TU WIEN Informatics

Feature-Size-aware Sampling of Loop Subdivision Surfaces

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Dr. Jörg Christian Reiher
Matrikelnummer 12213117

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Mag. rer. soc. oec. Stefan Ohrhallinger, PhD

Wien, 31. Juli 2026

Jörg Christian Reiher	Stefan Ohrhallinger

TU WIEN Informatics

Feature-Size-aware Sampling of Loop Subdivision Surfaces

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics and Visual Computing

by

Dr. Jörg Christian Reiher
Registration Number 12213117

to the Faculty of Informatics

at the TU Wien

Advisor: Mag. rer. soc. oec. Stefan Ohrhallinger, PhD

Vienna, July 31, 2026

Jörg Christian Reiher	Stefan Ohrhallinger

Erklärung zur Verfassung der Arbeit

Dr. Jörg Christian Reiher

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 31. Juli 2026

Jörg Christian Reiher

Danksagung

Mein Dank gilt meinem Betreuer Mag. rer. soc. oec. Stefan Ohrhallinger für die Unterstützung und die gute Zusammenarbeit. Ebenso danke ich allen Organisatorinnen und Organisatoren sowie den Teilnehmenden des Konversatoriums der Forschungsgruppe Computer Graphics für die Gelegenheit, meine Arbeit vorzustellen und Feedback zu erhalten. Darüber hinaus danke ich Martin Marinov und Leif Kobbelt dafür, dass sie mir ihre Implementierung [MK05] zur Verfügung gestellt haben.

Acknowledgements

I would like to thank my advisor Stefan Ohrhallinger for his support and the good collaboration. I am also grateful to all organizers and participants of the colloquium of the Computer Graphics research group for the opportunity to present my work and receive valuable feedback. Furthermore, I would like to thank Martin Marinov and Leif Kobbelt for providing their implementation [MK05].

Kurzfassung

Die mediale Achse (MA) einer geschlossenen Fläche und die von ihr induzierte lokale Feature-Größe (local feature size, lfs) sind wichtige Deskriptoren in der Geometrieverarbeitung. Bei Dreiecksnetzen verschwindet die lfs an Vertices und Kanten, wodurch auf der MA Äste und Blätter entstehen, die klassische Verfahren zur Approximation der MA von Dreiecksnetzen durch sorgfältig abgestimmte, netzspezifische Heuristiken entfernen oder verkürzen. Die vorliegende Arbeit umgeht derartige Heuristiken und die damit einhergehenden Probleme und Präzisionsverluste, indem die MA direkt auf einer glatten Fläche berechnet wird: der Loop-Subdivisionsgrenzfläche eines Dreiecksnetzes. Damit verallgemeinert diese Arbeit die Idee eines bestehenden, lfs-sensitiven 2D-Sampling-Verfahrens [OPM23] für planare Kurven auf Loop-Subdivision-Oberflächen in $\mathbb{R}^3$.

Die vorliegende Arbeit führt eine vierstufige Pipeline ein, die ein geschlossenes Dreieckskontrollnetz als Eingabe verwendet und ein trianguliertes lfs-sensitives Sampling von dessen Loop Subdivisionsfläche berechnet. Dies geschieht indem maximale, leere, tangentiale Kugeln an Tesselierungspunkten der Subdivisionsfläche berechnet werden. Dabei kommt ein Bisektionsverfahren für den Radius und das Newton-Verfahren zur Bestimmung dichtester Punkte zum Einsatz. Die Mittelpunkte dieser Kugeln liegen auf der MA und werden mithilfe der Topologie der Tesselierung zu Dreiecken trianguliert, die die MA beliebig gut approximieren. Mit der Abstandsfunktion zu dieser Approximation wird eine Methode zur Gewinnung eines ε-Samplings der Patches der Subdivisionsfläche hergeleitet und mithilfe einer 2D Constrained Delaunay Triangulierung werden diese Samples der Patches zu einem triangulierten ε-Sampling der gesamten Subdivisionsfläche zusammengefügt.

Da für die MA einer Loop-Subdivisionsfläche keine geschlossene Lösung bekannt ist, wird die Methode empirisch validiert. Im Vergleich zu State-of-the-Art-Methoden erfasst der vorliegende Ansatz feinere Features der MA bereits bei niedrigen Tesselierungsstufen und erreicht eine Präzision, die die anderen Verfahren erst bei wesentlich höheren Auflösungen und nur durch manuelle Feinjustierung ihrer Parameter erzielen. Die Genauigkeit des vorliegenden Ansatzes wird hingegen im Wesentlichen nur durch einen einzigen Tesselierungsparameter gesteuert. Praktische Anwendungen werden in Objekterkennungspipelines, Rendering von Loop-Subdivisionsflächen auf mobilen Geräten und in der Evaluierung von Algorithmen zur Oberflächenrekonstruktion vorgestellt.

Abstract

The medial axis (MA) of a closed surface and the local feature size (lfs) it induces are important descriptors in geometry processing. For triangle meshes, the lfs vanishes at vertices and edges, causing branches and leaves on the MA, which classical methods for MA approximation of meshes remove by carefully tuned, mesh-specific pruning heuristics. This thesis avoids these instabilities by computing the MA directly on a smooth Loop subdivision surface of a triangle mesh. It therefore generalizes the idea behind an existing 2D lfs-aware sampling method [OPM23] for planar curves to Loop subdivision surfaces in $\mathbb{R}^3$.

This thesis introduces a four-step pipeline that takes a closed triangular control mesh as input and computes a triangulation of a lfs-aware sampling of its subdivision surface. First, the MA is sampled by growing at each foot point of a uniform tessellation of the Loop subdivision surface, a maximal empty tangent sphere. This reduces the problem to a one-dimensional binary search in the sphere radius, in which each candidate is evaluated by a Newton-scheme-based closest-point query on the subdivision patches. The obtained sphere centers are points on the MA. Second, these points inherit the combinatorics of the foot-point tessellation, which yields a manifold, possibly self-intersecting mesh and thus a triangle soup used for fast distance queries for the lfs. Third, for a parameter ε, that controls the average sampling density, an ε-sampling of the Loop subdivision surface is created with this MA by recursively subdividing each patch in its parameter domain until an enclosing sphere proxy, justified by the 1-Lipschitz continuity of the lfs, certifies the ε-condition over the whole sub-patch. This local criterion is proven to be sufficient to guarantee an ε-sampling of the entire surface. Fourth, the samples are triangulated per patch via a constrained Delaunay triangulation in parameter space, with shared patch boundaries enforced as constraints to produce a consistent manifold output mesh.

Since no closed-form solution for the MA of a Loop subdivision surface is available, the method is validated empirically. Compared against the state-of-the-art methods, the proposed approach captures fine medial features already at low tessellation levels and reaches a precision that competing methods attain only at substantially higher resolutions and after manual pruning, while its accuracy is controlled only by a single, monotone resolution parameter rather than per-mesh tuning. Practical applications are presented in shape recognition pipelines, rendering of subdivision assets, and evaluation of surface reconstruction algorithms.

Contents

CHAPTER 1

Introduction

In this chapter, the motivation for the present work is given, and its goals are outlined. Then fundamental concepts and definitions are introduced, and finally it is outlined how we achieve the goals derived in the motivation.

1.1 Motivation

Triangle meshes are one of the most common forms to describe and store geometries for computing and graphics applications, and most real-time graphics pipelines rely on this format. When a triangle budget for a geometry is given (e.g., in real-time rendering on mobile devices or in scientific simulations), it is necessary to assess which parts of a geometry can be modeled efficiently with few larger triangles and which parts require a higher amount of smaller triangles to model finer features. One option to measure how delicate or fine geometric features at a point are is to determine the distance of a point on the geometry to the medial axis (MA), which is a skeleton-like structure for the geometry. This distance is called the local feature size (lfs) of the point. The lfs has the inherent problem that it vanishes at edges and vertices of discrete meshes, but for smooth surfaces this is not the case. Loop subdivision surfaces, consisting of box spline patches, are such smooth surfaces, and in applications such as computer animation, artists can use Loop subdivision surfaces to model geometries. Therefore, it is desirable to find an approach to approximate the MA of a Loop subdivision surface since it allows a lfs-aware sampling of this surface, which yields an efficient triangle-mesh representation of the modeled geometry. Fig. 1.1 shows a dog modeled as a Loop subdivision surface that has been triangulated with a fixed budget of 3500 triangles through regular tesselation and with lfs-aware sampling. The latter uses more triangles on fine features such as creases behind ears while only using a few large triangles to model the less detailed core of the body.

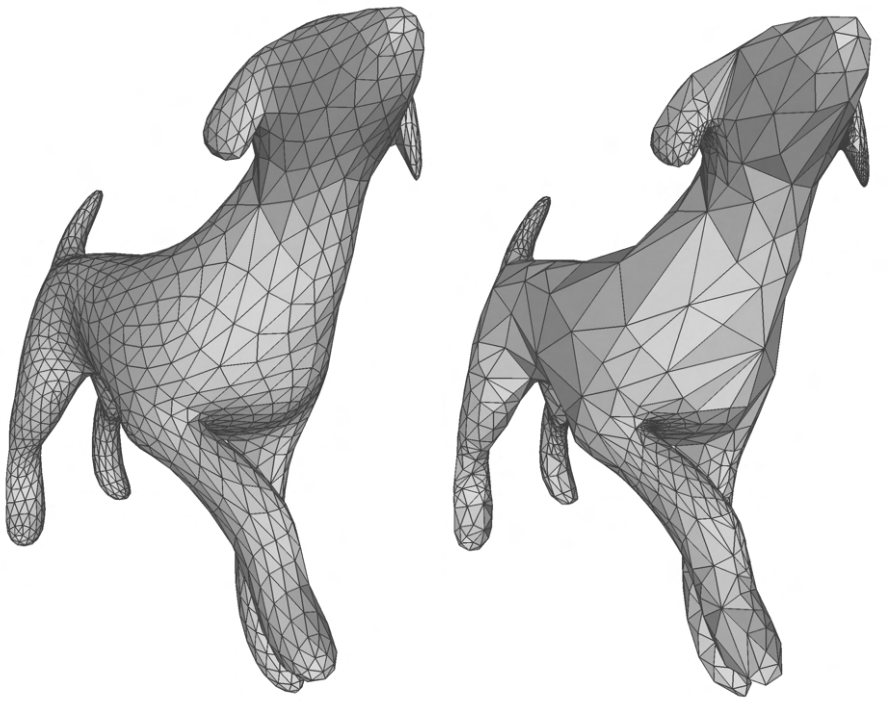

Figure 1.1: Left: a regular tesselation of a Loop subdivision surface with 3500 triangles, right: a lfs-aware triangulation of the same surface with 3500 triangles.

1.2 Fundamentals and definitions

In this subsection, formal definitions for all terms introduced in Ch. 1.1 and used throughout this work are given.

Loop subdivision surfaces

The Loop subdivision scheme [Loo87] uses an update rule to refine a coarse triangle mesh, referred to as the control mesh. A mesh vertex with six neighbors (valence is six) is called regular and extraordinary otherwise. The Loop subdivision scheme adds a new vertex on every edge as the weighted average of the surrounding vertices, as depicted in Fig. 1.2, splitting every triangle into four. It also repositions existing vertices as weighted averages of their surrounding vertices. This ensures that on repeated application of this rule the resulting sequence of meshes converges to a smooth limit surface, called the Loop subdivision surface. The Loop subdivision surface can be interpreted as a smooth collection of patches (one for every control triangle) that is C1-continuous at extraordinary vertices and C2 everywhere else. Fig. 1.3 shows a visualization of a Loop subdivision surface with colored patches. For a regular control mesh, these patches are a quartic box spline. Around extraordinary control vertices, the limit surface is an infinite collection of quartic box spline; see Ch. 2.1 for more detailed information.

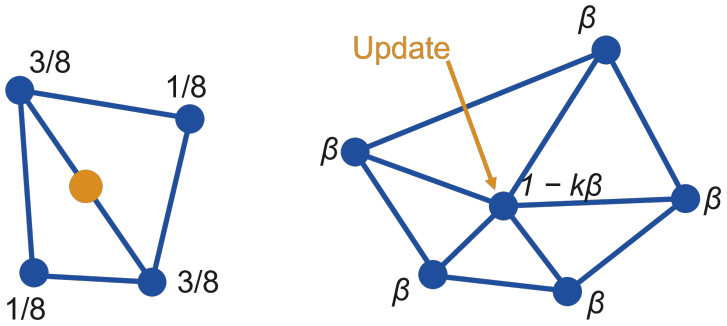

Figure 1.2: The Loop subdivision update rules for new vertices (left) and existing vertices (right) where k is the valence of the existing vertex to be updated, and $\beta := \frac{3}{8k}$ for $k > 3$ and $\beta := \frac{3}{16}$ for $k = 3$

The medial axis

The MA of a bounded domain in $\mathbb{R}^d$ is defined as the locus of centres of maximal inscribed spheres, i.e. spheres that are contained in the domain and that touch its boundary in at least two distinct points (see Fig. 1.4 for an illustration). Points on the MA are called medial points. For non-trivial boundaries, in particular for surfaces with sharp features, the MA is generally infeasible to obtain in closed form, since it is defined by the bisectors of curved surfaces (or surface patches) and requires the solution of high-order non-linear equations. Another problem is that the MAT is very unstable: arbitrarily small perturbations of the boundary can introduce new medial branches that reach all the way to the surface, where the lfs tends to zero. As a consequence, practical methods almost always have to compute an approximation of the MAT and pair it with an application-aware filtering or pruning step that suppresses unstable parts of the MA. See Ch. 2.2 for further details.

ε-sampling

Since the lfs measures the distance of a point on a boundary to the MA, it takes into account the curvature of the boundary and local diameter of the enclosed volume in the neighborhood of the point. Therefore, it can be used as a spatially varying measure of how densely a surface must be sampled such that the resulting samples are dense where the geometry is delicate and sparse where the surface is flat or far from other sheets (lfs-aware sampling). This makes the lfs a bridge between MA theory and practical surface discretizations since it turns geometric complexity into a target sampling density. An ε-sampling samples a boundary such that for every point p on the boundary, at least one sample point is closer than $\varepsilon \cdot \mathrm{lfs}(p)$ (w.r.t. the Euclidean metric of the space the boundary is embedded in). See Fig. 1.5 for an example of an ε-sampling of a 2D curve.

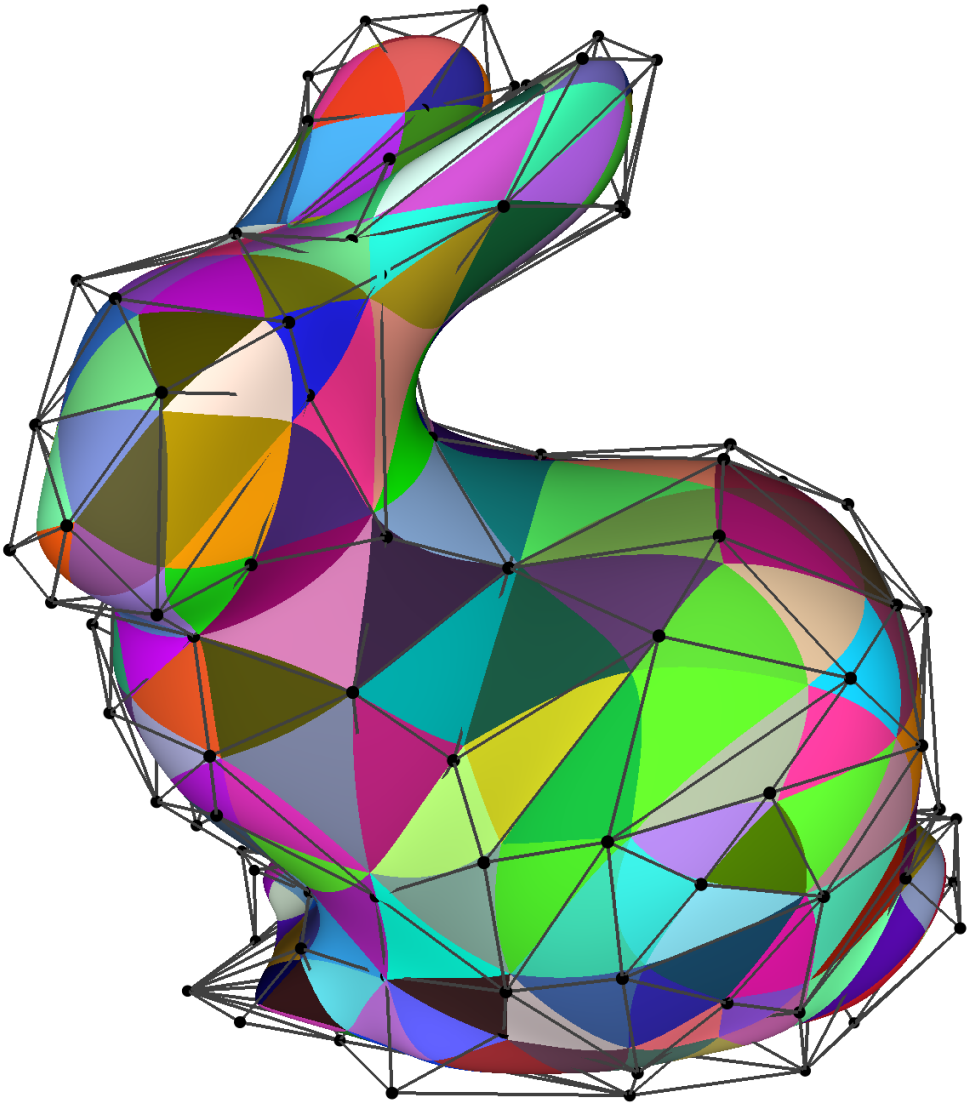

Figure 1.3: The Loop subdivision surface of a triangle Mesh. Every triangle of the control mesh gives rise to a patch. Patches are colored randomly for visual separation.

1.3 ε-sampling of Loop subdivision surfaces

In this work, we present an approach to approximate the MA of Loop subdivision surfaces up to arbitrary precision. This idea generalizes an earlier 2D approach [OPM23] to 3D surfaces. This opens up two practically relevant use cases: First, as previously mentioned in Ch. 1.1, it enables MA and lfs computation directly on loop subdivision surfaces created through standard modeling workflows, such as character meshes used in animation. Second, when a subdivision surface is fitted to a point cloud, our method can approximate the MA and lfs of the point cloud's underlying shape without the

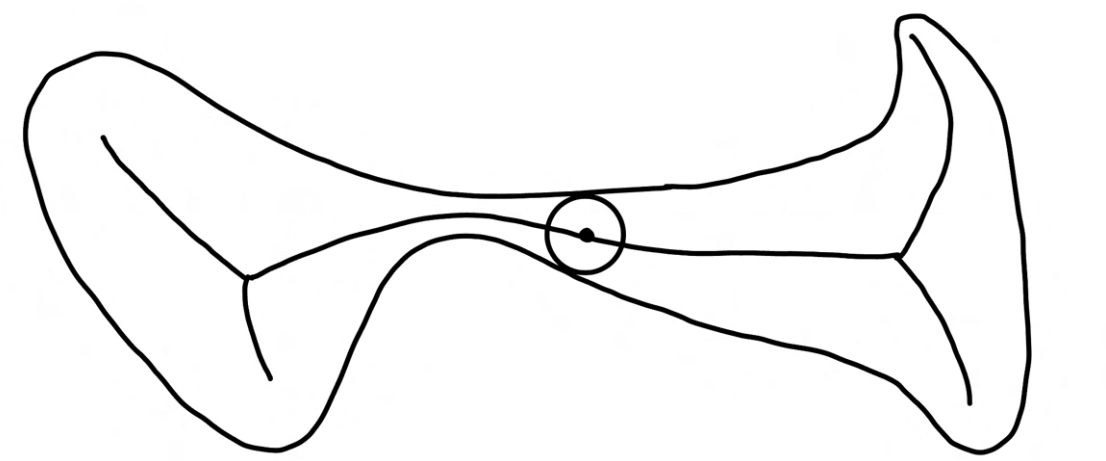

Figure 1.4: A curve in 2D with its inner medial axis and a medial ball around a medial point.

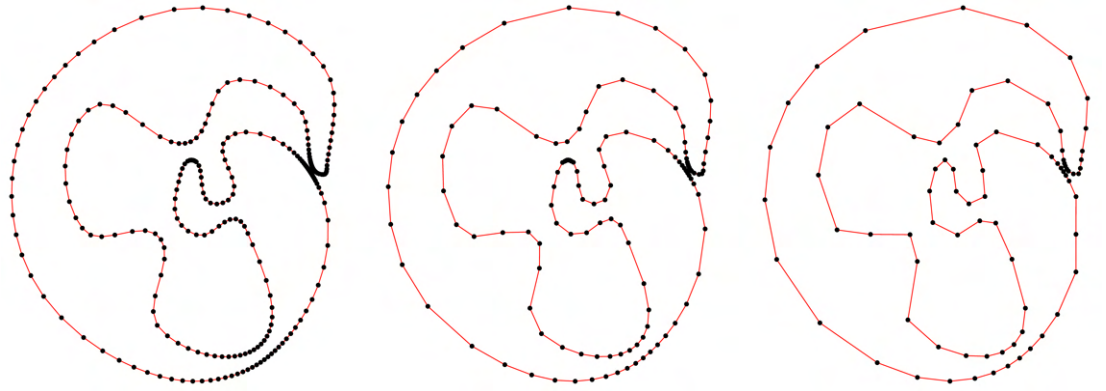

Figure 1.5: Depiction of ε-samplings of a 2D curve for $\varepsilon = 0.33$, 0.66, and 1 from the publication [OPM23] that inspired the present work.

instability and parameter-sensitivity that arise when computing these quantities directly on a triangulation of the point cloud.

