) as well as the manner in which the FMM (section 2.2.1) is formalized. Additionally, these properties explain in very basic terms why a small neighbourhood size can be sufficient for iterative and / or ordered propagation approaches in general.

7.5 Feature transform

Similar to Mullikin's ϵ-VDT [29], we can store all boundary points at the same minimum distance instead of only one. For the 2D case, storing one boundary points is entirely sufficient and we do not see a need for the ϵ-VDT here.

For the 3D case, we experiment with several values for ϵ between 0 and $\sqrt{3}$. We find that although $\epsilon = 0$ can result in small discrepancies, the overall results are still acceptable. Higher values of ϵ produce results that are clearly less erratic, but the differences do not appear significant enough to warrant the additional cost of the ϵ-VDT.

7.6 Atomic vs sequential updates

Consider the case when we are updating several points (such as multiple points being removed from the followband) at a time. When updating points in one atomic step (similar to how the FMM re-evaluates already computed distance values), we might lose information that is hidden in some way in which these points interact with or depend upon one another. Discovering this information or finding out how to extract this information is often tedious and difficult. For example, we can say the FMM deals with a comparable issue by correcting previously computed values of adjacent points, which is clearly not a trivial discovery.

Sorting adjacent points When processing a point, we can sort adjacent points and process them in order of ascending T value instead of inspecting them in a regular grid pattern (i. e. for 2D, left to right, top to bottom). We have attempted to use this approach at various places and stages during the implementation of the work in this thesis. Even though this approach is ultimately not used in our skeletonization method due to finding different solutions, we did notice many instances where a sorting approach improved the results considerably.

We did however apply this approach in the inheritcount measure (section 3.8) in the form of eq. (3.19). Interestingly, the sorting approach appears to be more effective when we use a simple FT, especially around areas where discretization artifacts are present.

As such, since a sorting approach is simple to implement at a moderate computational cost, it can serve as a fast and practical way to find or emphasize patterns.

8 Conclusions

In this chapter we summarize the contributions of the work presented in the previous chapters, and we make several recommendations for future work.

8.1 Contributions

In chapter 3 we presented a skeletonization method which computes the skeleton for 2D shapes and computes the boundary-collapse measure in an incremental fashion. We have shown that this method produces results similar to the AFMM Star method. In chapter 5 we show that this skeletonization method readily generalizes to 3D, and we present a proposed method to compute the multiscale importance metric for 3D shapes by decomposing the problem set into multiple similar problems, each of which can be solved with effectively one single method.

8.2 Future work

In this section we discuss several opportunities and considerations for future work relating to what is presented in this thesis.

8.2.1 Implementing our proposed method

An obvious candidate for future work is implementing the approach as suggested in chapter 5 and compare both the results and performance with the Reniers et al.[33] method. It would also be interesting to consider how our suggested approach can be incorporated into other skeletonization methods. Although our method was created with a focus on a volumetric representation, the basic approach could very well apply to skeletonization methods using a point-cloud or mesh representation.

8.2.2 Optimizations

The most costly component of our method is the computation of the FMM DT, requiring one or more binary searches in the sorted map for every processed point. Clearly operations on the sorted map are costly in terms of algorithmic complexity. The memory access pattern of such an operation is also costly, especially for large 3D shapes for which the sorted map does not fit or barely fits in the CPU cache. Since main memory access latencies are generally an order of magnitude larger than accessing the CPU cache, it is critical for performance to avoid delays caused by having to "wait" for data to become available.

There are several methods in existance which improve upon the computational cost of the FMM while retaining most of its useful properties. The following methods focus on minimizing the use and / or cost of a sorted map or not requiring such a data structure at all, trading the logarithmic component for computations with a constant or linear cost.

FMM alternatives A domain decomposition strategy is used by Herrmann[14] where the shape is subdivided and subdomains are processed by a different CPU. Inter-domain communication is required for CPU's that process adjacent subdomains, and a rollback mechanism is used to correct errors that can occur during the parallel marching. A variation is also introduced where the narrowband is subdivided instead of the entire shape. This method can perform optimally only if a suitable decomposition is found, which is not necessarily the case.