We focus on the Loop subdivision scheme for this purpose since its subdivision surface achieves C^2 regularity almost everywhere (and C^1 at irregular vertices) - a substantial improvement over the mere C^0 continuity inherent to plain triangle meshes. An attractive practical consequence of this choice is that since Loop subdivision operates on triangle meshes, any closed manifold triangulation can serve directly as the coarse control input as long as it does not produce self-intersections or singularities of the subdivision surface.

Our method derives the MA of a Loop subdivision surface through sphere fitting. For each foot point sampled on the subdivision surface, we grow a maximal sphere that

remains entirely inside the boundary (a medial sphere), tangent to the surface in at least two locations. This determination of medial spheres relies on closest point determination through a Newton scheme. Collecting these spheres yields both the MA samples and, through their radii, the full MA transform. A piecewise linear approximation of the MA then emerges naturally by connecting samples according to the adjacency structure of their surface foot points. Because this structure encodes proximity on the subdivision surface, the resulting representation admits straightforward distance-based queries to evaluate the lfs at any point. Working directly with the smooth subdivision surface carries a fundamental advantage: the resulting MA is inherently well-behaved. It does not reach into degenerate boundary regions where the lfs vanishes, and its computation requires no pruning step, a notorious source of parameter-sensitivity in methods applied to polygonal meshes. The precision of our approach is governed solely by the tolerances of the underlying numerical routines and the geometry of the control mesh, making it a reliable reference that downstream applications can treat as a ground truth. The lfs derived from this MA are finally used to create a lfs-aware sampling, an ε-sampling, of the subdivision surface patches, which by definition is also an ε-sampling of the whole subdivision surface.

CHAPTER 2

Related Work

To the best of our knowledge, lfs-aware sampling of Loop subdivision surfaces or computation of their MA has not yet been addressed in the scientific literature. Consequently, relevant prior work falls primarily into three areas: feature-size-aware sampling of surfaces, fundamentals on subdivision surface evaluation and general methods for MA computation. Since our approach computes the MA of a given Loop subdivision surface only using established tools for subdivision surface evaluation, we mainly focus on related work that develops comparable MA computation methods for other geometric entities.

2.1 Loop subdivision surfaces

The reconstruction and representation of smooth free-form surfaces from discrete geometric data is a central concern in geometric modeling and computer graphics. The most ubiquitous representation of 3D shapes in practice remains the polygonal mesh, but many downstream tasks such as level-of-detail rendering, shape editing, finite element analysis, or precision manufacturing require a representation that is sufficiently smooth. Classical tensor-product splines, in particular non-uniform rational B-splines (NURBS)[PT97] have for a long time provided such a smooth representation. However, their rectangular parameter domain forces complex shapes to be decomposed into several patches that must afterwards be stitched together with explicit continuity conditions, which can be difficult for arbitrary topologies and often leads to intricate trim-curve constructions or auxiliary fairing terms[KL96, EH96]. Subdivision surfaces[Zor00, PR08] are a flexible alternative that bypasses the topological constraints of tensor-product representations. A subdivision scheme starts from a coarse polygonal control mesh and produces a sequence of progressively refined meshes by repeatedly inserting new vertices and updating existing positions through prescribed averaging rules. Under suitable conditions, the obtained sequence of meshes converges to a well-defined subdivision surface that is C^2-smooth almost everywhere [Rei95]. The appeal of this approach lies in the fact that the same

procedural rule applies uniformly to meshes of arbitrary genus and connectivity and that the subdivision surface (which exists for common schemes) is determined by the topology of a coarse control mesh, which yields a compact representation and allows interactive editing[DKT98]. Among the various subdivision schemes proposed for triangle meshes, the construction introduced by Charles Loop [Loo87] has become one of the most widely used. It generalizes the recurrence relations of box splines [dBHR93], which describe quartic piecewise polynomials over a regular triangular grid, to triangular meshes of arbitrary topological type. Each refinement step performs a one-to-four split of every triangle, after which the new edge-midpoint vertices and the relocated original vertices are computed as weighted averages of their respective neighborhoods. The weights are chosen such that, in regions with regular vertices, the subdivision surface coincides with the underlying quartic box spline. However, extraordinary vertices occur in most non-trivial meshes [Zor00]. At regular vertices, the subdivision surface is twice continuously differentiable, whereas at irregular vertices it is only once continuously differentiable[Rei95, Sch96]. The detailed eigenstructure of the local subdivision matrix, together with the characteristic map analysis associated with it, has been investigated in depth in the literature on stationary subdivision [Rei95, PR08] and provides the rigorous foundation for these smoothness statements. A fundamental difficulty when working with subdivision surfaces is that the subdivision surface is, in general, not available as a closed-form parametric expression at extraordinary points. For many practical applications such as computation of partial derivatives and curvatures, point and tangent evaluation at arbitrary parameter values is indispensable. Stam showed in his seminal work[Sta98] on Loop subdivision surfaces that exact evaluation can be achieved by diagonalizing the local subdivision matrix and assembling the subdivision surface from regular triangular spline patches whose contributions are scaled by powers of the eigenvalues. This result effectively established subdivision surfaces as a fully parametric primitive on the same footing as classical splines. It has since been the basis of most algorithms that need to query a Loop subdivision surface [MK05, NMM08] and is also used in the library [Stu25a] used for evaluation of subdivision surfaces in this work. Beyond pure surface representation and animation, Loop subdivision has also become an important tool for shape reconstruction [HDD+94, MK05, NMM08] by fitting a subdivision surface (and its control mesh) to a point cloud e.g. obtained from a 3D scan.

2.2 Medial axis approximation

Together with the radius function that maps each medial point to the radius of its associated maximal sphere, the MA forms the medial axis transform (MAT), introduced by Blum in 1967 [Blu67] as a "grass-fire" transform that propagates a wavefront inward from the boundary until opposing fronts meet (these extinction points are the medial points). Blum's motivation were problems in shape perception but the MAT has since become an an important tool in geometry processing since it is a compact, structure-aware, and reversible geometric representation: From the MAT one can recover the original shape as a union of balls and the MAT itself encodes thickness, part hierarchy, and topology in

a lower-dimensional, semantically aligned form [SP08, TDS+16]. In three dimensions, the MAT is used for shape representation and compression [SCYW16], skeletons for animation rigging [BP07], tolerance checking in CAD/CAM [KG13], surgical planning [SPSP02], topology-aware mesh simplification [LWS+15] and adaptive remeshing driven by local feature size (lfs) [BO05].

So the MA has long served as a compact way to describe the geometry and topology of a shape through the centers of its maximal empty spheres which makes it inherently difficult to calculate. Even for clean inputs, small boundary changes can create large changes in the MA, and near non-smooth or noisy regions the lfs can even vanish. Lieutier showed that the MAT retains the homotopy type of open bounded sets, while Chazal and Lieutier analyzed stability under perturbation, giving theoretical support for approximations that preserve the meaningful parts of the skeleton rather than every unstable branch [Lie04, CL05]. Surveys by Attali et al., Siddiqi and Pizer, and Tagliasacchi et al. place this trade off between geometric fidelity, stability, and computational cost at the center of MA research [ABE09, SP08, TDS+16].

Many practical algorithms exploit Voronoi or Delaunay structures because the MA of a sampled boundary is reflected in Voronoi cells, poles, and dual power-diagram configurations. The Power Crust method uses a power diagram view of boundary samples to recover a surface together with an approximate MAT, interpreting the shape through a union of balls [ACK01]. Alpha shapes provide a related geometric language for offsets and ball unions, and have influenced later filtering and reconstruction methods [EM94]. Dey and Zhao proved that, under suitable sampling assumptions, the MA can be approximated from the Voronoi diagram with convergence guarantees, making explicit how sampling density controls the quality of the recovered medial structure [DZ04]. For planar free form boundaries, Aichholzer et al. replace the input by circular arc pieces so that the exact MA of the approximating curve can be computed efficiently, although such replacement can concentrate error when the original curve is represented by higher-order splines such as Bézier segments [AAA+09]. Yang et al. use overlapping spheres along the boundary in both two and three dimensions, so the cost follows the size of the produced medial approximation more closely than the size of the boundary representation; the achievable precision, however, is tied to the density and placement of those spheres [YBM04]. Ramanathan and Gurumoorthy instead march along curved planar boundaries with a user-selected step size, avoiding explicit bisector tracing for efficiency but leaving the approximation error without a formal bound [RG03].

Another major line of work derives skeletons from distance fields. Euclidean distance transforms make the distance-to-boundary function available on a grid, after which medial ridges can be detected as locations where the distance function changes its closest boundary support. Danielsson's distance mapping [Dan80] is an early basis for this type of approach, and fast-marching variants such as the method of Telea and van Wijk [TvW02] compute center lines by following fronts or ridges in a discretized distance landscape. Jalba et al. later proposed a multiscale framework that treats curve, surface, and volumetric skeletonization in a unified way, allowing simplification to be coupled

with the scale at which distance information is analyzed [JST15]. The VoxelCore method combines voxelized input with Voronoi reasoning to produce three-dimensional MA approximations with provable error control [YLJ18]. These approaches are attractive for volumetric data and imaging applications because they are robust and scalable, but the finest structures that can be recovered are limited by the underlying discretization.

Medial representations are also obtained by topological thinning. Such methods iteratively delete boundary pixels or voxels while enforcing constraints that keep the homotopy type unchanged. Zhang and Suen's two-dimensional thinning procedure [ZS84] and Palagyi and Kuba's parallel three-dimensional thinning algorithm [PK99] are representative examples of this approach. Their appeal lies in simple implementation and speed on raster data. Their drawback is that the produced skeleton is mainly a discrete topological object where metric accuracy, placement of branches, and sensitivity to grid resolution typically require subsequent cleanup. Consequently, thinning is often used when topology is more important than exact medial geometry, or as a preprocessing step before pruning and smoothing.

Since an unfiltered MA amplifies small boundary artifacts, much research has focused on selecting or simplifying stable skeletal content. The λ-MA approach and related stability results give a principled way to ignore branches caused by small Hausdorff perturbations [CL05]. The Scale Axis Transform modifies medial ball radii and then reconstructs a simplified skeleton from the scaled representation, producing a hierarchy of shape abstractions that can be tuned by scale [GMPW09]. Optimization-based methods pursue the same goal from an error-minimization viewpoint. Q-MAT formulates the MA transform using quadratic error minimization, and Q-MAT+ extends this idea with error control and feature-sensitive simplification [LWS$^+$15, PWG$^+$19]. These methods are valuable when a compact and visually stable skeleton is needed, although their output is often a simplified medial representation rather than a dense reference suitable for measuring lfs at high precision.

Recent work has mainly moved between two goals: producing fewer skeletal primitives and improving the faithfulness of the medial representation. The emphasis is often on robust abstraction, compression, or downstream usability. The Coverage Axis [DLX$^+$22] searches for a small set of interior medial spheres that still covers the input shape effectively. The medial skeletal diagram [GWM24] groups medial primitives into generalized envelopes, providing a compact representation for closed manifold meshes. Huang et al. formulate variational MA sampling, splitting spheres in regions where the current approximation causes high error between the medial representation and the target shape [HKTB24]. Wang et al. [WHS$^+$24] introduce MATTopo, a topology-preserving 3D MAT method that uses a volumetric restricted power diagram and refines it with local sphere insertion to ensure correct homotopy. Learning-based methods also use skeletons as shape priors: Deep Medial Voxels [PSL$^+$25] predicts medial structures from volumetric medical data and reconstructs anatomy with convolution surfaces, but such models are limited by the scarcity of exact MA ground truth for training and evaluation.

In the context of the present work, which seeks to provide accurate lfs on smooth

geometry, Voronoi and power-diagram approaches provide strong sampling theory but depend on dense boundary discretizations. Distance-field and thinning methods are robust on gridded data but inherit grid resolution limits. Pruning, scale-space, and optimization approaches produce cleaner skeletons, but they intentionally suppress or aggregate medial detail. This motivates the aim to compute a medial representation that is dense, geometrically precise, and provides reliable lfs information, while avoiding the instability and parameter dependence that have shaped much of the earlier work presented here.

2.3 Feature-size-aware sampling

The notion of lfs has long been central to guaranteeing correctness in surface reconstruction. Amenta et al. [ABK98] introduced a Voronoi-based reconstruction algorithm that provably recovers the correct surface topology when the input points sample the surface densely relative to the lfs, using the poles of the Voronoi diagram to estimate the medial axis. In this context, they introduced the rigorous ε-condition that every point must have a sample point closer than ε times its lfs.

Feature sensitivity is equally important when surfaces are extracted directly from volumetric data. Kobbelt et al. [KBSS01] proposed a feature-sensitive extraction scheme that detects sharp edges and corners from directed distance information and places samples so that these features are faithfully reproduced rather than smoothed away. By adapting the mesh to local geometric detail, their method avoids the aliasing artifacts that uniform marching-cubes-style extraction produces at fine features.

Adaptive sampling becomes considerably harder when several materials meet along shared interfaces. Meyer et al. [MWK+08] developed a particle-based approach that distributes samples over the surfaces of multi-material volumes, using inter-particle energies that adapt the local density to surface curvature and feature size. The resulting point sets yield high-quality, feature-aware meshes in which material boundaries and junctions are consistently sampled across all incident materials.

More recently, learning- and estimation-driven pipelines have brought feature-size awareness to raw, unstructured inputs. Fu et al. [FHA24] presented an approach to reconstruct adaptive isotropic meshes directly from unoriented 3D point clouds by estimating the lfs from curvature, thickness, and separation cues. This lfs-aware formulation allocates mesh resolution where geometric detail warrants it, producing adaptive tessellations without requiring oriented normals or a prescribed target density.

In parallel with reconstruction from pre-existing point samples, Delaunay-refinement methods studied how feature size should govern the generation of the samples themselves. Ruppert [Rup95] established the foundational connection between local feature size, mesh grading, and element quality in planar domains, while Chew [Che93] developed guaranteed-quality refinement for curved surfaces. For implicitly represented smooth surfaces, Cheng et al. [CDRR04] provided an adaptive sampling and triangulation

algorithm with guarantees on topology, geometric fidelity, and triangle quality. Boissonnat and Oudot [BO05] subsequently introduced loose ε-samples and a constructive procedure that simultaneously produces a sparse sample and a restricted Delaunay surface mesh. Since these formulations commonly assume access to feature-size information, Rand and Walkington [RW11] addressed the complementary problem of estimating lfs through local Delaunay-refinement operations, avoiding dependence on an exact lfs oracle.

Across most of these settings, the lfs is either estimated from an already discretized input or approximated indirectly. Because a Loop subdivision surface can be evaluated exactly, it offers precisely the setting in which a dense, accurate MA (and thus a precise lfs) could be obtained without discretization error.

CHAPTER 3

Method

We follow a four-step pipeline that creates an ε-sampled mesh from the Loop subdivision surface of a control triangle mesh. The input is a closed manifold triangular control mesh $\mathcal{P}$. The outputs are a triangulated approximation $\mathcal{T}$ of the MA $\mathcal{M}(\mathcal{S})$ of the Loop subdivision surface $\mathcal{S}$ of $\mathcal{P}$, and an ε-sampled closed manifold triangulation $\mathcal{Q}$ of $\mathcal{S}$ with sampling density conforming to the lfs.

The four steps are as follows (after notations in Ch. 3.1):

1. Sampling the MA (Ch. 3.2),
2. Triangulating the MA samples (Ch. 3.3),
3. Creating an ε-sampling of $\mathcal{S}$ (Ch. 3.4),
4. Triangulating the ε-sampling of of $\mathcal{S}$ (Ch. 3.5).

3.1 Notation and definitions

Control mesh. Let $\mathcal{P} = (V, E, \mathcal{F})$ be a closed, connected, orientable triangular manifold mesh with n vertices $V = \{\mathbf{p}_i\}_{i=1}^{n} \subset \mathbb{R}^3$, edge set E, and triangle set $\mathcal{F}$ with size $K = |\mathcal{F}|$. A vertex is called *irregular* if its valence differs from six. We normalize all measures, such as errors, to the diameter D of the axis-aligned bounding box (AABB) of $\mathcal{P}$. Open boundaries, multiply connected meshes, and non-manifold configurations are not supported.

Loop subdivision surface. Applying the Loop subdivision scheme to the control mesh $\mathcal{P}$ tessellates the mesh and in the limit yields a smooth surface $\mathcal{S} \subset \mathbb{R}^3$, which is

C^2 at regular vertices and C^1 at irregular ones [Sta98]. Each triangle $f_k \in \mathcal{F}$, together with its one-ring of control points, defines a *patch* $S_k : T \to \mathbb{R}^3$, where

$$T = \{(u, v) \in \mathbb{R}^2 : u \geq 0,\ v \geq 0,\ u + v \leq 1\} \tag{3.1}$$

defines the barycentric coordinates in the triangle f_k, and $\mathcal{S} = \bigcup_{k=1}^{K} S_k(T)$. The function T for each patch is a piecewise polynomial of degree four [Sta98]. At any $(u, v) \in \text{int}(T)$, the position, and the first and second partial derivatives are available:

$$S_k(u, v), \quad \partial_u S_k, \quad \partial_v S_k, \quad \partial_{uu} S_k, \quad \partial_{uv} S_k, \quad \partial_{vv} S_k \ \in \ \mathbb{R}^3. \tag{3.2}$$

For regular patches, this evaluation is a standard quartic B-spline computation over a 12-point stencil. For patches adjacent to an irregular vertex, Stam's eigenspace analysis [Sta98] decomposes the subdivision surface spectrally; near the extraordinary vertex, the evaluation converges to an exact C^1 limit by subdividing until the query point falls in a regular sub-triangle [Sta98, BLZ00]. However, in our implementation with OpenSubdiv [Stu25a], regions very close to an extraordinary vertex are approximated by Gregory [Gre74] end caps (see also Ch. 5.3). All evaluations are performed via the OpenSubdiv Buffer-Free Refinement (BFR) API [Stu25a]. The unit outward normal at $S_k(u, v)$ is:

$$\mathbf{n}_k(u, v) = \frac{\partial_u S_k \times \partial_v S_k}{\|\partial_u S_k \times \partial_v S_k\|}. \tag{3.3}$$

Medial axis. The *medial axis* $\mathcal{M}(\mathcal{S})$ is the closure of the set of centres of maximal inscribed spheres (medial spheres) in $\mathbb{R}^3 \setminus \mathcal{S}$. A sphere $B(\mathbf{c}, r)$ is *maximally inscribed* if it does not intersect $\mathcal{S}$, is tangent to $\mathcal{S}$ at two or more points, and is not contained in any larger such sphere. For a closed surface, $\mathcal{M}(\mathcal{S})$ has an inner component $\mathcal{M}^-(\mathcal{S})$ (inside the enclosed volume) and an outer component $\mathcal{M}^+(\mathcal{S})$ (outside), which are disjoint.

Local feature size. The *local feature size* at a point $\mathbf{p} \in S$ is

$$\text{lfs}(\mathbf{p}) = d(\mathbf{p},\, \mathcal{M}(\mathcal{S})), \tag{3.4}$$

where $d(\cdot, \cdot)$ is the Euclidean distance. This function is 1-Lipschitz-continuous [Dey07]:

$$\big|\text{lfs}(\mathbf{p}) - \text{lfs}(\mathbf{q})\big| \leq \|\mathbf{p} - \mathbf{q}\| \qquad \text{for } \mathbf{p}, \mathbf{q} \in \mathbb{R}^3. \tag{3.5}$$

ε-sampling. A finite set $\mathcal{Q} \subset \mathcal{S}$ is an *ε-sampling* of $\mathcal{S}$ for $\varepsilon \in (0, 1]$ if for every $\mathbf{p} \in \mathcal{S}$ there exists a sample $\mathbf{s} \in \mathcal{Q}$ with

$$d(\mathbf{p}, \mathbf{s}) < \varepsilon \cdot \text{lfs}(\mathbf{p}). \tag{3.6}$$

The UV-space tolerance is fixed at $\tau_{\text{UV}} = 10^{-9}$, and the spatial tolerance is $\tau_{\text{mesh}} = D \cdot 10^{-m}$ with default $m = 6$ (so $\tau_{\text{mesh}} = 10^{-6} D$). The binary search of Ch. 3.2 terminates after at most $N_{\max}^{\text{sphere}} = 100$ steps, and every Newton iteration after at most $N_{\max}^{\text{Newton}} = 500$ steps.

3.2 Sampling the medial axis

Overview In order to sample the MA, we grow maximal empty tangent spheres on the boundary, whose centers must lie on $\mathcal{M}(\mathcal{S})$, resulting in the MAT for free: For each foot point F_i on a uniform tessellation of $\mathcal{S}$, we compute the center $\mathbf{c}_i^-$ and radius r_i^- of the maximal inner sphere tangent to $\mathcal{S}$ at F_i and empty of $\mathcal{S}$, and likewise $\mathbf{c}_i^+$, r_i^+ for its outer pendant. Because $\mathcal{S}$ is smooth, the center of such a sphere is constrained to the normal line at F_i, reducing the search to a one-dimensional problem in the sphere radius r. The center of the maximal inner sphere at F_i lies on the inward normal ray such that $\mathbf{c}_i^- = F_i - r_i^-\mathbf{n}_i$, where r_i^- is the largest $r > 0$ such that $\mathcal{S} \cap int(B(\mathbf{c}(r), r)) = \emptyset$ and $B(\mathbf{c}(r), r)$ is tangent to $\mathcal{S}$ at a second point besides F_i. We find r_i^- by binary search: at each step, the closest point on $\mathcal{S}$ to the current candidate center $\mathbf{c}(r) = F_i - r\mathbf{n}_i$ is computed by a Newton(-Raphson) minimization on each candidate patch, and r is enlarged or reduced accordingly in a binary search.

User parameter: footpoint tessellation level. The level of tessellation for the footpoints ℓ (default: $\ell = 4$) determines the density of the samples from the MA. Higher ℓ yields a finer approximation of $\mathcal{M}(\mathcal{S})$, at a cost proportional to ℓ^2 in both the number of binary-search problems and the number of Newton start-value evaluations. The secondary tessellation level ℓ' for Newton start values is set independently (default: $\ell' = 64$); increasing ℓ' reduces Newton iterations at the cost of more start-value evaluations, avoiding local minima. See Fig. 3.1 for the tesselation pattern.

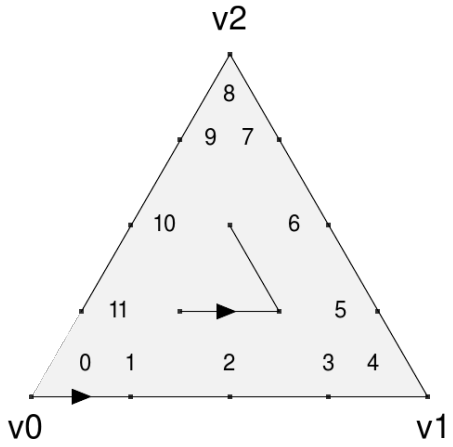

Figure 3.1: Tessellation scheme at tessellation level four for a triangle as introduced in the OpenSubdiv Bfr Documentation [Pixa]. Image courtesy of Pixar [Stu25b]

.

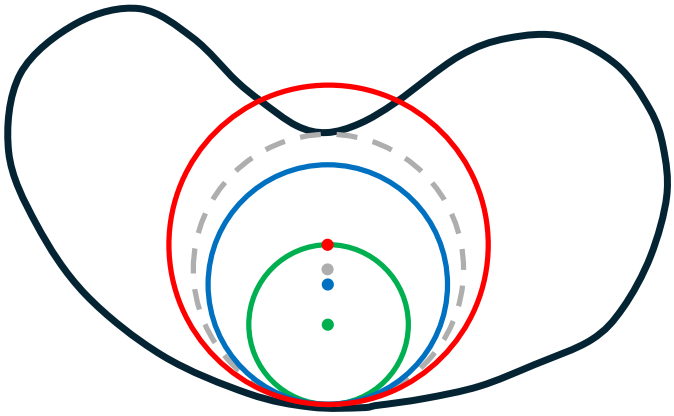

Figure 3.2: The sphere shrinking process as a bisection process for approximating the medial sphere (dashed line). The red sphere intersects the boundary and is the current upper bound. The green sphere touches the boundary only at the foot point and thus is the current lower bound. The blue sphere is the median of the upper and lower bound, does not intersect the boundary and thus becomes the new lower bound.

Binary search. The search is initialized with bounds $r_{\min} = 0$ and $r_{\max} = r_0 + \tau_{\text{mesh}}$ (for the initial value r_0 see *Initialization of* r below in Ch. 3.2). At each step the midpoint $r \leftarrow \frac{1}{2}(r_{\min} + r_{\max})$ is evaluated (see Fig. 3.2). Let $d^*(r) = \min_{k \in \mathcal{C}(r)} d_k(r)$ be the distance from $\mathbf{c}(r)$ to its closest point on $\mathcal{S}$. Here $d_k(r)$ is the distance of $\mathbf{c}(r)$ to the closest point on patch k, obtained from the Newton scheme of Ch. 3.2 *Closest-point query* below and $\mathcal{C}(r)$ is the set of patches with a convex hull of control points that intersect $B(\mathbf{c}(r), r)$). The patch containing F_i is always added to $\mathcal{C}(r)$.

Tessellation pre-check. Before invoking Newton, the level-ℓ' tessellation vertices of every patch in $\mathcal{C}(r)$ are tested against $\mathbf{c}(r)$. If any vertex is closer than $r - \tau_{\text{mesh}}$ to $\mathbf{c}(r)$, the secondary tangency criterion is satisfied without invoking Newton at this step. Likewise, the Newton scheme of Ch. 3.2 returns as soon as it produces an iterate closer than $r - \tau_{\text{mesh}}$: it is used as a sufficient test, not to locate the global closest point. The bounds are updated as follows. Let $\mathbf{p}^*$ be the closest surface point found at $\mathbf{c}(r)$ by the procedure above.

- If $d(\mathbf{p}^*, \mathbf{c}(r)) < r - \tau_{\text{mesh}}$ (secondary intersection): $r_{\max} \leftarrow r$.
- Otherwise (no point closer than r is found, the only such point coincides with F_i): $r_{\min} \leftarrow r$.

Convergence is declared when $|r - r_{\text{old}}| < \tau_{\text{mesh}}$ or after $N_{\max}^{\text{sphere}}$ steps where r_{old} is the solution from the previous step. The outer sample $\mathbf{c}_i^+ = F_i + r_+^* \mathbf{n}_i$ is computed with negated $\mathbf{n}_i$. All footpoints are processed independently and in parallel on the CPU.

Closest-point query

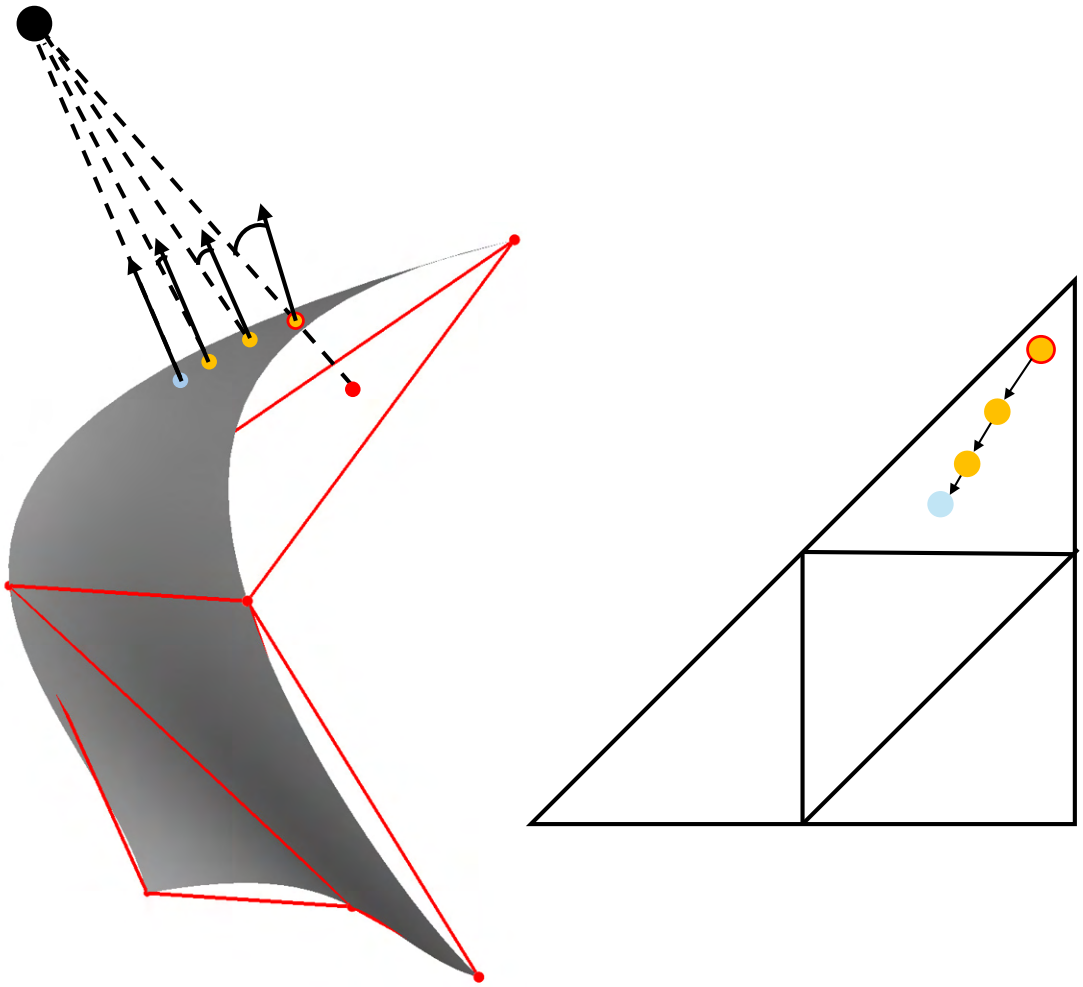

Figure 3.3: The Newton process for finding the closest point of a surface patch to a given reference point (black) finds the zero of the function that measures the angle between the normal of the current solution and the line segment between the current solution and the reference point. The closest point (red) on a tesselation of the patch serves as the starting point.

Goal. Given a query point $\mathbf{q} \in \mathbb{R}^3$ and a patch k, find $(u^*, v^*) = \arg\min_{(u,v)\in T} \|S_k(u,v) - \mathbf{q}\|^2$.

Overview. The squared-distance objective

$$f_k(u,v) = \|S_k(u,v) - \mathbf{q}\|^2 \tag{3.7}$$

for the quartic patch is an 8th-order polynomial or rational function on $\mathrm{int}(T)$, so we can find its minimum with Newton iterations using the gradient ∇f_k and Hessian H_k. This process iteratively minimizes the angular deviation between the surface normal and the line segment to $\mathbf{q}$, as depicted in Fig. 3.3.

The gradient and Hessian are

$$\nabla f_k = 2 \begin{pmatrix} \langle \partial_u S_k,\ S_k - \mathbf{q} \rangle \\ \langle \partial_v S_k,\ S_k - \mathbf{q} \rangle \end{pmatrix}, \tag{3.8}$$

$$H_k = 2 \begin{pmatrix} \langle \partial_u S_k, \partial_u S_k \rangle + \langle \partial_{uu} S_k, S_k - \mathbf{q} \rangle & \langle \partial_u S_k, \partial_v S_k \rangle + \langle \partial_{uv} S_k, S_k - \mathbf{q} \rangle \\ \langle \partial_u S_k, \partial_v S_k \rangle + \langle \partial_{uv} S_k, S_k - \mathbf{q} \rangle & \langle \partial_v S_k, \partial_v S_k \rangle + \langle \partial_{vv} S_k, S_k - \mathbf{q} \rangle \end{pmatrix}. \tag{3.9}$$

The update at iterate $(u^{(t)}, v^{(t)})$ is:

$$(u^{(t+1)}, v^{(t+1)}) = (u^{(t)}, v^{(t)}) + \delta^{(t)}, \qquad \delta^{(t)} = -H_k^{-1} \nabla f_k. \tag{3.10}$$

On average, 3-4 iterations are required in test cases, as expected due to the Newton scheme's quadratic convergence. Convergence is declared when the tangential residual $\rho_{\tan} = \sqrt{\langle r, e_1 \rangle^2 + \langle r, e_2 \rangle^2} < \tau_{\mathrm{mesh}}$, where $r = S_k(u, v) - \mathbf{q}$ and (e_1, e_2) is the Gram–Schmidt orthonormalization of $(\partial_u S_k, \partial_v S_k)$ (if $\partial_v S_k$ is collinear with $\partial_u S_k$ the tangent plane degenerates to one dimension and only $\langle r, e_1 \rangle$ enters). The iteration is also stopped after 100 consecutive iterations with $|\sqrt{f_k^{(t+1)}} - \sqrt{f_k^{(t)}}| < \tau_{\mathrm{mesh}}$, and after $N_{\max}^{\mathrm{Newton}}$ steps. When the full Hessian fails to be positive-definite ($\det H_k \le \tau_{\mathrm{UV}}$ or $(H_k)_{11} \le \tau_{\mathrm{UV}}$), we fall back to the Gauss–Newton approximation $H_k^{\mathrm{GN}} = 2 J_k^T J_k$ with $J_k = [\partial_u S_k \mid \partial_v S_k] \in \mathbb{R}^{3\times 2}$; if H_k^{GN} is itself near-singular ($|\det H_k^{\mathrm{GN}}| < \tau_{\mathrm{UV}}$), we apply a Levenberg-Marquardt [Lev44] diagonal shift $H \leftarrow H + \lambda I$ with $\lambda = \tau_{\mathrm{UV}} \max(h_{11}, h_{22}, |h_{12}|)$; if H is still singular after the shift, we take a steepest-descent step $\delta^{(t)} = -\tau_{\mathrm{UV}} \nabla f_k / \|\nabla f_k\|$.

Brent line search. A Newton step can overshoot, increasing the objective f_k. If $f_k^{(t+1)} > f_k^{(t)}$ we replace the step by a 1D minimization of $\phi(s) = f_k((u^{(t)}, v^{(t)}) + s\,\delta^{(t)})$ on $s \in [0, 1]$ using Brent's method [Bre73] (with 20-bit accuracy which is sufficient here since the Newton iteration has already produced a good direction). If Brent fails to improve on $f_k^{(t)}$, the iterate is reverted, and the iteration terminates. Brent's method finds a local minimum by combining Golden Section Search and Parabolic Interpolation. It uses parabolic interpolation for speed and falls back to golden section search for safety. Double precision means 24-bit precision here. Since this method is only a fallback to get the Newton scheme back on track, we only use 20 bits to speed things up and rely on the Newton loop to reach the desired precision.

Start value and boundary handling. We initialize (u_0, v_0) by projecting $\mathbf{q}$ onto the surface image of the level-ℓ' tessellation of patch k: the closest tessellation triangle yields barycentric coordinates which we convert to T-coordinates and nudge $2\,\tau_{\mathrm{UV}}$ inward

whenever any coordinate would lie on ∂T. If an iterate $(u^{(t+1)}, v^{(t+1)})$ falls outside T, we proceed as follows. If it is within τ_{UV} of a vertex of T, we snap it to that vertex. Otherwise we replace it by the intersection of the segment $[(u^{(t)}, v^{(t)}), (u^{(t+1)}, v^{(t+1)})]$ with ∂T, then the Newton loop breaks. When the 2D Newton terminates with an iterate on ∂T, the solution is refined by a 1D Newton sweep along the closest edge (see below).

1D boundary refinement. Let $\gamma : [0, 1] \to \partial T$ be the affine parametrization of the closest edge of T and $g_\gamma(t) = \|S_k(\gamma(t)) - \mathbf{q}\|^2$. We sample $2\,\ell'$ uniformly spaced points $\gamma(t_i)$, project $\mathbf{q}$ onto the resulting polyline to obtain a starting value t_0, then run a 1D Newton iteration using $g'_\gamma(t) = 2\langle S_k(\gamma(t)) - \mathbf{q},\, dS_k/dt\rangle$ and $g''_\gamma(t) = 2\|dS_k/dt\|^2 + 2\langle S_k(\gamma(t)) - \mathbf{q},\, d^2S_k/dt^2\rangle$. When $g''_\gamma(t)$ is non-positive, the curvature term is dropped (1D Gauss–Newton). If the unconstrained Newton step leaves $[0, 1]$, or worsens g_γ, we invoke Brent's method between t and t_{new}.

Spatial acceleration and initialization

Patch bounding volume hierarchy. Evaluating all K patches at every binary-search step would be prohibitively expensive. The patches of the Loop subdivision surface are contained in the convex hull of their control points [Sta98]; a further, more relaxed bound is its AABB. We therefore can test the sphere-patch intersection using an AABB tree in $O(\log K)$ time for sphere-box intersection and then test the resulting AABBs against a precomputed triangulation of the convex hulls of their patches to reduce the number of candidate patches (partially) contained in $\mathcal{C}(r)$. On average, $O(K)$ patches remain to be queried for the closest point, implying $O(K^2)$ time complexity for calculating all spheres on all patches.

Initialisation of r. To reduce the initial search radius r_0 for a foot point F from the maximum D, we first test all patches $S_k \in S_F$ contained entirely in the negative halfspace of its normal direction, as these each provide a tighter upper bound for r. The bound for a patch S_k corresponds to the smallest tangential sphere at F in direction of its inward normal $\mathbf{n}$ enclosing all its control points $\mathbf{p}_j^k$:

$$r_k = max_j(r_j^k), \quad r_j^k = \frac{\|F - \mathbf{p}_j^k\|^2}{2\,\langle \mathbf{n}, F - \mathbf{p}_j^k\rangle}. \tag{3.11}$$

The upper bound for S_F is the smallest sphere that completely encloses the convex hull of a patch, with radius $r_0 := \min_{k \in S_F} r_k$. If no patch in S_F has its convex hull entirely on the inward side of the tangent plane at F (so no positive radius is produced by Eq. (3.11)), we fall back to $r_0 = D$ for the inner sphere and $r_0 = 10\,D$ for the outer sphere.

3.3 Meshing the medial axis samples

Goal. Construct a triangle mesh $\mathcal{T}^{\pm}$ from the MA samples $\{\mathbf{c}_i^{\pm}\}$ that supports fast distance queries for approximating the lfs.

Topology inheritance. A naive application of Delaunay triangulation to the MA point cloud did not work in most cases: the irregular geometry of $\mathcal{M}(\mathcal{S})$ with leaves and branches produced configurations that Delaunay could not reliably resolve. The foot points, however, already form a triangulated surface with well-defined combinatorics which can be transferred to the medial points: Each foot point F_i generates a medial point $\mathbf{c}_i^{\pm}$. Since foot points are vertices of a (tessellated) triangulation T_ℓ on $\mathcal{S}$, their triangles $(i, j, k) \in T_\ell$ map bijectively onto triangles $(\mathbf{c}_i^{\pm}, \mathbf{c}_j^{\pm}, \mathbf{c}_k^{\pm})$ in $\mathcal{T}^{\pm}$ on the MA. The resulting triangle soup will self-intersect, which is not an issue for distance queries on $\mathcal{T}^{\pm}$. At this point, we only store a triangle soup rather than the complete inherited mesh since we only need unorganized triangles for the distance queries for an ε-sampling.

Distance queries and LFS approximation. For a query point $\mathbf{q}$, the lfs is approximated as:

$$\mathrm{lfs}(\mathbf{q}) \approx \min\big(d(\mathbf{q}, \mathcal{T}^-),\ d(\mathbf{q}, \mathcal{T}^+)\big), \tag{3.12}$$

To accelerate these distance queries, we construct an AABB tree for each of $\mathcal{T}^-$ and $\mathcal{T}^+$, yielding $O(\log N_\ell)$ time complexity for N_ℓ triangles for tessellation level ℓ. This approximation converges to the exact lfs as $\ell \to \infty$. It has been shown [Ohr] that the lfs error at a surface point $\mathbf{p}$ has an upper bound in the error of the MA at its closest point m since the lfs error is the error of m if $\overrightarrow{\mathbf{pm}}$ is orthogonal to the MA and smaller otherwise (triangle inequality).

3.4 Creating an ε-sampling

Goal. Compute a finite set $\mathcal{Q} \subset \mathcal{S}$ satisfying the ε-condition (3.6) everywhere on $\mathcal{S}$, given $\varepsilon \in (0, 1]$ as a user parameter.

Overview. An optimal ε-sampling that minimizes $|\mathcal{Q}|$ would require solving a difficult global optimization problem on a curved surface, which is beyond the scope of this work. Instead, we simply exploit the patch decomposition of $\mathcal{S}$: it suffices to verify the ε-condition on each patch independently, and to subdivide any patch that does not yet satisfy it. Because lfs is 1-Lipschitz, a single sphere with its center on the patch that is enclosing the patch acts as a proxy: if the condition holds at the center, it holds everywhere on the patch.

Algorithm. We initialize $\mathcal{Q} = \mathcal{V}$. Each patch $S_k \in \mathcal{S}$ can be recursively subdivided in the triangular parameter domain T into four similar sub-triangles. For each sub-triangle Δ with vertices $\mathbf{a}, \mathbf{b}, \mathbf{c} \in T$, we compute an enclosing sphere of its surface image, the sub-patch $S_k(\Delta)$, with center $c \in S_k(\Delta)$ and radius r. If the condition

$$\mathrm{lfs}(c) > \frac{3\,r}{\varepsilon} \tag{3.13}$$

holds, we stop at that sub-patch. Otherwise, we split Δ into four new sub-triangles by inserting edge midpoints, and add these to $\mathcal{Q}$. We discard any Δ with minimum height smaller τ_{mesh} as negligible.

Correctness. Let $\mathbf{s}$ be any vertex of Δ (already in $\mathcal{Q}$) and $\mathbf{p} \in S_k(\Delta)$ be any surface point. Since $S_k(\Delta) \subset B(c, r)$, we have $d(\mathbf{p}, \mathbf{s}) \leq 2r$. By the Lipschitz property (3.5) and condition (3.13):

$$\text{lfs}(\mathbf{p}) \geq \text{lfs}(c) - r > \frac{3r}{\varepsilon} - r = r\Big(\frac{3}{\varepsilon} - 1\Big),$$

hence, using $\varepsilon \leq 1$, the following satisfies condition (3.6):

$$\varepsilon \cdot \text{lfs}(\mathbf{p}) > \varepsilon \cdot r\Big(\frac{3}{\varepsilon} - 1\Big) = r(3 - \varepsilon) \geq 2r \geq d(\mathbf{p}, \mathbf{s}),$$

where the penultimate step uses $\varepsilon \leq 1$, satisfying Condition (3.6).

Patch-enclosing sphere calculation

Goal. Given a sub-triangle Δ with vertices $\mathbf{a}, \mathbf{b}, \mathbf{c} \in T$, find a point $c_0 \in S_k(\Delta)$ and radius r such that $S_k(\Delta) \subset B(c_0, r)$, with r as small as practical (since the smaller r the more efficient the algorithm).