Other methods which focus on avoiding binary searches are the Group Marching Method (GMM) [20] and the Fast Sweeping Method (FSM) [19]. In [18] the Fast Iterative Method (FIM) is introduced as more efficient solution, suitable for parallelized implementations on the CPU and GPU. While the GMM, FSM and FIM methods are considerably faster than the FMM in most cases, their performance can drop significantly when certain conditions occur in the input.

Intuitive approach A large speedup may be attained by using a parallel DT implementation such as the method of Meijster et al.[27] to precompute all the DT values. These DT values can then be put in a container in order of ascending DT value. Subsequently, since we will be visiting these items in order, memory access patterns will be highly predictable and should therefor allow prefetching mechanisms to effectively reduce or eliminate these delays.

Sorting the container is however a costly operation that may easily negates any improvements elsewhere. If a suitable parallel sorting method with a high speedup factor is feasible, this approach may be able to compete with other approaches in terms of running time. The reason for this is that the FMM alternatives are typically harder to parallelize. Additionally, generic parallel sorting methods have received widespread attention and effort in order to improve parallelization speedups even for a high number of processors. As a result, a solution consisting of several very efficient, well-parallelizable methods can outperform other methods simply by scaling better with the number of processors.

The main advantage of this approach is that it is a drop-in replacement for the FMM method, requiring no other changes to our method. Another advantage is that this approach depends only on the resolution of the input and, unlike the FMM alternatives, not on particular properties of the shapes present in the input. Although this dependency argueably concerns mostly rare or uncommon cases and does not affect the average running time over all sorts of shapes, a slower method with a more predictable running time may be preferrable in some applications.

8.3 Acknowledgements

I would first like to express my deepest gratitude to my supervisor Alex Telea for his continued guidance, support, enthusiasm and endless patience. Our initial discussions inspired, motivated and enlightenened me more than I dare describe in a few words and this thesis would not have been written otherwise. Thank you for being an example, for sharing your thoughts and knowledge, for helping me reach my goals and for all the troubles I put you through.

I would also like to thank my second supervisor Michael Wilkinson for his suggestions, words of encouragement and for going out of his way to help me out of a tight spot. Thank you for making me feel better about myself.

I would like to thank my family, friends and co-workers for being supportive and understanding and putting up with me and my thesis writing life. Thank you for dealing with an absent-minded me and giving me a place to relax and recharge and having a life besides my thesis.

Finally I want to express my gratitude to my other half, Sabine, for understanding and supporting me all the way, dealing with all the loose ends, always being there for me and for giving me what I needed most to get me to where I am today. Thank you for you, thank you for giving me strength and pushing me to better myself and thank you for being the reason to persevere.

Bibliography

[1] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 44:1–44:10, New York, NY, USA, 2008. ACM. ISBN 978-1-4503-0112-1. doi: 10.1145/1399504.1360643. URL `http://doi.acm.org/10.1145/1399504.1360643`.

[2] Harry Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn, editor, *Proc. Models for the Perception of Speech and Visual Form*, pages 362–380, Cambridge, MA, November 1967. MIT Press.

[3] Harry Blum. Biological shape and visual science (part i). *Journal of Theoretical Biology*, 38(2):205 – 287, 1973. ISSN 0022-5193. doi: http://dx.doi.org/10.1016/0022-5193(73)90175-6. URL `http://www.sciencedirect.com/science/article/pii/0022519373901756`.

[4] Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. Point cloud skeletons via laplacian based contraction. In *Proceedings of the 2010 Shape Modeling International Conference*, SMI '10, pages 187–197, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4072-6. doi: 10.1109/SMI.2010.25. URL `http://dx.doi.org/10.1109/SMI.2010.25`.