Algorithm. The problem reduces to maximizing the distance from an assumed center c_0 over the surface image of Δ. The farthest point from c_0 on $S_k(\Delta)$ is found by a 2D Newton maximization on the interior, similar to the minimization in Ch. 3.2, and separate 1D Newton sweeps on each edge, since on a bounded domain the maximum can lie in the boundary without the first derivative vanishing there. To obtain a center c_0, we tessellate the subtriangle Δ at level ℓ' (additionally to its base level) to obtain $M = (\ell' + 1)(\ell' + 2)/2$ surface points $\{P_j\}$. For each i, let $\rho(i) = \frac{1}{2} \max_j \|P_i - P_j\|$. We choose $c_0 = P_{i^*}$ with $i^* = \arg\min_i \rho(i)$ and initialize the farthest-point search at $j^* = \arg\max_j \|P_{i^*} - P_j\|$ (see Figure 3.4). We apply this procedure to all patches in parallel.

3.5 Triangulating the ε-sampling

Goal. Given the sample set $\mathcal{Q}$ with UV coordinates, produce a triangle mesh that covers $\mathcal{S}$ consistently across patch boundaries.

Algorithm. A 2D triangulation in UV parameter space is well-posed and lifts correctly to 3D. Samples on shared patch edges must be hard constraints so that adjacent patches produce the same edge. We use the 2D Constrained Delaunay Triangulation (CDT) [Che87] that retains such edges while maximising triangle quality. For visual smoothness, we remove collinear triangles on edges, which leaves very small holes.

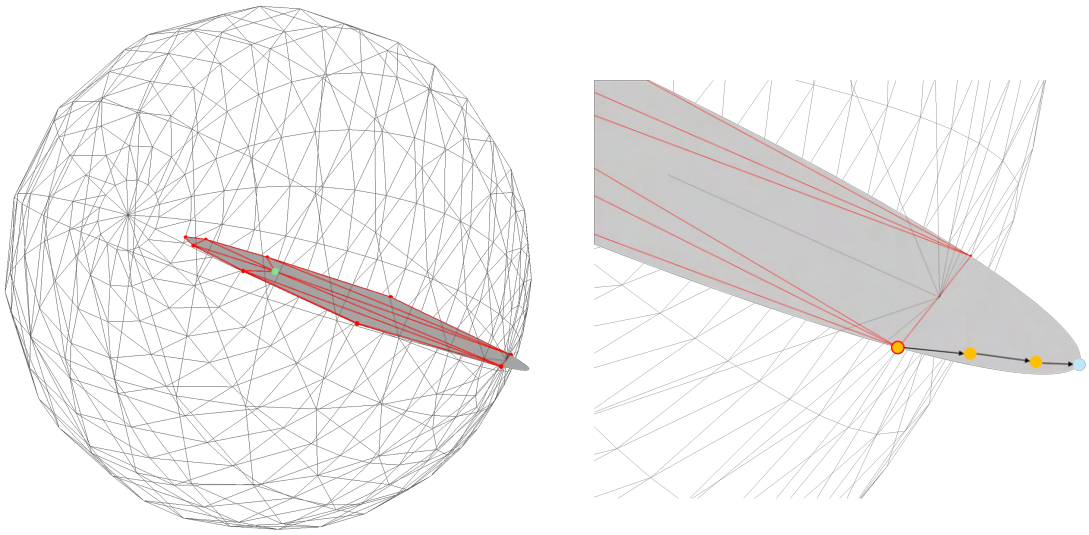

Figure 3.4: Calculating the patch-enclosing sphere. Left: First, we determine the center c_0 (green) by selecting the tessellation vertex with the minimum distance to any other tessellation vertex of the patch (red). This yields an initial guess for the sphere. Right: Then we initialize a farthest point search over the surface with the tessellation vertex that is farthest from c_0.

Triangulation. For each patch k, let $\{(u_i, v_i)\}$ be the UV coordinates of the samples in $\mathcal{Q}_k = \mathcal{Q} \cap S_k(T)$. A CDT of this 2D point set is computed using CGAL's `Constrained_Delaunay_triangulation_2` [Prob]. Samples lying on ∂T (UV distance below τ_{UV} to an edge of T) are inserted as constrained edges, enforcing consistency with adjacent patches. Each UV vertex is then replaced by its 3D position in $\mathcal{Q}_k$ to obtain the output mesh.

Global index assignment and degenerate triangle removal. Interior samples receive fresh global indices at creation. Edge samples use the canonical half-edge key $h^*(e)$ from Ch. 3.4 so that each boundary sample appears only once in the global vertex list. CDT triangles whose three UV vertices are collinear (all on the same edge of T, a consequence of numerical rounding) are detected and removed. This leaves a negligible gap in sub-patches but improves renderings visually.

CHAPTER 4

Implementation

This chapter is concerned with the software architecture, the third-party libraries and the way they are combined, the data structures that hold the tessellations and surface derivatives, the parallelisation strategy, and the numerical and engineering measures that make the method usable in practice. Our prototype [RO26] is written in C++17 and, apart from a handful of small header files (`meshLoader.h`, `objWriter.h`, `far_utils.h`, `shape_utils.h`, `fileAndInputManagement.h`, `saveVectorToFile.h`), is contained in a single translation unit, `LoopEpsilonSampling.cpp`. The central abstraction is the evaluation of the Loop subdivision surface, which is delegated in its entirety to Pixar's OpenSubdiv library [Stu25a]. Every other component of the system is organized around the ability to sample points, derivatives, and normals of that surface. The remainder of the chapter therefore begins with the overall architecture and build, then treats the OpenSubdiv usage in detail, the geometric data structures built on top of CGAL[Proa], the parallelisation, and the numerical robustness.

4.1 Software architecture and build

Class decomposition. The program is structured around three principal classes and one lightweight result record. `SubdivSurfEvaluator` is a thin, stateful wrapper around the OpenSubdiv Buffer-Free Refinement (BFR) API that owns the control-mesh vertex buffer and a surface factory, and exposes point/derivative evaluation on any control face. `LoopMedialPoints` implements Steps 1 and 2 of the pipeline from Ch. 3: it tessellates the limit surface into foot points, builds the spatial acceleration structures, grows the inner and outer medial spheres, and inherits the tessellation connectivity onto the medial samples. `LoopEpsSampling` implements Steps 3 and 4: it holds the MA triangle soups (as distance-query oracles for the lfs), performs the adaptive per-patch subdivision, and triangulates the resulting samples in UV space. The record `SubDivSurfEvalResult` bundles the position and all five partial derivatives returned

by a single surface evaluation and offers accessors that convert the raw `double` arrays into CGAL `Point` and `Vector` objects. This separation mirrors the pipeline stages of Ch. 3 and keeps the surface evaluation concern isolated behind a single interface.

Dependencies. Five external libraries are used:

- OpenSubdiv [Stu25a] provides topology refinement, patch construction, and limit-surface evaluation of the Loop scheme.
- CGAL 6.1 [Proa] supplies the geometric kernel, convex hulls, bounding-volume hierarchies, and the constrained Delaunay triangulator.
- GLM [Ric05] provides the compact two-component vector type `glm::dvec2` that is used throughout for UV coordinates (This avoids confusion of UV and 3D quantities).
- Eigen [JG+10] and Boost.Math [Boo26] provide dense linear algebra and Brent's one-dimensional minimizer is used inside the Newton closest-point solver (see Ch. 3.2).

Behind the scenes, the GNU Multiple Precision library (GMP) is linked as CGAL's exact-arithmetic backend. The parallel algorithms of the C++ standard library are backed by Intel Threading Building Blocks (TBB) on the Linux/`libstdc++` tool-chain.

Build integration. The implementation was built and tested on Linux under Ubuntu 24.04 LTS (gcc 13.3) as well as on Windows 11 using Visual Studio 2026 with the vcpkg package manager. All builds used the C++17 standard. With MSVC, two compiler settings are required, namely `/bigobj` (the heavily templated OpenSubdiv and CGAL headers exceed the default object-section limit) and `_CRT_SECURE_NO_WARNINGS`. The C++ standard is fixed to C++17, which is required both for the parallel algorithms and for `std::filesystem`, `std::optional` and structured bindings used in the code.

Command-line interface and run modes. The executable is driven entirely from the command line and supports three modes selected by the first argument. Mode `axis` computes and caches only the medial axis. Mode `sampling` runs the full pipeline including the ε-sampling and its triangulation. Mode `analysis` loads a previously computed axis and evaluates sampling error at an independent, finer tessellation level. All modes share the same positional parameters: the input OBJ file name (resolved relative to an `inputMeshes` folder), the foot-point tessellation level ℓ, the secondary tessellation level ℓ' used for Newton start values, a flag selecting the Newton derivative order, and the tolerance order of magnitude m (see Ch. 3.1). The mode `sampling` additionally takes ε, and `analysis` an analysis tessellation level. Invalid or missing arguments cause a usage message to be printed and the program to exit. These parameters are exactly the user parameters discussed in Ch. 3.

On-disk caching and output layout. Recomputing the medial axis is the most expensive stage, so results are cached on disk as OBJ files and reused across runs. On start-up, the program ensures a fixed tree of output sub-folders exists in the folder `outputMeshes`:

- `medialAxisPointClouds` for samples of inner and outer medial points,
- `medialAxisTriangulations` for triangle meshes and soups of the inner and outer medial axis,
- `medialSpheres` for triangulations of the inner and outer medial spheres, only written to when debug variable `saveMedialSpheres` is activated,
- `surfaceTesselations` for tesselation meshes of subdivision surface tesselations only written to when debug variable `saveSurfaceTesselations` is activated,
- `epsMeshes` for triangle meshes of the ε-samplings of subdivision surfaces,
- `epsSampling` for point clouds that are ε-samplings of subdivision surfaces,
- `errorAnalysis` for results of mode `analysis`,
- `tesselatedPatches` for separate tesselations of every patch of a subdivision surface, only written to when debug variable `savePatches` is active.

Output file names encode the mesh name together with the tessellation levels and tolerance, e.g. `motherChild_tessLevels=4_12_tol=5_inner_medialAxis.obj`, so that runs with different parameters do not collide. Before Step 1, the program checks with `std::filesystem::exists` whether both the inner and outer medial-axis triangulations are already present in the folder `medialAxisTriangulations` for the requested parameters. If so, and the mode is not `axis`, the MA is loaded from disk (via `CGAL::IO::read_polygon_mesh`) instead of being recomputed, and the pipeline proceeds directly to the ε-sampling. This turns the otherwise costly MA computation into a one-time cost per (mesh, ℓ,ℓ' m) combination.

4.2 Evaluating the Loop limit surface with OpenSubdiv

All access to the geometry of the subdivision surface (positions, first and second derivatives, normals, patch control points, and uniform tessellations) goes through OpenSubdiv. This section describes how the control mesh is imported, how the BFR surface objects are constructed and configured, how they are evaluated, and how the results are adapted to the rest of the system.

4.2.1 Importing the control-mesh topology

The pipeline consumes triangular control meshes in Wavefront OBJ format. The file is first parsed into OpenSubdiv's internal `Shape` representation, and a `Far::TopologyRefiner` is then constructed from it. The subdivision scheme is fixed to `Sdc::SCHEME_LOOP` and the scheme options are set to edge-only boundary interpolation and uniform creasing. The refiner defines the topology and the BFR API evaluates the subdivision surface directly from this base topology.

4.2.2 Constructing BFR surfaces and the surface factory

The BFR API models the limit surface as a collection of per-face `Surface` objects created on demand by a `RefinerSurfaceFactory`. The wrapper `SubdivSurfEvaluator` owns one such factory together with the control-vertex buffer. The factory is configured through `RefinerSurfaceFactory::Options`. The two relevant settings are the smooth and sharp *approximation levels*, both set to 10:

```
SurfaceFactory::Options opts;
opts.SetApproxLevelSmooth(6);
opts.SetApproxLevelSharp(4);
```

These approximation levels govern how OpenSubdiv treats faces near sharp edges and extraordinary vertices.
`SetApproxLevelSharp` controls how many levels of adaptive refinement (feature isolation) the factory uses to approximate the limit surface around sharp and semi-sharp features (creases, corners, etc.). We do not allow such features; therefore this setting is not relevant for us.
`SetApproxLevelSmooth` governs patch approximation near extraordinary vertices, where the Loop limit surface is not a single polynomial. Rather than evaluating Stam's exact eigen basis limit, OpenSubdiv isolates the irregular region by local refinement up to this approximation level and then represents the remaining, still-irregular neighborhood of the extraordinary vertex by a *Gregory patch* (a Gregory end cap) [Gre74]. A level of 6 means the irregular vertex is isolated through six rounds of local refinement before the end cap is fitted, so that the end cap covers only a parametrically tiny neighborhood of the extraordinary vertex. Away from that neighborhood, the evaluation is the exact quartic box spline patch. This is the source of the small, controlled approximation near extraordinary vertices already noted in Chapter 3, and its effect is analyzed in Ch. 5.3. Every foot point, closest-point query, and ε-sample is evaluated through the same factory, so the whole pipeline is consistent with respect to this representation.

For each control face, the wrapper obtains its surface with `InitVertexSurface`, which returns `false` for faces that cannot be evaluated (mainly holes, and in rare configurations boundary faces). Such faces are skipped, and the corresponding result is flagged invalid rather than causing a failure. Usually we expect the control mesh not to have any holes, though.

4.2.3 Evaluating positions and derivatives

For every requested UV coordinate, the surface is evaluated with a single call that returns the position together with all five partial derivatives required downstream:

```
faceSurface.Evaluate(&uv[2*i], patchPoints.data(), 3,
                     &outPos[j], &outDu[j], &outDv[j],
                     &outDuu[j], &outDuv[j], &outDvv[j]);
```

The second derivatives $\partial_{uu}S_k$, $\partial_{uv}S_k$, and $\partial_{vv}S_k$ are requested because they populate the Hessian of the closest-point objective (3.9). Obtaining them in the same call avoids re-evaluating the surface. The flat `double` output arrays are wrapped in a `SubDivSurfEvalResult`, whose accessors (`pos()`, `du()`, ..., `dvv()`) return CGAL types so that the geometric code never manipulates raw offsets. Convenience overloads (`evaluateAsPoint`, `evaluateDu`, `evaluateAsTuple`, ...) exist for the many call sites that need only a subset of the outputs. All coordinates are stored in a structure-of-arrays layout ($x_1y_1z_1x_2y_2z_2\ldots$) matching OpenSubdiv's expectations, which also makes the buffers directly writable to OBJ.

4.2.4 Uniform tessellation

Two uniform tessellations of every patch are produced up front and reused throughout the pipeline: a coarse one at level ℓ whose vertices are the foot points of medial spheres and the finer one at level ℓ' that supplies discrete start values for the Newton searches (see Ch. 3.2 and 3.4. Both are generated with the BFR `Tessellation` class. For a given patch, a `Tessellation` is built from the surface's `Parameterization`, a uniform rate, and a `Tessellation::Options` object whose facet size is set to 3 (triangular facets; the code also supports quads for Catmull-Clark meshes but the Loop pipeline always uses triangles). The tessellation then yields the UV coordinates of its vertices via `GetNumCoords`/`GetCoords`, and its connectivity via `GetNumFacets`/`GetFacets`. Facet indices are stored twice: once in patch-local numbering (`localFacetIndices`) and once offset into a global vertex numbering with `TransformFacetCoordIndices`, so that the concatenation of all patch tessellations forms a single indexed surface mesh. This global tessellation is written to the folder `surfaceTesselations/` and later re-read as a CGAL `Surface_mesh`. It is the triangulation whose combinatorics are inherited onto the medial samples in Ch. 3.3. The vertices' UV coordinates, positions, and all derivatives are cached in a `TesselatedPatch` record per patch, so that the surface is tessellated only once even though foot point processing revisits every vertex many times.

4.3 Geometric data structures with CGAL

All non-subdivision geometry (bounding volumes, convex hulls, distance queries, and the final planar triangulation) is built with CGAL.

The kernel is `Exact_predicates_inexact_constructions_kernel` (EPICK), chosen because the pipeline needs exact geometric *predicates* (orientation and in-sphere tests inside the convex-hull and Delaunay code must be robust) but only floating-point *constructions* (coordinates of hull vertices, box corners, and triangulation points are used numerically and do not need to be exact). For example, the convex-hull construction explicitly relies on this kernel to work correctly. EPICK's exact predicates prevent the combinatorial failures that a purely inexact kernel produces on near-degenerate patch hulls. From this kernel the usual aliases `Point`, `Vector`, `Triangle`, `Plane`, `Sphere`, `Iso_cuboid_3`, and `Polyhedron_3` are derived.

Per-patch convex hulls and the patch bounding-volume hierarchy. The spatial acceleration of Ch. 3.2 is realized as a two-level structure. For every patch, the convex hull of its control points is computed with `CGAL::convex_hull_3` and stored as a `Polyhedron`. The axis-aligned bounding box (AABB) of that hull is stored as an `Iso_cuboid_3`. To index the boxes, the implementation defines a custom AABB primitive pair, `BoxWithId` (a box plus its patch index) and `BoxWithIdPrimitive`, which adapts `BoxWithId` to CGAL's `AABB_tree` interface by exposing the box as the primitive's `Datum` and a corner as its `reference_point`. A single `AABB_tree` over these primitives answers box-sphere overlap queries in $O(\log K)$ time and returns the offending patch indices via the stored id. The candidate set $\mathcal{C}(r)$ (see Ch. 3.2) is then refined by testing the sphere against the exact convex-hull triangles of the surviving patches, for which each hull is additionally stored as a `std::list<Triangle>`. The tree over hull boxes is thus a cheap, conservative filter in front of the more precise hull test. Because CGAL's `AABB_tree` is not safely copyable or movable, the per-patch triangle lists are stored in a pre-sized `std::vector` and referenced in place rather than appended, an implementation detail required to keep the trees valid.

MA distance oracle. The lfs approximation (3.12) reduces to unsigned distance queries against the two MA triangle soups $\mathcal{T}^-$ and $\mathcal{T}^+$. Each soup is stored as a `std::vector<Triangle>` and indexed by an `AABB_tree` built from `AABB_triangle_primitive`. `accelerate_distance_queries()` is called once so that closest-point queries use the tree's internal KD-tree [Ben75] of primitive reference points. The lfs at a query point is then the minimum of the two `squared_distance`-based results, as in (3.12). Only unsigned distance is queried, which is well defined regardless of the self-intersecting triangle soup.

Constrained Delaunay triangulation of the samples. The per-patch UV triangulation of Ch. 3.5 is delegated to CGAL's `Constrained_Delaunay_triangulation_2` (wrapped in `Constrained_triangulation_plus_2`) for its constraint-management utilities [Prob]. The helper `buildCDT` takes the two-dimensional sample coordinates and a list of index pairs describing the edges that must be preserved (the samples lying on shared patch boundaries). It inserts all points, recording each returned `Vertex_handle` in a map so that constraints can be expressed by sample index. It then inserts each

boundary edge with `insert_constraint`, guarding against out-of-range and degenerate (identical-endpoint) constraints. Finally, it goes through the finite faces and emits triangles as index triples into the patch's local numbering. Constraining the shared-edge samples guarantees that two patches meeting along a control-mesh edge generate identical boundary edges, which is what makes the union of per-patch triangulations watertight across patch seams. The collinear triangles that occasionally arise on ∂T from rounding are detected and dropped afterwards.

Sample de-duplication across seams. Samples that lie on a control-mesh edge shared between two patches, or on a shared control vertex, must be emitted only once in the global vertex list. The implementation keys such samples on CGAL half-edge and vertex handles: `unordered_maps` (`m_ctrlMeshEdgesAlreadyTesselated`, `m_ctrlVerticesAlreadyTesselated`) record, on first encounter, the patch that owns a given edge or vertex and the global indices it produced. When an adjacent patch revisits the same edge or vertex, the stored indices are reused, and the duplicate foot points are skipped. The same mechanism is used in the medial-sphere calculation so that a foot point shared by several patches yields a single medial sample rather than one per incident patch, which would produce many redundant spheres.

4.4 Parallelisation

Parallelization with the gather/scatter approach. The three parallel stages of the pipeline are

1. growing the medial spheres (one independent problem per foot point),
2. adaptively sampling the patches (one independent problem per control face),
3. and evaluating the lfs at analysis vertices

They are all parallelized with the same approach: gather/scatter around a single `std::transform` with the `std::execution::par_unseq` policy. The work items are first gathered into a contiguous input vector, an equally sized output vector is pre-allocated, and `std::transform` maps a stateless lambda over them. Results are scattered back into the member containers afterwards. For example, sphere growing builds a vector of `(patchIndex, localVertexIndex)` pairs and transforms it into a vector of inner/outer `CalculationResultMedialPoint` pairs, while patch sampling transforms the index range $0, \ldots, K-1$ (filled with `std::iota`) into per-patch lists of accepted samples.

This gather-then-scatter structure exists to satisfy thread-safety constraints. The output slots are written independently and only merged into shared state single-threaded after the parallel region, so the lambdas never write to shared containers concurrently. The

one piece of shared mutable state that the parallel region does touch is the OpenSubdiv surface factory, which is not thread-safe by default because it caches surface topology internally. Rather than construct a separate factory per thread, the evaluator installs a thread-safe cache, `SurfaceFactoryCacheThreaded` parameterised with `std::mutex` and `std::lock_guard`, so that concurrent `InitVertexSurface` calls from different threads share one factory safely. Progress is reported from within the parallel region through a single `std::atomic<int>` counter that is incremented per work item and printed roughly once per percent, which is safe under concurrency and avoids serializing the workers. For debugging, the multithreading can be disabled with a flag (`m_useMultiThreading`), in which case the identical lambda is invoked in a sequential loop. Keeping the two paths behind one lambda guarantees that the parallel and serial runs compute the same result and simplifies debugging. On Linux, the parallel algorithms require the TBB backend of `libstdc++`.

The adaptive-sampling worklist. The recursive patch subdivision (see Ch. 3.4) is turned into an iterative computation. Each control face is processed by a lambda that maintains an explicit LIFO worklist, a `std::vector` of (triangle-in-UV, subdivision-depth) pairs seeded with the full parameter triangle at depth 0. The loop pops a sub-triangle, evaluates its three corner positions on the surface, and stops descending if the lifted triangle is smaller than the height threshold. Otherwise, it obtains the patch-enclosing sphere (see Ch. 3.4), queries the lfs at its center through the medial-axis oracle, and accepts the sub-triangle if the ε-condition (3.13) holds. If the condition fails, the three edge midpoints are computed in UV with `glm::dvec2` arithmetic, the four child sub-triangles are pushed with incremented depth, and the three new midpoint samples are recorded together with the depth at which they were introduced. Carrying the subdivision depth alongside each sample allows for outputting a color-encoded ε-sampled mesh that displays the recursion depth. Because each control face owns an independent worklist and writes only to its own result list, the entire adaptive sampling parallelizes with the same gather/scatter idiom as the other stages.

4.5 Numerical Details

Scale-adaptive tolerances. As introduced in Ch. 3.1, all spatial tolerances are made relative to the size of the model so that the pipeline behaves identically under uniform scaling. On construction, `LoopMedialPoints` computes the diameter D of the control-mesh AABB and its centre of gravity in a single pass over the vertices, and derives the mesh tolerance $\tau_{\text{mesh}} = D \cdot 10^{-m}$ from the user-supplied order of magnitude m. The helper `orderOfMagnitude` (a guarded $\lfloor \log_{10} |x| \rfloor$) is used where a data-dependent order of magnitude is needed and clamps values near zero to avoid a logarithm of zero. Iteration budgets are defined as constant member variables.

Sanitizing divergent outer spheres. The outer medial sphere at a foot point can legitimately be unbounded. For a locally convex region, the maximal outer empty

sphere degenerates to a half-space, so its center and radius tend to infinity. Left unhandled, such $\pm\infty$ or NaN values would corrupt the outer axis triangulation and every subsequent distance query. The outer spheres are therefore swept, and any non-finite component (detected with `std::isnan`/`std::isinf`) is replaced by a large finite sentinel (0.9 FLT_MAX). This pushes the offending medial triangles effectively to infinity without breaking the floating-point invariants of the AABB tree, so that a query point never finds them as its nearest outer medial primitive. Degenerate sub-triangles produced during adaptive sampling are handled analogously by discarding any triangle whose minimum height falls below τ_{mesh} (see Ch. 3.4).

Robust closest-point iterations. When the full Hessian is not positive definite, the Levenberg-Marquardt diagonal shift when the Gauss-Newton normal matrix is near-singular, and a steepest-descent step as a last resort, are implemented with EIGEN for the 2×2 and 3×2 systems, using its determinant and solve routines with the τ_{UV} thresholds as singularity tests. The optional Brent line search that guards against Newton overshoot is provided by Boost.Math's `brent_find_minima` restricted to $s \in [0, 1]$ with a modest bit budget, since the Newton direction is already good and only a step-length correction is needed.

CHAPTER 5

Results

In this chapter, we present results of our algorithm for several control meshes: Bunny [joT], Hand [ow], Mother [eot], Dog [pria], Fish [prib]. Some of the results were obtained in another research project [Ohr] that used the present work and are marked as such. The meshes have been converted to triangle meshes and reduced to 300-400 faces using quadric edge collapse decimation. The exact number of faces (see Table 5.1) was determined for each mesh empirically by visual inspection. All evaluations for illustrations ran on an AMD Ryzen 7 5800X 8-Core 4.85 GHz CPU and 64GB of memory [Ohr]. Only the script for the RMSE lfs plots ran on an AMD EPYC 9845 160-Core 2.1 GHz CPU to accelerate the very fine tessellation computations [Ohr]. Using these meshes, the convergence of our

Model	**Triangles**	**Vertices**	**Regular**	**Irregular**
Bunny	292	148	51	97
Hand	482	243	92	151
Mother	300	144	56	88
Dog	400	202	66	136
Fish	400	202	66	136

Table 5.1: Number of triangles and regular/irregular vertices for each of the models' control meshes.

model is analyzed, and our approach is compared to state-of-the-art approaches for MA approximation. The behavior of our solution around extraordinary vertices is analyzed using a very simple geometry, and the impact of Gregory patch approximation, which is used in our implementation, is examined. A few results for ε-samplings of the subdivision surfaces of the aforementioned meshes are also presented. The presented results not only confirm the advantages of our approach but also demonstrate its limitations, which are also discussed.

5.1 Convergence analysis

We need to assess the convergence behavior of our approximation of the lfs and of medial points to assess the precision of our approach. The convergence has to be assessed in two different dimensions:

1. Convergence of a medial point as the tolerance decreases
2. Convergence of the lfs as density of medial samples increases for a fixed tolerance

Both must be examined to assess the precision of the resulting ε-sampling.

5.1.1 Convergence of medial points

We can also analyze the convergence behavior of medial points for decreasing tolerance τ_{mesh}. For solutions m_1 and m_2 for a medial point obtained for $tol_1 < tol_2$ we expect $|m_1 - m_2|$ in the order of magnitude of tol_2 if the algorithm converges to the analytical solution for decreasing tolerance. Fig. 5.2 and 5.3 show that this is not always the case. Most likely, this can be attributed to large medial spheres where the second touch point lies close to the foot point. Near the foot point, the sphere is tangential to the surface such that the radius is very sensitive to errors in the closest point. This means that the radius can have an error much larger than the tolerance of the Newton scheme for the closest point search. This becomes clear from calculating the gradient of the radius w.r.t. the second touch point (here in 2D for easy visualization). We define p as the foot point which has the surface normal n and q as the second touch point on the surface:

$$R = \frac{\|\mathbf{q}-\mathbf{p}\|^2}{2\,|\mathbf{n}\cdot(\mathbf{q}-\mathbf{p})|}, \tag{5.1}$$

$$\nabla_{\mathbf{q}} R = \frac{\mathbf{q}-\mathbf{p}-R\,\mathbf{n}}{\mathbf{n}\cdot(\mathbf{q}-\mathbf{p})}. \tag{5.2}$$

As one can see in Fig. 5.1, the gradient becomes very large when q lies very close to p and near the surface tangent at p. This is driven by the denominator of the formula 5.2 for the gradient.

5.1.2 Convergence of local feature size

Since a closed-form computation of the MA for our subdivision surfaces is not possible, no ground truth is available for a proper convergence analysis. Therefore, we first construct the MA approximations $\hat{\mathcal{M}}(\mathcal{S})_\ell$ for many subsequent levels of tessellations $\ell \in \{4, 6, 8, 10, 12, 14, 16\}$. Then the lfs at tessellation vertices of level four is computed as the distance of a tessellation vertex to the closest point on a triangle of $\hat{\mathcal{M}}(\mathcal{S})_\ell$. For each $\hat{\mathcal{M}}(\mathcal{S})_\ell$ we thus obtain a set of lfs values. For each lfs value x_ℓ, we compute the errors

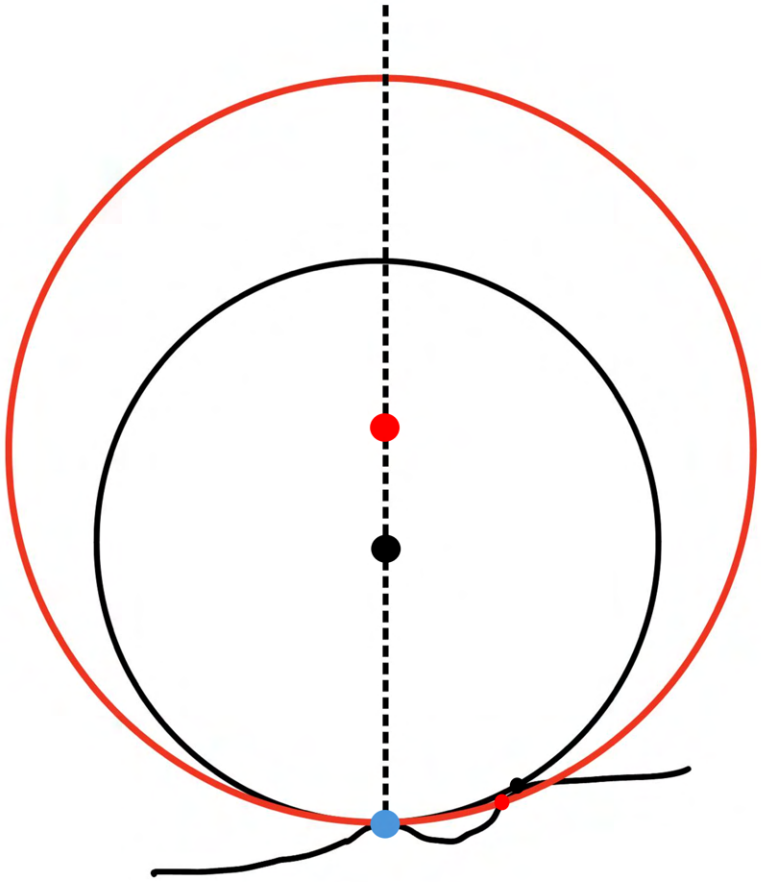

Figure 5.1: Two medial spheres tangential to the surface point p. A small error in the second touch point yields a much larger error in the radius.

$|x_\ell - x_{16}|/diameter$ and then calculate the RMSE for these errors. Figure 5.4 shows that the RMSE converges up to tessellation level $\ell = 8$, for all models, while above $\ell = 12$ no significant improvement can be observed (below 10^{-3} for most models). We therefore choose $\ell = 12$ to run evaluations in this chapter.

5.2 Comparison to state-of-the-art

Since no approaches for direct calculation of the MA of subdivision surfaces seem to have been published so far, we have to compare our approach to methods that approximate the MA of meshes or point clouds. Such methods can be applied to a tesselation of the subdivision surface, which allows a direct comparison to our method. The absence of ground truth for the MA of subdivision surfaces also makes the comparison of our approach to state-of-the-art approaches for MA approximation difficult. We can only

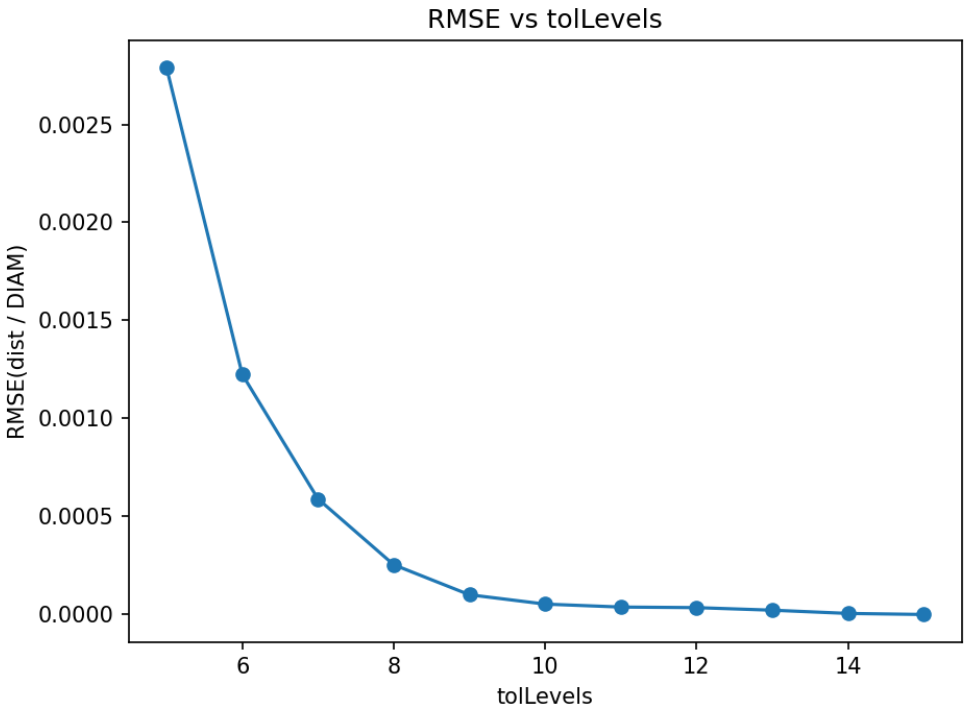

Figure 5.2: RMSE of the deviation of medial points at lifted control vertices for different tolerances $1E - x$ from their solution at tolerance $=1E - 15$ diameter (mesh: Bunny, $\ell' = 64$)

perform a visual comparison to find shortcomings and a quantitative comparison to confirm similarities. Using a torus as a very simple regular control mesh, we can find visual cues that support our claim of improved precision compared to state-of-the-art methods.

5.2.1 Visual comparison

We compare our MA with state-of-the-art approximations of the MA for meshes: PowerCrust [ACK01], VoxelCore [YLJ18], and Q-MAT [LWS+15]. Tessellations of the subdivision surface served as input for these implementations. Since these algorithms are rooted in the Voronoi Diagram of the mesh vertices or a voxel representation of the volume, they produce an approximation of the MA with unwanted branches that need to be pruned. Therefore,

- 'simplify' has been applied to the PowerCrust MA approximation,
- the pruning parameter $\lambda = 0.1$ has been set for VoxelCore (grid size 512),
- the number of vertices from Q-MAT has been reduced to 10%,

to ensure comparability of results [Ohr]. Figure 5.5 shows the MA for all models. We see that the VoxelCore result fails to capture large parts of the MA and those parts it

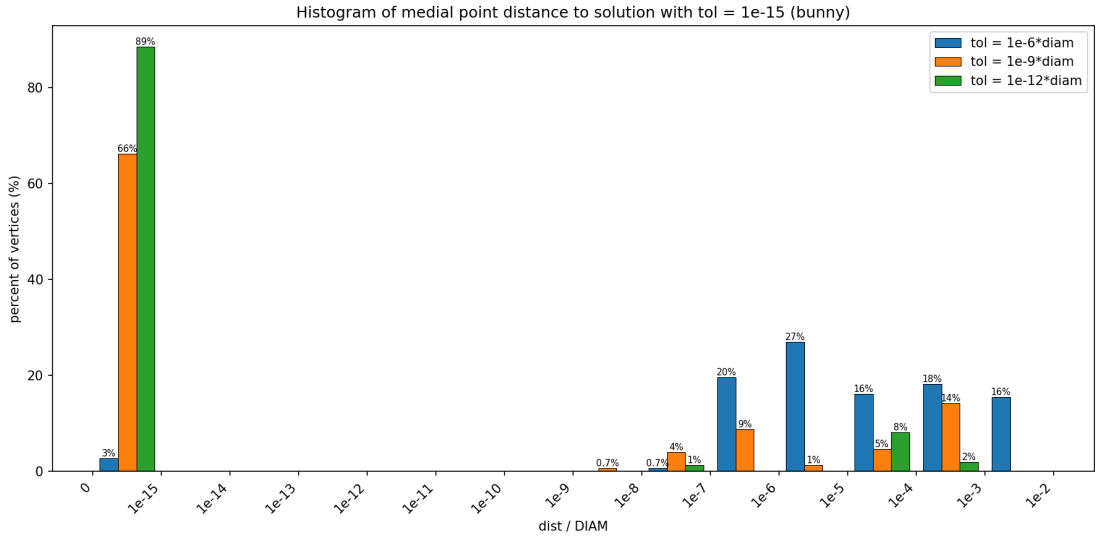

Figure 5.3: Histogram of the deviation of medial points at lifted control vertices for different tolerances from their solution at tolerance $= 1E-15$ diameter (mesh: Bunny, $\ell' = 64$)

captures lack detail. This can mainly be observed on tubular structures such as fingers, legs, and arms in the meshes Hand, Mother and Dog. For Fish it even fails to capture fins which occupy much larger regions. Q-MAT performs much better but fails to capture finer details which can best be observed on the fins of Fish, where also some small holes are visible. These problems can, to some degree, be dealt with by adapting the pruning parameters. This, however, would be an empirical setup that requires a lot of fine-tuning for each mesh and highlights an advantage of our approach: The user only needs to control the required resolution with tessellation levels ℓ, ℓ' where higher tessellation levels automatically yield better results. PowerCrust is the state-of-the-art model that performs best, capturing even fine details.

Figure 5.6 compares how the different algorithms perform at several tessellation levels. The renderings show that our approach requires a much lower tessellation level to capture all essential details of the MA than the other solutions. The visual quality of our solution at tessellation level $\ell = 2$ is only reached by PowerCrust at tessellation levels 8 or 12.

Figure 5.7 compares the boundary of the MA for our algorithm with PowerCrust at tessellation levels $\ell = 4$ and $\ell = 12$. The comparison shows that our approach captures the smooth boundary of the MA already at low tessellation levels much better than PowerCrust. Only at level 12 does PowerCrust approach the boundary that our solution already captures at tessellation level 4. This again shows that our approach has the advantage that the user does not need to spend much time on adapting parameters for obtaining a good approximation of the MA since all essential features are already captured even at low tessellation levels.

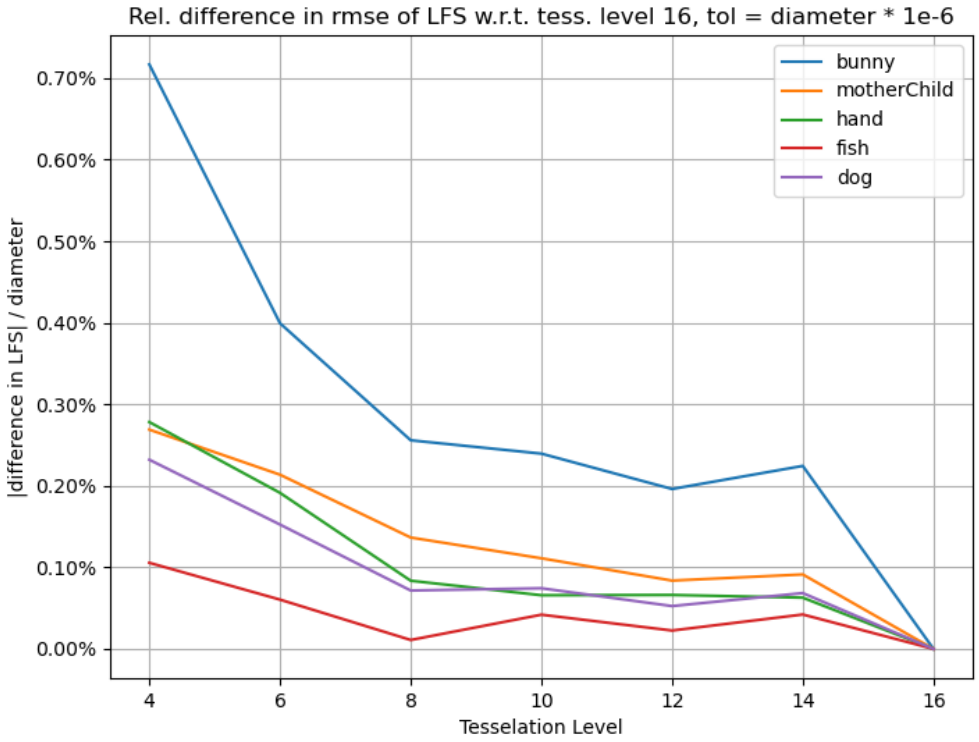

Figure 5.4: RMSE lfs accuracy for different models converges for increasing MA levels (tolerance = 10^{-6}).

5.2.2 Quantitative comparison

Algorithm	$\ell = 2$	$\ell = 4$	$\ell = 8$	$\ell = 12$
PowerCrust	0.0112	0.0055	0.0037	0.0031
Q-MAT	0.0174	0.0101	0.0078	0.0068
Ours	0.0039	0.0022	0.0008	0.0005

Table 5.2: RMSE between other MAs and Ours with $\ell = 12$. The value for Ours with $\ell = 12$ shows pure comparison error. Data source: private communication [Ohr]

Table 5.2 [Ohr] shows the error of the MA of different algorithms at several tessellation levels compared to our solution at tessellation level $\ell = 12$. The MA triangulations were sampled with 10M points each, then their Chamfer distances were computed. The entry Ours with $\ell = 12$ is the comparison error for reference. We see from this table that the errors of our algorithm and the other two algorithms converge. This is another indicator of convergence of our solution where no ground truth is available.