[5] Thanh-Tung Cao, Ke Tang, Anis Mohamed, and Tiow-Seng Tan. Parallel banding algorithm to compute exact distance transform with the gpu. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 83–90, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-939-8. doi: 10.1145/1730804.1730818. URL `http://doi.acm.org/10.1145/1730804.1730818`.

[6] Ulrich Clarenz, Martin Rumpf, and Alexandru Telea. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):516–524, September 2004. ISSN 1077-2626. doi: 10.1109/TVCG.2004.34. URL `http://dx.doi.org/10.1109/TVCG.2004.34`.

[7] Nicu D. Cornea and Deborah Silver. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13: 530–548, 2007.

[8] Luciano da Fontoura Da Costa and Roberto Marcondes Cesar, Jr. *Shape Analysis and Classification: Theory and Practice*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2000. ISBN 0849334934.

[9] Tamal K. Dey and Jian Sun. Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 143–152, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. ISBN 3-905673-36-3. URL `http://dl.acm.org/citation.cfm?id=1281957.1281975`.

[10] Tamal K. Dey and Wulue Zhao. Approximate medial axis as a voronoi subcomplex. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, SMA '02, pages 356–366, New York, NY, USA, 2002. ACM. ISBN 1-58113-506-8. doi: 10.1145/566282.566333. URL `http://doi.acm.org/10.1145/566282.566333`.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[12] Mark Foskey, Ming C. Lin, and Dinesh Manocha. Efficient computation of a simplified medial axis. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, pages 96–107, New York, NY, USA, 2003. ACM. ISBN 1-58113-706-0. doi: 10.1145/781606.781623. URL `http://doi.acm.org/10.1145/781606.781623`.

[13] Yaorong Ge and J. Michael Fitzpatrick. On the generation of skeletons from discrete euclidean distance maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18 (11):1055–1066, November 1996. ISSN 0162-8828. doi: 10.1109/34.544075. URL `http://dx.doi.org/10.1109/34.544075`.

[14] M. Herrmann. A domain decomposition parallelization of the fast marching method. *CTR Annual Research Briefs*, pages 213–225, 2003.

[15] Wim H. Hesselink and Jos B. T. M. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(12):2204–2217, December 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.21. URL `http://dx.doi.org/10.1109/TPAMI.2008.21`.

[16] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Trans. Graph.*, 32(4):65:1–65:8, July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461913. URL `http://doi.acm.org/10.1145/2461912.2461913`.

[17] Andrei Jalba, Jacek Kustra, and Alexandru Telea. Surface and curve skeletonization of large 3D models on the GPU. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 35(6):1495–1508, 2013.

[18] Won-Ki Jeong and Ross T. Whitaker. A fast iterative method for eikonal equations. *SIAM J. Sci. Comput.*, 30(5):2512–2534, July 2008. ISSN 1064-8275. doi: 10.1137/060670298. URL `http://dx.doi.org/10.1137/060670298`.

[19] S. Kim. An $O(n)$ level set method for eikonal equations. *SIAM Journal on Scientific Computing*, 22(6):2178–2193, 2001. doi: 10.1137/S1064827500367130. URL http://dx.doi.org/10.1137/S1064827500367130.

[20] Seongjai Kim. An o(n) level set method, 2000.

[21] Ron Kimmel, Doron Shaked, Nahum Kiryati, and Alfred M. Bruckstein. Skeletonization via distance maps and level sets. *Comput. Vis. Image Underst.*, 62(3): 382–391, November 1995. ISSN 1077-3142. doi: 10.1006/cviu.1995.1062. URL http://dx.doi.org/10.1006/cviu.1995.1062.

[22] Gisela Klette. Simple points in 2d and 3d binary images. In Nicolai Petkov and MichelA. Westenberg, editors, *Computer Analysis of Images and Patterns*, volume 2756 of *Lecture Notes in Computer Science*, pages 57–64. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40730-0. doi: 10.1007/978-3-540-45179-2_8. URL http://dx.doi.org/10.1007/978-3-540-45179-2_8.

[23] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885, September 1992. ISSN 0162-8828. doi: 10.1109/34.161346. URL http://dx.doi.org/10.1109/34.161346.

[24] F. Leymarie and M.D. Levine. Simulating the grassfire transform using an active contour model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(1):56–75, Jan 1992. ISSN 0162-8828. doi: 10.1109/34.107013.

[25] Xuetao Li, Tong Wing Woon, Tiow Seng Tan, and Zhiyong Huang. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pages 35–42, New York, NY, USA, 2001. ACM. ISBN 1-58113-292-1. doi: 10.1145/364338.364343. URL http://doi.acm.org/10.1145/364338.364343.

[26] Jaehwan Ma, SangWon Bae, and Sunghee Choi. 3d medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer*, 28(1):7–19, 2012. ISSN 0178-2789. doi: 10.1007/s00371-011-0594-7. URL http://dx.doi.org/10.1007/s00371-011-0594-7.

[27] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink. A general algorithm for computing distance transforms in linear time, 2000.

[28] Balint Miklos, Joachim Giesen, and Mark Pauly. Discrete scale axis representations for 3d geometry. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 101:1–101:10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0210-4. doi: 10.1145/1833349.1778838. URL http://doi.acm.org/10.1145/1833349.1778838.

[29] James C. Mullikin. The vector distance transform in two and three dimensions. *CVGIP: Graph. Models Image Process.*, 54(6):526–535, November 1992. ISSN 1049-9652. doi: 10.1016/1049-9652(92)90072-6. URL http://dx.doi.org/10.1016/1049-9652(92)90072-6.

[30] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 63–69, 1992.

[31] Dennie Reniers and Alexandru Telea. Quantitative comparison of tolerance-based feature transforms. In *Proceedings of the First International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 107–114, 2006.

[32] Dennie Reniers and Alexandru Telea. Tolerance-based feature transforms. In *Advances in Computer Graphics and Computer Vision*, volume 4 of *Communications in Computer and Information Science*, pages 187–200. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-75272-1. doi: 10.1007/978-3-540-75274-5_12. URL `http://dx.doi.org/10.1007/978-3-540-75274-5_12`.

[33] Dennie Reniers, Jarke J. van Wijk, and Alexandru Telea. Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. *IEEE TVCG*, 14(2):355–368, 2008. URL `http://www.win.tue.nl/~alext/ALEX/PAPERS/TVCG07/paper.pdf`.

[34] K. Siddiqi, S. Bouix, A Tannenbaum, and S.W. Zucker. The hamilton-jacobi skeleton. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 828–834 vol.2, 1999. doi: 10.1109/ICCV.1999.790307.

[35] A. Sobiecki, H. Yasan, A. Jalba, and A. Telea. Qualitative comparison of contraction-based curve skeletonization methods. In *Mathematical Morphology and Its Applications to Image and Signal Processing*, 2013.

[36] A. Sobiecki, A. Jalba, and A. Telea. Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recognition Letters*, 2014. doi: 10.1016/j.patrec.2014.01.012.

[37] Avneesh Sud, Mark Foskey, and Dinesh Manocha. Homotopy-preserving medial axis simplification. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, SPM '05, pages 39–50, New York, NY, USA, 2005. ACM. ISBN 1-59593-015-9. doi: 10.1145/1060244.1060250. URL `http://doi.acm.org/10.1145/1060244.1060250`.

[38] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 71:1–71:9, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi: 10.1145/1576246.1531377. URL `http://doi.acm.org/10.1145/1576246.1531377`.

[39] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the Symposium on Data Visualisation*, pages 251–259. VisSym, 2002.

[40] Ming Wan, Frank Dachille, and Arie Kaufman. Distance-field based skeletons for virtual navigation. In *Proceedings of the Conference on Visualization '01*, VIS '01, pages 239–246, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7803-7200-X. URL `http://dl.acm.org/citation.cfm?id=601671.601708`.