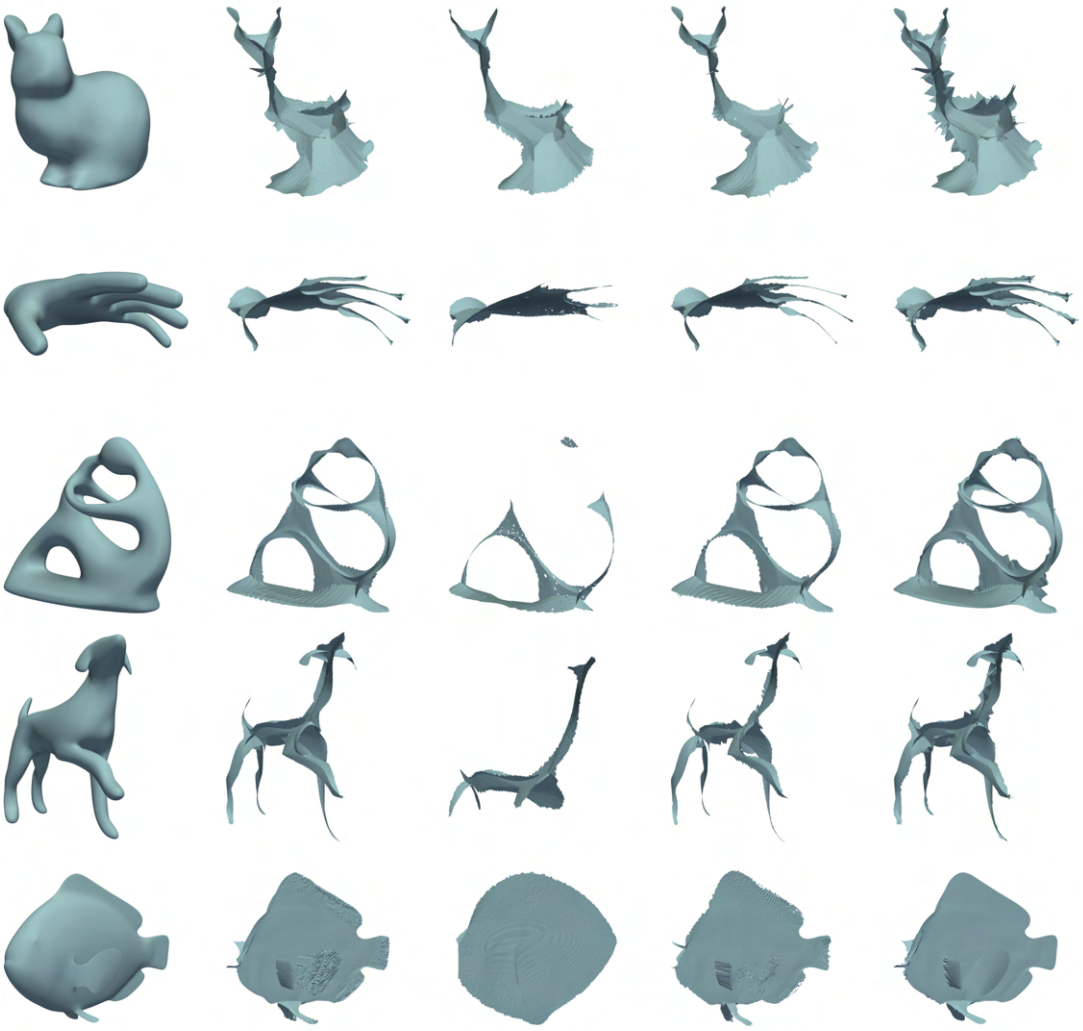

Figure 5.5: Medial axis for all models. Left to right: Loop subdivision surface tessellated at level 12, PowerCrust, VoxelCore, Q-MAT, Ours. Source: Private communication [Ohr]

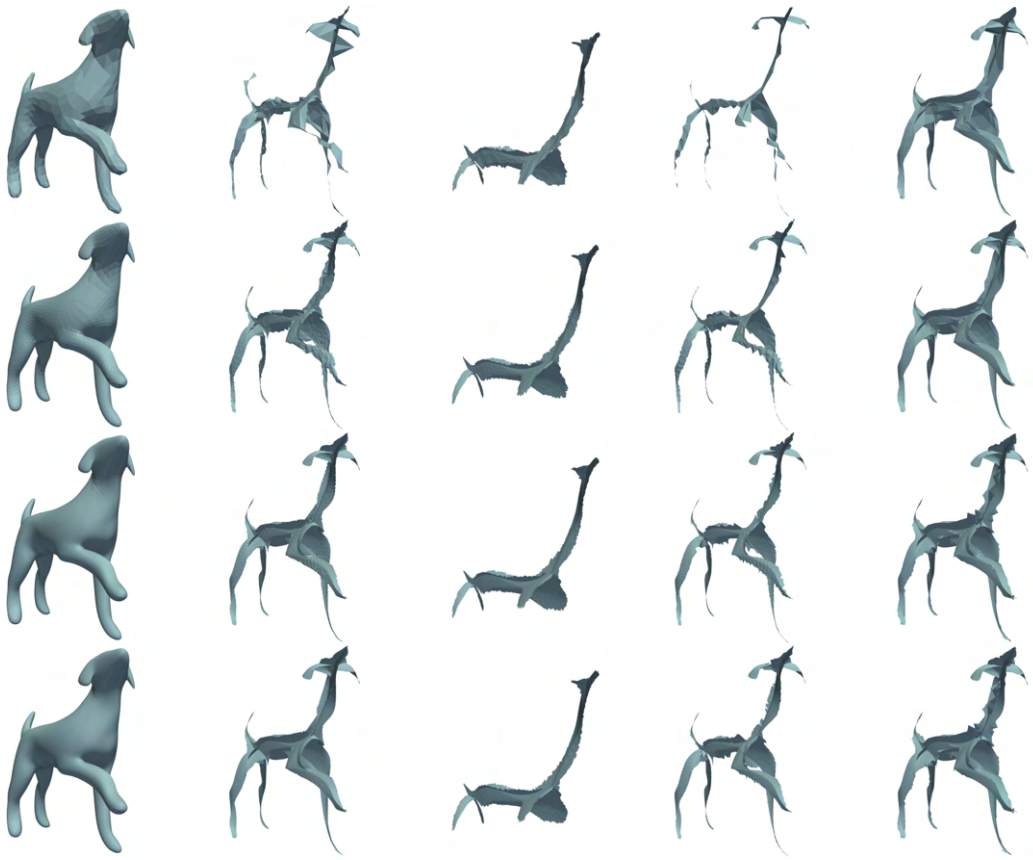

Figure 5.6: Medial axis for tessellation levels 2, 4, 8, and 12 (default). Left to right: Loop subdivision control mesh, PowerCrust, VoxelCore, Q-MAT, and Ours. Image source: private communication [Ohr]

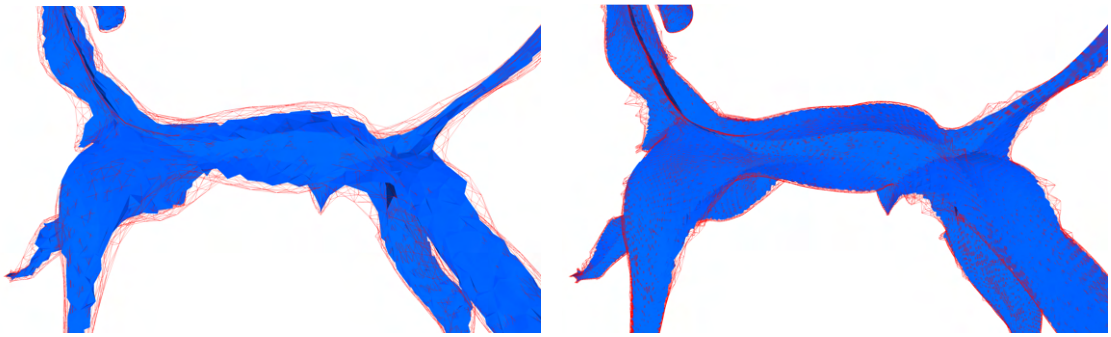

Figure 5.7: Medial axis analysis of Ours (red edges) and PowerCrust (blue faces). Left: $\ell = 4$, right: $\ell = 12$. PowerCrust's medial axis expands with ℓ but only approaches ours at $\ell = 12$.

5.2.3 Approximating a torus

Although in general there is no closed-form solution for the MA, some special geometries exist that are simple enough to deduce the shape of the MA. A sphere, torus, cuboid, or ellipsoid are examples of such simple geometries. For Loop subdivision surfaces, the situation is even more complicated. Since these surfaces depend on a triangulation, their shape cannot be deduced even for control meshes that stem from sampling a simple geometry as described above. In reverse, given a simple surface, it is not possible to determine a control mesh such that the surface is the subdivision surface of this mesh. Therefore, the only option is to use the subdivision surface of a very fine triangulation of a simple shape for which the MA is known. This subdivision surface will be very close to the original mesh, and thus its MA will be close to the known MA of the original shape. Here, the somewhat vague expression "close" refers to the fact that our triangulations of a smooth shape converge to the shape at least in the Hausdorff sense as the triangle size goes to zero (the longest edge goes to zero and the triangle shape is the same everywhere). From the convex-hull property of Loop subdivision surfaces follows that the subdivision surface converges to the original shape in the same sense if the subdivision surface is C^2 (no extraordinary vertices in the control mesh). Thus, comparing an approximation of the MA of the subdivision surface to the MA of the original shape provides useful information on how well the approximation works. This approach is as close to a ground truth comparison as we can get. We chose a torus as the original shape. The inner MA of a torus is a circle. The torus has the advantage that it can be triangulated regularly, i.e., every vertex has valence six. Thus the subdivision surface is a smooth C^2 surface. As mentioned previously, this is important for the convergence of the MA, see also Ch. 5.3. We chose the torus to have main radius R=3 and tubular radius r=1 using the following parametrization:

$$X(u, v) = ((R + r\cos(v))\cos(u), (R + r\cos(v))\sin(u), r\sin(v)).$$

A regular triangulation Torus128x64 is defined by using 128 equidistant u-samples and 64 equidistant v-samples, resulting in 16384 faces. Fig. 5.8 shows a rendering of Torus128x64. Fig. 5.9 shows a plot of the distance of 1024 sample points on the MA of Torus128x64 in the x-z-plane (tubular cross-section) to the original torus. The plot reveals that the subdivision surface of our torus triangulation has a tubular cross-section that is slightly egg-shaped. This implies that the corresponding tubular cross-section of the MA of the subdivision surface must have the shape of a short, almost straight line. The plot also shows that the subdivision surface lies completely inside the original torus. In fact, the subdivision surface of the triangulation approach we use for the torus is always a deformed and slightly concentric shrunk torus. The reason for that is the non-uniform triangle size, with smaller triangles near the center of the torus and much larger triangles further outside. The coarser this triangulation type gets, the more pronounced the shrinking aspect becomes. Fig. 5.10 shows for a coarse triangulation (128 faces) this shrinking effect clearly. This implies that the MA of the subdivision surface of Torus128x64 also lies closer to the center of the torus than the MA (circle) of the original torus. Fig. 5.11 shows the tubular cross-section in the x-z-plane. It

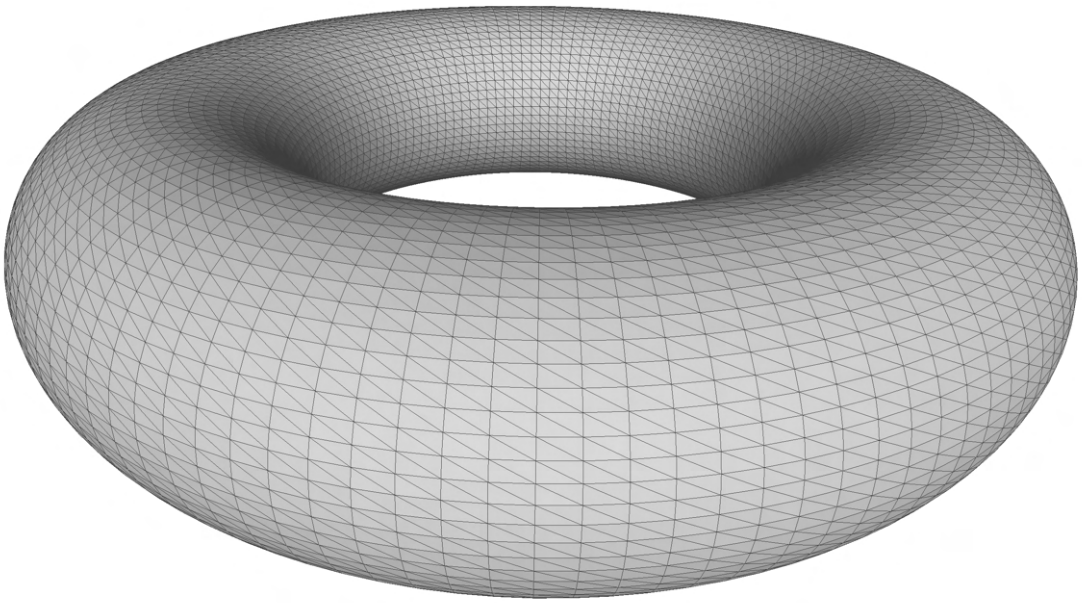

Figure 5.8: Torus128x64 control mesh with 128 u-samples and 64 v-samples (16384 faces) of the torus R=3, r=1.

depicts the cross-section of our MA, which clearly has the aforementioned shape of a short line slightly shifted away from the MA of the torus towards the center of the torus. It also shows the cross-section of the PowerCrust MA, which does not have the expected shape of a line but approximates this shape. For comparison, a simple solution Sampling2D [OMW16] for a 2D MA of the cross-section of the subdivision surface of Torus128x64 is plotted. We obtain this 2D MA by modifying an algorithm for 2D curves [OMW16] such that it takes the 1024 cross-sectional samples of the subdivision surfaces and calculates medial points as the centers of inscribed circles of the point cloud of samples. (For this purpose, neighboring points of this point cloud are connected by line segments, transforming the cross-sectional point cloud into a piecewise linear 2d curve.) This 2D approach yields a much worse result, though. The cross-sectional comparison shows that our solution is much more precise than the other approaches.

5.3 Extraordinary vertices and Gregory patches

Extraordinary vertices pose a problem for Loop subdivision surfaces not only because smoothness drops to C^1 at these points but also because evaluation becomes considerably more complicated even with Stam's limit approach [Sta98] since this approach requires infinitely many patch functions to evaluate a whole patch with an extraordinary vertex. One approach to circumvent this is to use end caps with Gregory patches [Gre74] as implemented in OpenSubdiv [Pixb] based on an idea by Loop et al. [LSNCn09]. This

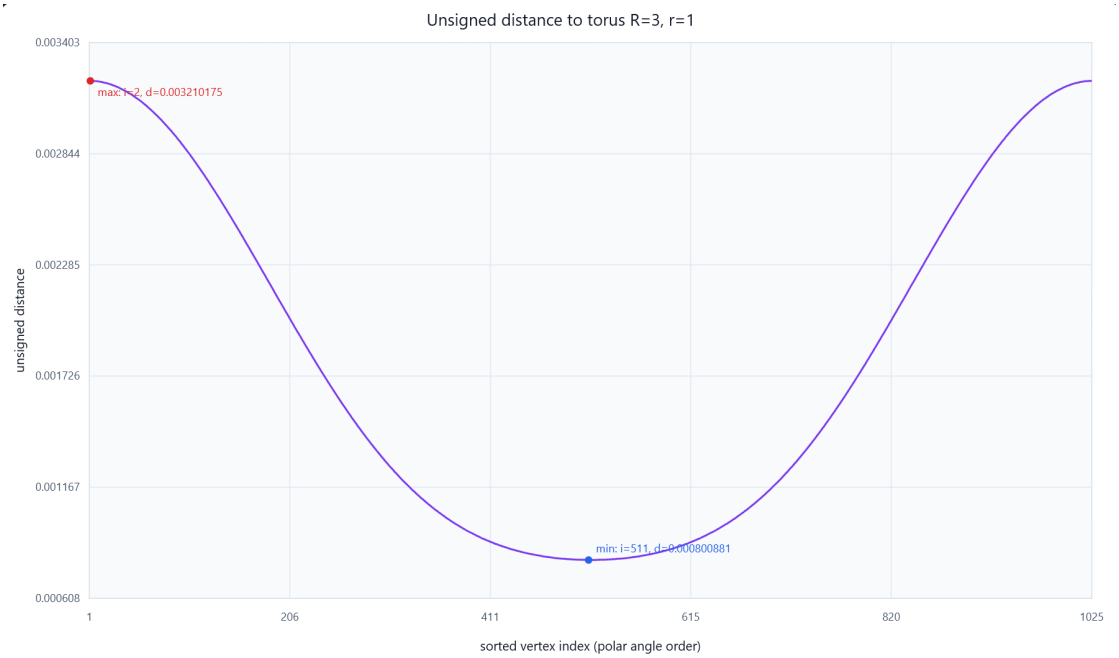

Figure 5.9: Distances of 1024 samples points on the MA of Torus128x64 in the x-z-plane to the original torus.

approach in OpenSubdiv subdivides a patch a few times (OpenSubdiv default: 6) before approximating the remaining (triangular) neighbourhood of an extraordinary vertex with a Gregory patch. At its boundary, this Gregory patch connects with G1 continuity to the neighbouring patch, be it another Gregory patch or a box spline [LSNCn09]. (The OpenSubdiv documentation does not state this directly, but only explains that "Gregory patches are used to approximate irregular areas" [Pixb].) The Gregory patches are quartic rational patches that have quartic polynomial boundary curves. Since our implementation uses OpenSubdiv, it approximates the subdivision surface with these small Gregory patch modifications. On many MAs in our results, we see areas on the MA that look like spikes. These formations stem from extraordinary vertices and their Gregory Patch approximation. Fig. 5.12 compares a Gregory patch fitted to approximate a subdivision surface patch in OpenSubdiv. Both surfaces match at the vertices. Fig. 5.13 shows a patch in OpenSubdiv with end cap subdivision level one and ten, the latter visualizing the box spline except in extremely small neighborhoods of the extraordinary vertices that are visually not perceivable where the subdivision surface is approximated by a Gregory patch. The rendering shows how the level-one-Gregory-end-caps are applied near two extraordinary vertices, where they are perceived as triangular areas each taking up about a fourth of the patch that deviate slightly from the subdivision surface. At higher end cap levels, these caps become much smaller.

We study the influence of extraordinary vertices and their Gregory end caps on a

Figure 5.10: A fine triangulation of the torus (fine black mesh 1M faces), a coarse triangulation of the torus (thick black mesh, 8 u- and 8 v-samples) and the Loop subdivision surface of the coarse triangulation (light grey faces).

triangulated cuboid (Cuboid) with 6 extraordinary vertices and its subdivision surface. The subdivision level for the Gregory end cap in OpenSubdiv is limited, most likely due to memory constraints. In our setup, the limit is 10. The size of the Gregory end caps can be further reduced by Loop-subdividing the Cuboid and then using it as a new control mesh Cuboid2Loop. This process yields the same Loop subdivision surface, but the triangles around extraordinary vertices become much smaller, which allows us to study the influence of the Gregory end caps. The two control meshes and the subdivision surface they yield are shown in Fig. 5.14. Fig.5.15 compares MAs of the subdivision surface of Cuboid for very large and very small end caps. The large end caps occupy one fourth of the triangular domain of a patch while the small end caps occupy less than 0.025%. The renderings show that the Gregory end caps decrease the lfs, causing the clearly visible "flaps" near those end caps. For the very small end caps, these "flaps" become spikes, which shows that the decrease in lfs is strongly influenced by the size of these end caps, but it is not clear what these regions would look like for the actual subdivision surface without end patch approximation. This means that at this point it remains unclear what influence extraordinary vertices have on the lfs.

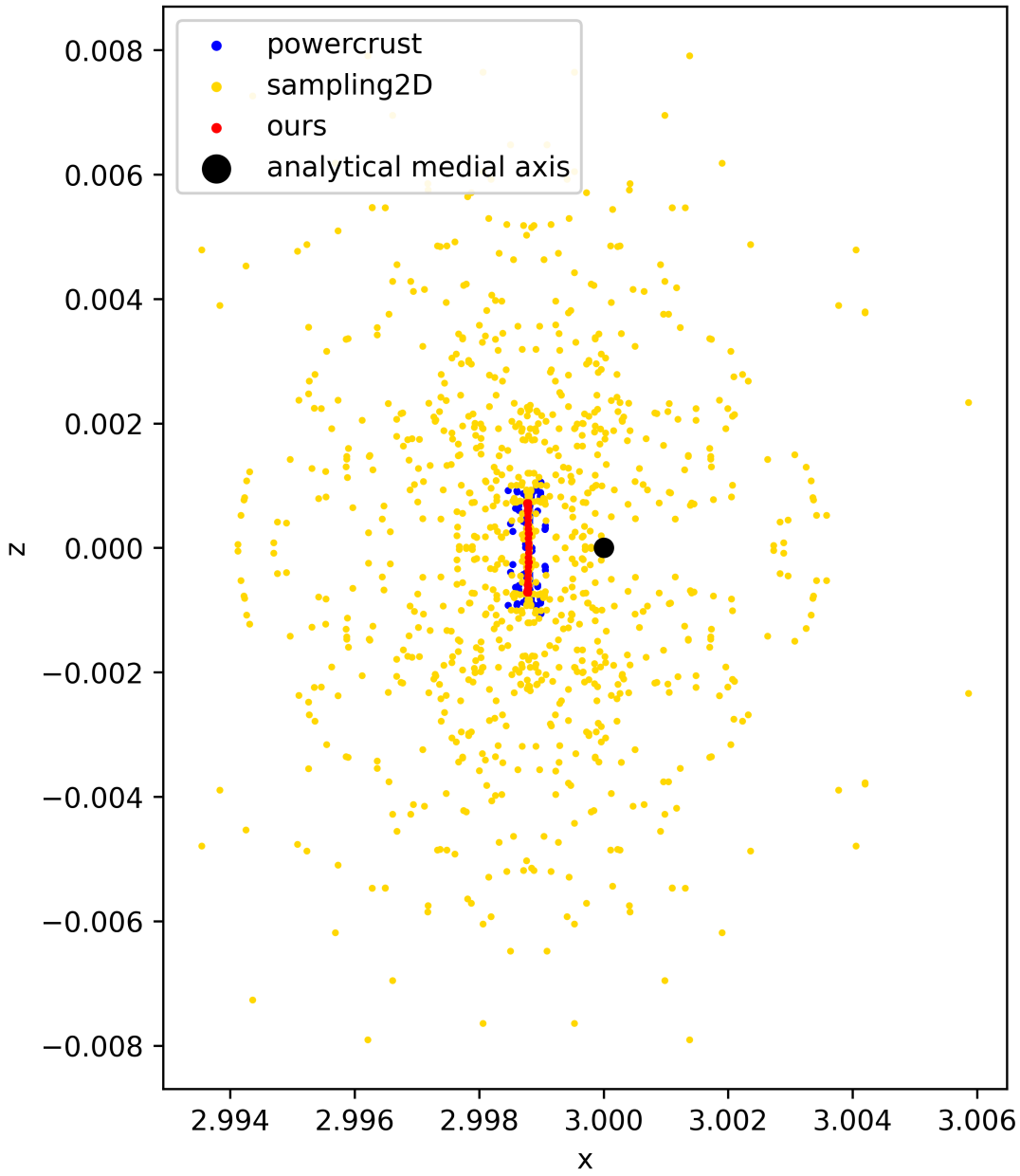

Figure 5.11: A zoomed-in tubular cross-section of Ours, PowerCrust, and Sampling2D [OMW16] MAs of the deformed circle limit curve, along with the analytical medial axis point of the real torus.

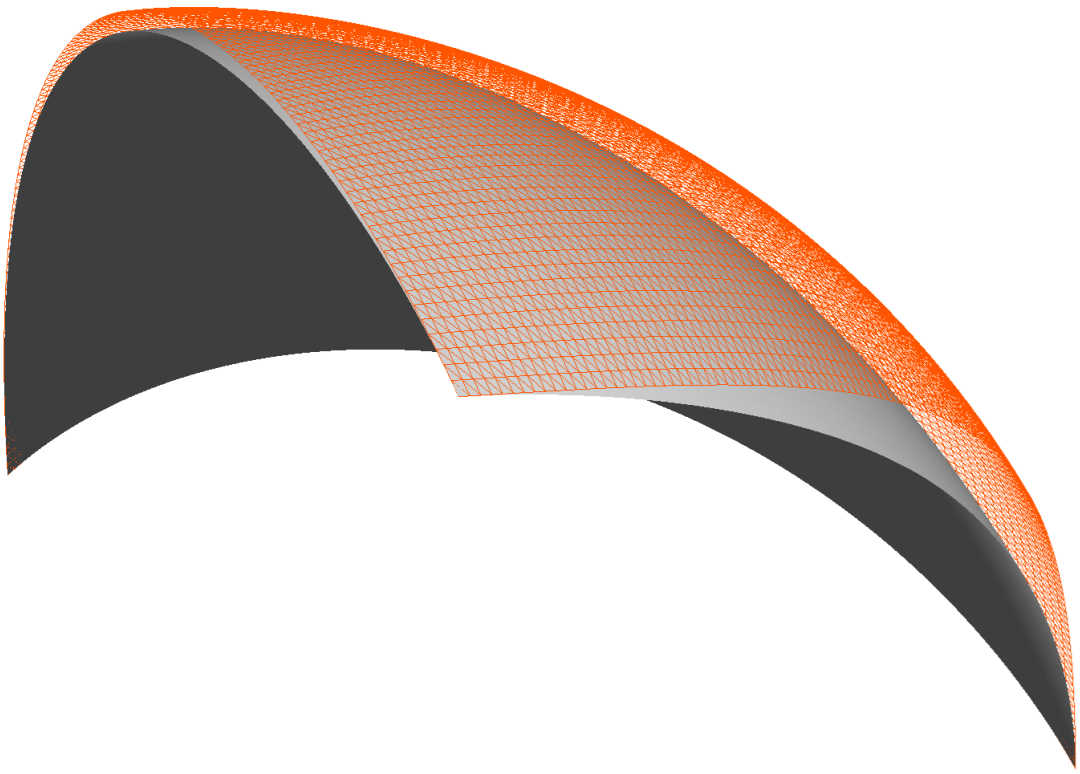

Figure 5.12: A Gregory patch (red) and the corresponding box spline (grey) for a patch with two extraordinary vertices as used in OpenSubdiv. (The box spline is rendered using end cap subdivision level 10.)

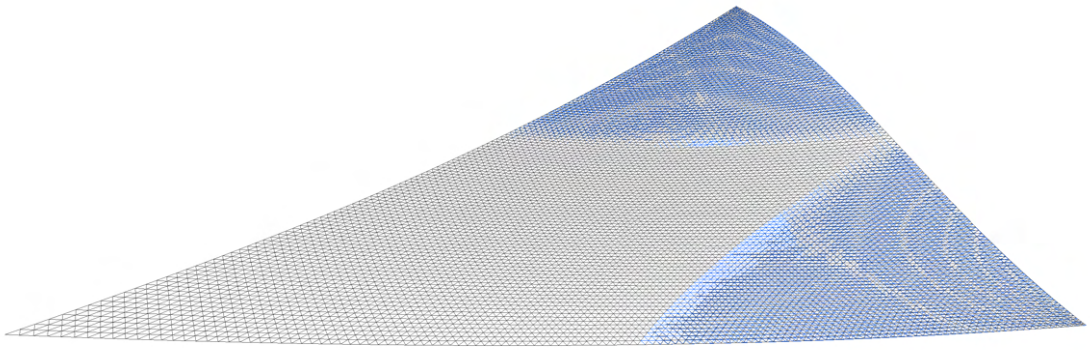

Figure 5.13: Rendering of a patch with two extraordinary vertices (right side of image) with end cap subdivision level 10 (grey) and level 1 (blue) as used in OpenSubdiv.

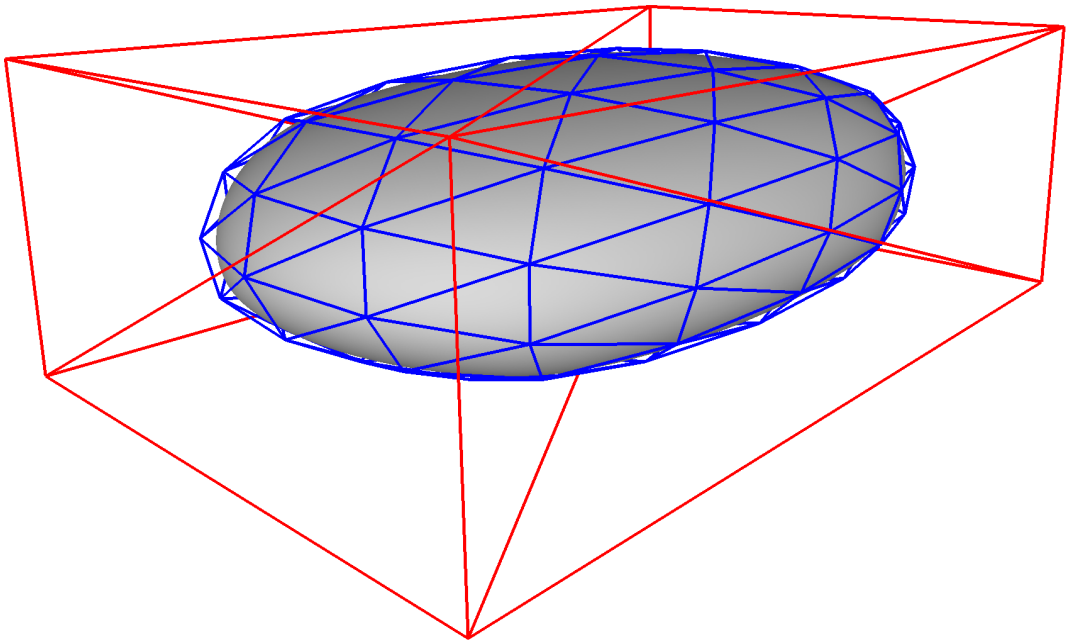

Figure 5.14: Red: Triangulation of a cuboid (CUBOID), blue: Level 2 Loop subdivision of (CUBOID2LOOP), grey: Loop subdivision surface of CUBOID and CUBOID2LOOP.

5.4 ε-sampling examples

Fig. 5.16 [Ohr] shows results for our ε-sampling approach with several values for ε. The renderings show that the samplings adapt to lfs with higher recursion depth for tessellation in areas that are narrow or have high curvature.

5.5 Meshing the medial axis

Our approach collapses the surface tesselation mesh into the medial axis, which leads to a self-intersecting triangle soup as can be seen in Fig. 5.17. For our purpose of lfs calculation, this does not pose a problem since we only perform distance calculation. However, for several reasons it is desirable to have a clean triangle mesh. So while our triangulation approach, being simple and fast, suffices, other approaches were also tested but did not produce satisfying results. These approaches are based on Delaunay triangulations (or tetrahedralizations) and have only been tested for a triangulation of the inner medial points.

5.5.1 Conforming constrained Delaunay triangulation

The first alternative approach is a conforming constrained Delaunay triangulation. The points to be triangulated are the inner medial points and the corresponding tesselation

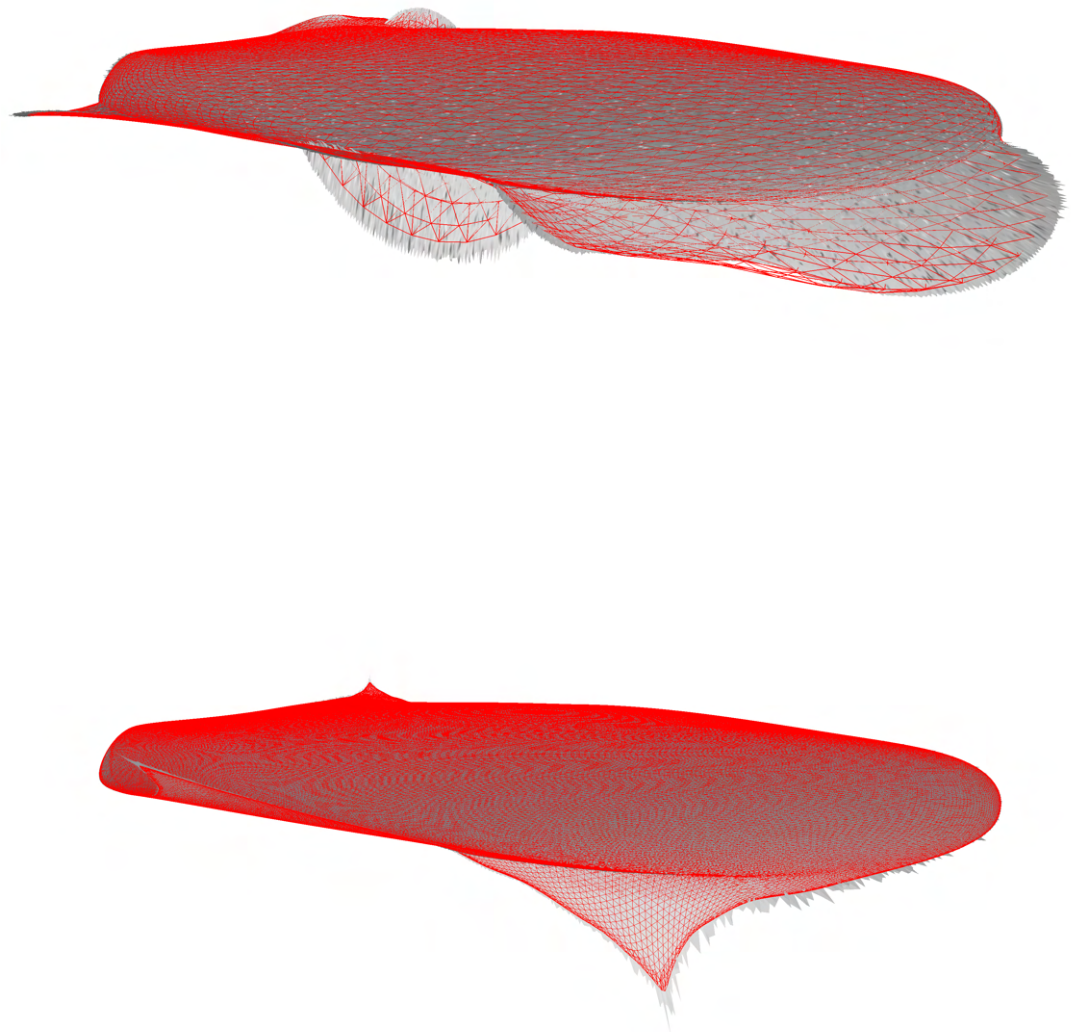

Figure 5.15: Top: MA of Cuboid with level 1 Gregory end caps. Our MA in red was obtained with $\ell = 32, \ell' = 64$; PowerCrust used a subdivision surface tessellation at level 256. Bottom: MA of Cuboid2Loop with level 10 Gregory end caps. Our MA in red was obtained with $\ell = 32, \ell' = 32$; PowerCrust used a subdivision surface tessellation at level 32.

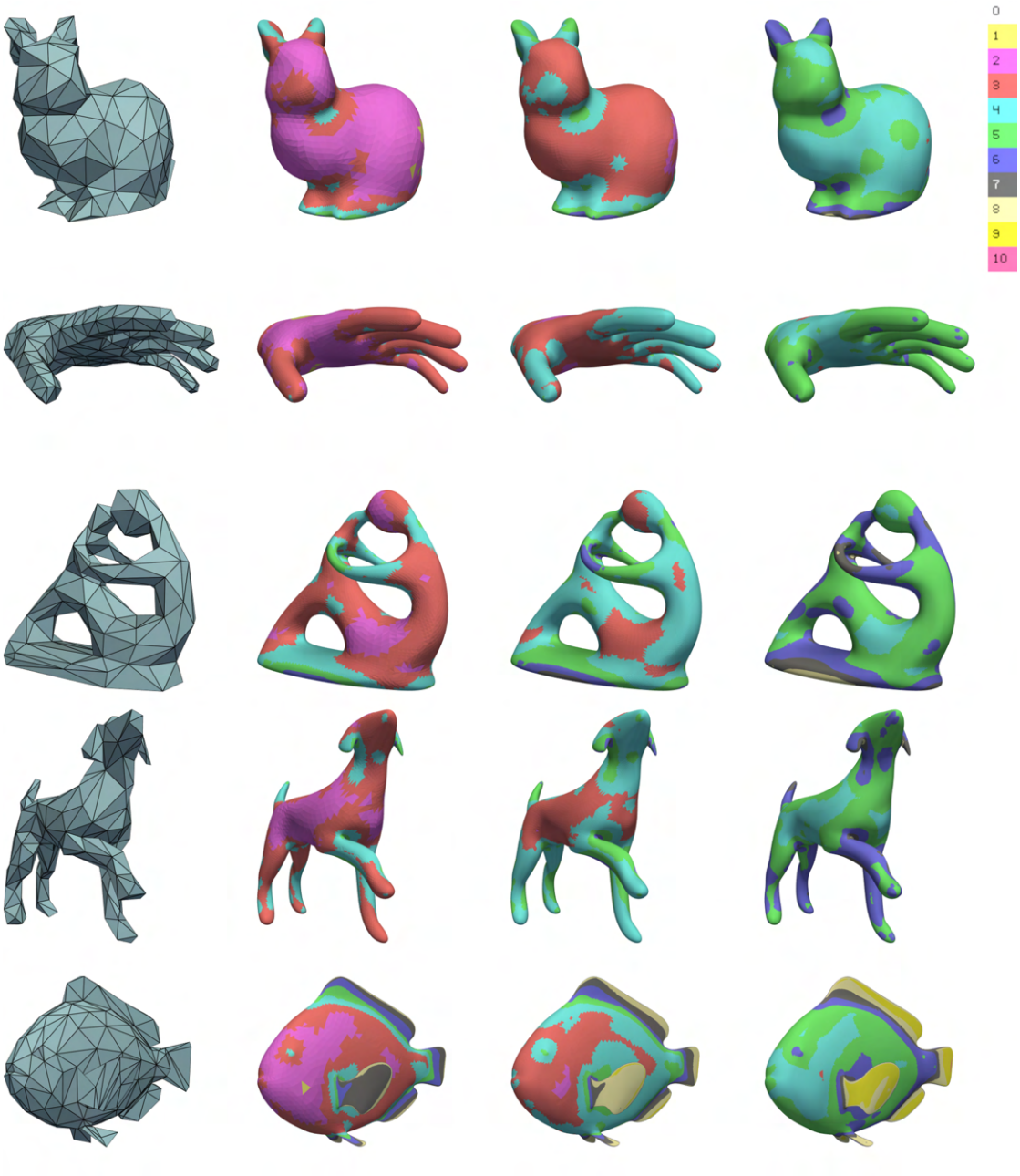

Figure 5.16: Control mesh and triangulations of ε-samplings of its Loop subdivision surfaces. (number of tessellation recursions according to legend in top right) for $\varepsilon = 1, 0.5, 0.2$: Bunny, Hand, Mother, Dog, Fish. Image source: private communication [Ohr].

points of the subdivision surface. The tesselation triangles of the subdivision surface are the constraint triangles. Applying the triangulation fills the volume enclosed by the subdivision surface with tetrahedra that have the inner medial points and surface tesselation points as vertices. We only keep triangles that have inner medial points as vertices. Then we go through all tetrahedra that only have inner medial points as vertices and remove the largest triangle face (largest circumference). The last step is necessary since the inner MA cannot be the boundary of a closed volume. In the second row of Fig. 5.18 one can see that this method creates many triangles that connect medial sheets that should not be connected and thus produces a wrong topology.

5.5.2 Regular Delaunay triangulation

The second alternative approach is a regular triangulation with a power diagram: The inner and outer medial points are weighted with their corresponding squared medial radius, and a regular triangulation of these weighted points is created. This yields a tetrahedralization of the inner and outer medial points. Then only those triangles are kept that have inner medial points as vertices. This approach separates the medial sheets well, as can be seen in the bottom row of Fig. 5.18. This approach still produces (rather flat) tetrahedra inside sheets, but these could be removed with the same approach as in the conforming constrained Delaunay triangulation. However, the boundary of the inner medial axis (red edges in Fig. 5.17) is very jagged in this triangulation method, which makes it less suitable for the purpose of lfs calculation than inheriting the topology of the subdivision surface tesselation used for medial sphere shrinking.

5.6 Limitations

Finding closest points through our Newton scheme with sufficient precision requires longer computation times the finer the tessellations get. Badly designed control meshes, e.g., with tightly folded concavities, could lead to self-intersecting subdivision surfaces, which we do not handle.

Furthermore, our sphere fitting approach can lead to spheres that are too large if the tessellation level for finding start values is not high enough (local minima) or if the mesh is badly conditioned for the Newton scheme, e.g., near focal points with low curvature.

Since our implementation with OpenSubdiv always uses Gregory end caps, we do not obtain the MA of the actual subdivision surface.

Our ε-sampling method causes extremely long runtimes near sharp features. These are often present in artificial shapes and industrial applications. This can be seen on the Fish model (see Fig. 5.5), where sharp edges in the control mesh occur at the tail fin. This causes very high recursion levels and therefore a long runtime.

Another problem with our ε-sampling method is that it is far from being minimal. Our subdivision condition (3.13) is only a rough upper bound that ensures the ε-sampling and

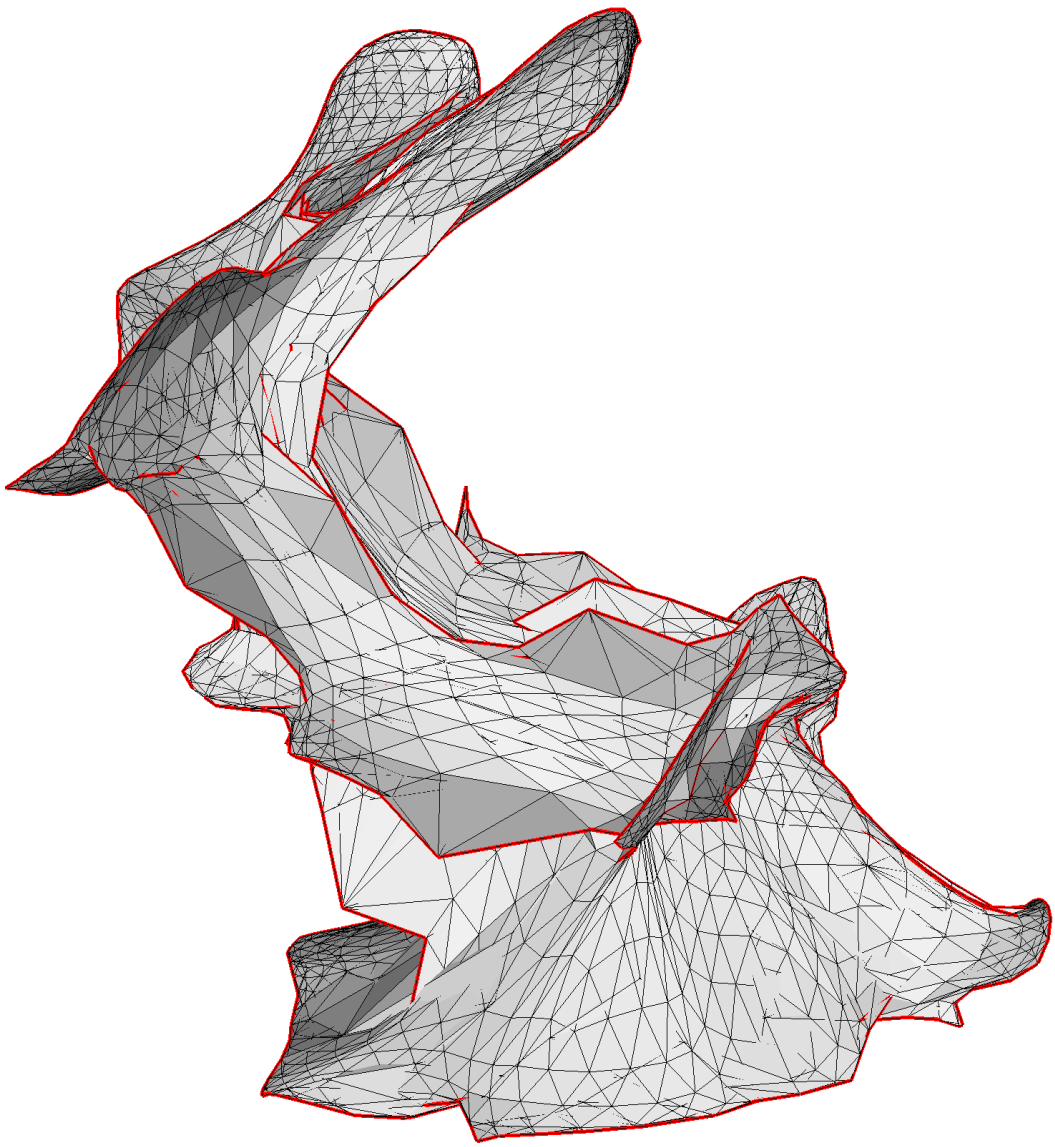

Figure 5.17: The mesh of our medial axis with border edges marked in red. In several areas, especially near borders, self-intersecting faces are clearly visible.

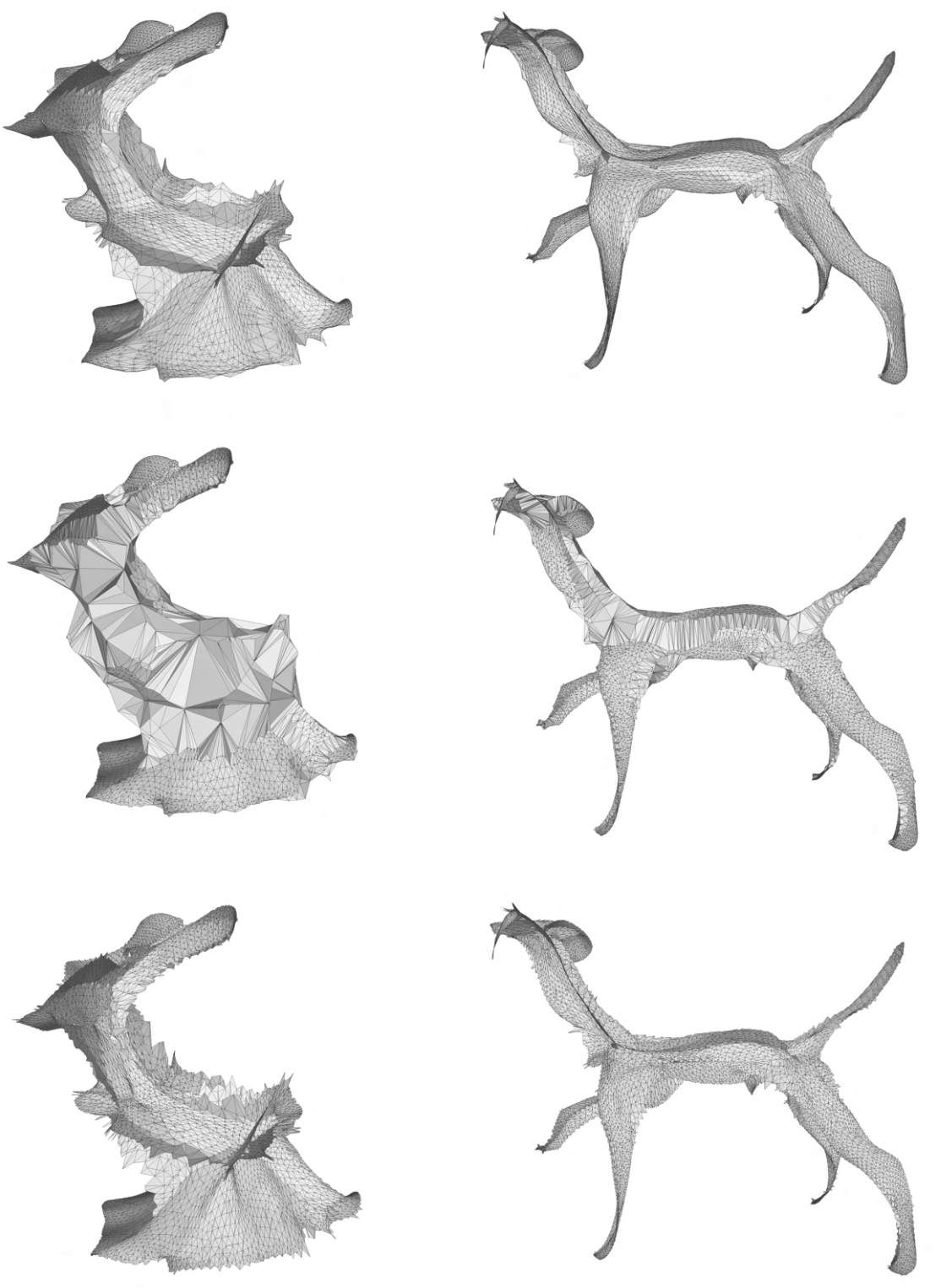

Figure 5.18: Top to bottom: MA triangulations with inherited topology from subdivision surface tesselation, a conforming constrained Delaunay triangulation, and power diagram weighted Delaunay triangulation. For both models $\ell = 8$

is easy to check. Even with a subdivision condition that ensures a minimal ε-sampling, our approach still only operates locally per patch. A minimal ε-sampling per patch does not imply a globally minimal ε-sampling of the surfaces consisting of the minimally ε-sampled patches.

Our MA is a triangle soup, which suffices for our purpose of lfs calculation. Since our approach utilizes the topology of the tesselation of the subdivision surface, we could also collapse the tesselation mesh into our MA. This would yield a mesh of the MA that inherits its topology from the tesselation. This manifold mesh would cover every sheet of the MA from two sides, which is not desirable. The MA can have sheets and branches and thus is non-manifold. So our mesh would not be topologically correct and can self-intersect since each sheet is triangulated from two sides. In applications of the MA, such as shape segmentation and curve skeletons that require correct topological structure, our MA cannot be used.

CHAPTER 6

Applications

The canonical application of our method lies in the computation of the MA of triangle meshes obtained from point clouds. Such point clouds can, for example, come from a 3D scan and thus our approach can provide a MA of such a point cloud: A subdivision surface can be fitted to the point cloud with existing methods [MK05] and then our MA is calculated for the fitted subdivision surface. The MA of the fitted subdivision surface will then also be very close to the MA of the originally scanned surface. Using this fitting approach has the inherent advantage that our MA is not sensitive to noise in the point cloud. Therefore, our method can, for example, be used for 3D object classification methods with machine learning that rely on the MA of the objects such as the recent publication [HWQ+19] where the authors note that many of the ModelNet40 [WSK+15] shapes they used had to be repaired before they could apply Q-MAT, which we saw performs worse than our approach. This application is also an advantage that our method has over methods [FHA24] that estimate the lfs directly without calculating or approximating the MA. In addition, our approach has been shown to have several useful applications [Ohr] for re-meshing subdivision surfaces and evaluating surface reconstruction algorithms, which we will shortly outline here.

6.1 Precomputing feature-adaptive subdivision surfaces tessellations

Although subdivision surfaces are widely used, real-time applications such as computer games still tend to rely on polygonal meshes for representing geometry. This is largely because current GPUs are optimized for polygon rendering and handle it efficiently. In many production workflows, however, subdivision surfaces are already used during asset creation, e.g., for cut scenes. OpenSubdiv has been specifically designed to render these subdivision surfaces efficiently on the GPU with custom tessellation levels per patch. Our approach enables two useful applications in this context:

1. If the graphics pipeline requires a polygon mesh, we can provide a triangulation that uses a limited triangle budget efficiently for modeling a subdivision surface asset with as much detail as possible.

2. If the graphics pipeline uses OpenSubdiv for efficient rendering of subdivision surfaces, our ε-sampling allows us to derive a tessellation depth or pattern for every patch that matches the maximal or average lfs in the patch. This approach yields better lfs awareness the smaller the patches are. The patch size must also be taken into account in cases where it varies significantly between patches.

6.2 Remeshing subdivision surfaces based on local feature size

Another approach that yields a lfs-aware remeshing of a subdivision surface is Delaunay-based remeshing. It ensures the Delaunay property, providing better aspect ratios for triangles than our method. An implementation [Ohr] demonstrates that 35-75x fewer triangles for our models were required than with our ε-sampling approach. Fig. 6.1 [Ohr] shows renderings of various Delaunay-based remeshings with much better triangle quality than our approach, which inherits the triangle shapes from the control mesh, but optimal control triangle shape for patch shaping does not necessarily provide good aspect ratios.

6.3 Unbiased evaluation of surface reconstruction algorithms

It has been shown [Ohr] that our approach can be used for evaluating surface reconstruction algorithms on how they perform in feature-adaptive sampling. This evaluation method proposes to use a Loop subdivision surface as an unbiased reference rather than a manually created or otherwise constructed mesh e.g. from a 3D scan. It samples each mesh very densely and evaluates the one-sided Hausdorff distance for each sample point to the subdivision surface. This allows us to compare a meshing method to our epsilon sampled triangulation of the subdivision surface.

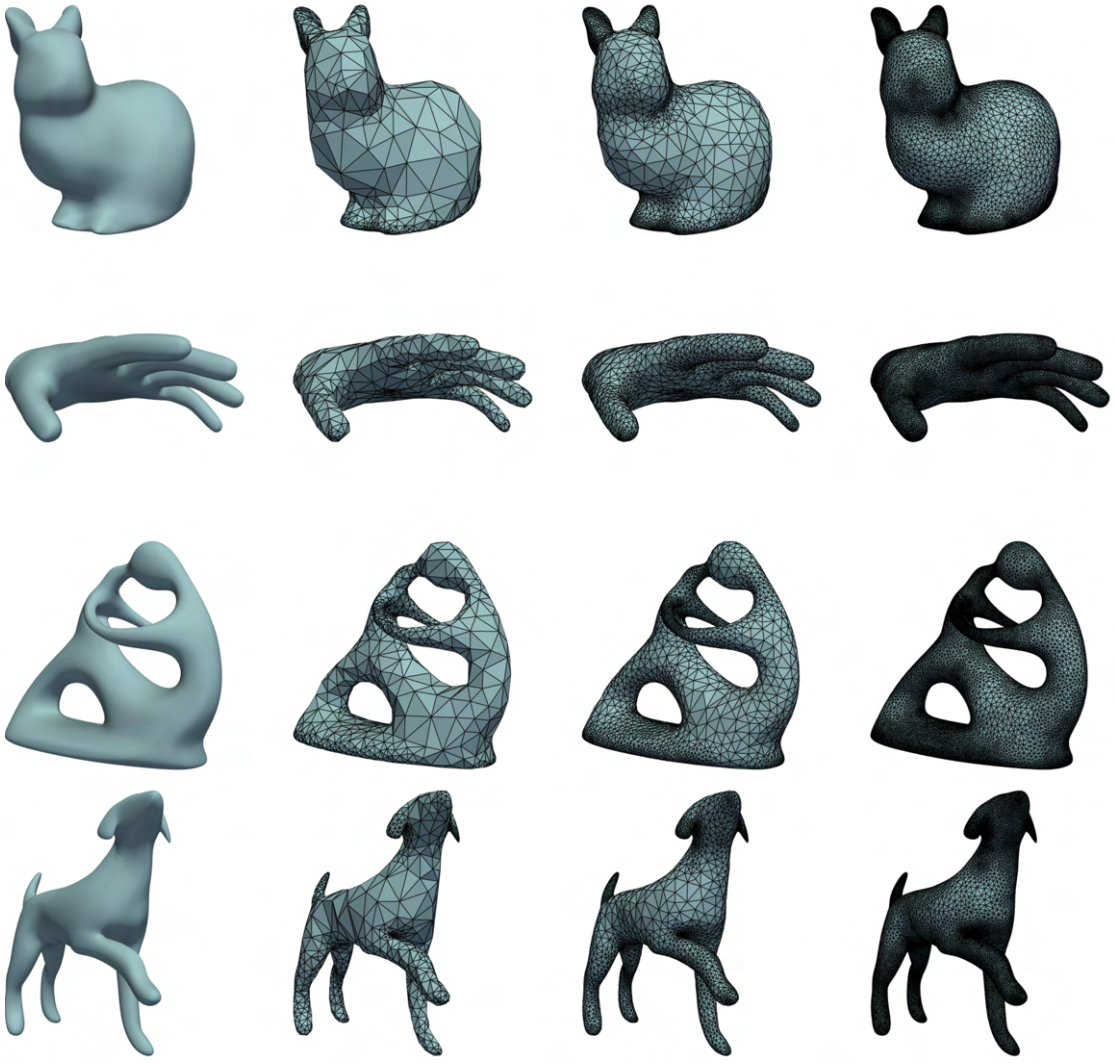

Figure 6.1: Loop subdivision surface and Delaunay-mesh to lfs for $\varepsilon = 1, 0.5, 0.2$: Bunny, Hand, Mother, Dog. Image source: private communication [Ohr].

CHAPTER 7

Conclusion and further work

In this chapter, we summarize our approach and its results and identify the most important issues that require further work.

7.1 Conclusion

In this thesis, we presented a method to approximate the medial axis (MA) and the local feature size (lfs) of Loop subdivision surfaces up to arbitrary precision and to turn the resulting lfs into a provably correct ε-sampling of the limit surface. In doing so, we generalized an existing two-dimensional, lfs-aware sampling approach [OPM23] from planar curves to Loop subdivision surfaces in $\mathbb{R}^3$.

The core idea of our approach is to operate directly on the smooth Loop subdivision surface $\mathcal{S}$ rather than on a polygonal approximation of it. We organized the computation as a four-step pipeline. First, the MA is sampled by growing tangent spheres, which reduces to a one-dimensional binary search in the sphere radius, where each candidate radius is tested through a Newton-based closest-point query on the surface patches, accelerated by an AABB hierarchy over the patch convex hulls. The collected sphere centers lie on the MA.

Second, the MA samples are meshed by inheriting the combinatorics of the foot point tessellation, which avoids the instability of applying a Delaunay triangulation to a raw medial point cloud and produces a triangle soup that supports fast distance queries for the lfs.

Third, the lfs is used to construct an ε-sampling by recursively subdividing each patch in its parameter domain until a single enclosing-sphere proxy, justified by the 1-Lipschitz continuity of the lfs, certifies the ε-condition over the whole sub-patch. We proved that this local criterion is sufficient to guarantee an ε-sampling of the entire surface.

Fourth, the samples are triangulated per patch through a constrained Delaunay triangulation in UV space, with shared patch boundaries enforced as constraints so that the output is a consistent closed manifold mesh.

Working on the smooth subdivision surface instead of a triangulation carries the central advantage of our method: the resulting MA is inherently well-behaved. Because the limit surface has no creases or vertices at which the lfs vanishes, the approximated MA does not reach into degenerate boundary regions, and its computation requires no pruning step, which is a notorious source of parameter sensitivity in MA methods that operate on polygonal meshes. The accuracy of our result is governed solely by the numerical tolerances of the underlying routines and by the chosen tessellation levels, so that the quality of the approximation is controlled by a single, monotone resolution parameter rather than by per-mesh fine-tuning.

Our experiments support these claims. In the absence of an analytic ground truth, we established convergence empirically: the RMSE of the lfs changes negligibly beyond tessellation level $\ell = 12$, and the Chamfer distance between our MA and those produced by PowerCrust and Q-MAT decreases consistently with refinement. In direct comparison with the state-of-the-art methods PowerCrust, VoxelCore, and Q-MAT, our approach captures finer medial features already at low tessellation levels, reaching a quality that competing methods attain only at substantially higher resolutions and after careful pruning-parameter tuning. On a torus subdivision surface, we confirmed that our reconstruction matches the expected medial geometry more closely than PowerCrust, and thus serves as the closest available proxy for a ground-truth validation. Another advantage of our method in this context is that it tessellates the control mesh on-the-fly, while PowerCrust, although being mostly faster, cannot handle meshes with millions of points and always constructs the Voronoi diagram. Finally, we saw that our ε-sampling method enables practical downstream applications, including triangle-budget-aware tessellation and Delaunay triangulation of subdivision assets as well as the use of a subdivision surface as an unbiased reference for evaluating surface reconstruction algorithms [Ohr].

Our method is not without limitations. Because our implementation builds on OpenSubdiv, which approximates the neighborhoods of extraordinary vertices with Gregory end caps, we strictly compute the MA of this end cap approximation rather than that of the exact Loop subdivision surface, which leads to additional spikes and leaves on the MA. The influence of extraordinary vertices on the lfs therefore remains only partially understood, as the end caps introduce localized reductions in the lfs whose limiting behavior we could not isolate. Furthermore, the closest point Newton scheme becomes more costly as tessellations are refined, sharp features in the control mesh trigger very deep recursion and correspondingly long runtimes, and ill-conditioned configurations near low-curvature focal points can yield oversized spheres. The ε-sampling itself is correct but far from minimal, since the subdivision criterion is a conservative per-patch upper bound and offers no global optimality guarantee. We discuss directions to address these shortcomings below.

7.2 Future work

Several extensions follow directly from the limitations identified in Ch.5.6.

7.2.1 Stam's evaluation

The most immediate extension concerns the treatment of extraordinary vertices. Replacing the OpenSubdiv Gregory end caps with an exact evaluation of the Loop subdivision surface, for instance through Stam's eigenstructure-based scheme [Sta98], would let us compute the MA of the true subdivision surface and finally clarify how extraordinary vertices shape the lfs in their vicinity, where our current results show end-cap-induced spikes and flaps whose limiting form remains open. Such an implementation is complicated though, since a control mesh can have vertices with arbitrary valence, requiring the implementation of all the corresponding basis functions.

7.2.2 Performance

The closest-point query dominates the cost of MA sampling; its runtime grows with tessellation level. A GPU implementation of the per-foot-point sphere growing and the ε-sampling per patch, which are already parallel, would make our method practical on larger and more detailed control meshes than the decimated models we used. Since OpenSubdiv is designed specifically for GPU applications, this improvement is much easier to implement than the first one. Further performance gains could come from adapting the patch tessellation level ℓ' for Newton start values to the size and curvature of the patch.

7.2.3 Tighter ε-sampling

A third direction concerns the ε-sampling itself. Our per-patch subdivision criterion guarantees correctness but produces far more samples than necessary. Tightening the subdivision condition and, more importantly, coupling the per-patch decisions across patch boundaries would move the result closer to a globally minimal ε-sampling and the lfs-aware Delaunay remeshing mentioned in Ch.6.2 suggests that substantial reductions are attainable.

7.2.4 Meshing the medial axis

One approach for constructing a topologically correct mesh of the MA without self-intersections would be to fix the regular triangulation from Ch.5.5.2. A regular triangulation method was used successfully in a recent publication [CD23] where the surface points were added with a designated weight. It could be suitable for adaptation to our case.

We could also try to fix the regular triangulation 5.5.2 using results and methods we already discussed. Two major faults need to be fixed: the presence of tetrahedra and the jagged boundary.

Tetrahedra could be removed by finding all tetrahedra that only have inner medial points as vertices. On each of these tetrahedra, we remove a triangle that intersects other triangles of the triangulation. If no triangle in the inner tetrahedron intersects any other triangle of our triangulation, we remove the largest triangle in the tetrahedron. The latter option would be a heuristic that could result in holes, a problem that would need further investigation.

An approach to fix the jagged boundary that is worth investigating could be the introduction of the boundary polyline we can obtain from the MA mesh that inherits the topology of the surface tesselation it comes from. The boundary can be found by filtering for very sharp crease edges or triangles that have a neighbour whose non-shared vertex lies close to the triangle. This is not a trivial task. Then the boundary edges must be added to the mesh from the regular triangulation. This, however, is difficult since the boundary vertices are not always present in the regular triangulation. Fig. 7.1 shows the boundary and the regular triangulation. We see that the boundary is so far from the mesh that lfs will vary considerably. The detail in Fig. 7.2 clearly shows the problem of boundary vertices that are not part of the mesh as well as boundary edges intersecting the mesh. Another problem to be solved that can be observed in this detail is self-intersection of the boundary.

If we only want to utilize the inherited tesselation topology rather than the regular triangulation, we have to remove redundant triangulations from every sheet of the MA since each sheet has been covered from both sides:
Pick a random triangle and add it to the new mesh. Then repeat the following steps.

1. Repeatedly pick a neighboring triangle and check for intersection with the new mesh.

2. In case of intersection: mark the triangle as a member of sheet i and as intersecting sheet.

3. In case of no intersection: Add triangle to the mesh in case of no intersection. If the previous triangle was intersecting, mark the shared edge with the current triangle as a seam.

This generates a mesh without intersections. The process triangulates sheets but leaves them as disconnected components. For a topologically correct mesh, we need to connect the sheets correctly. This can be done by finding for each edge marked as a seam the closest edge in the new mesh and merging them into a non-manifold edge. This approach might create some geometrically wrong connections but only at seams where multiple

sheets meet. See Fig. 7.3 for the process of identifying triangles of the mesh and Fig. 7.4 for the stitching process that connects disjoint medial sheets.

7.2.5 Curvature and focal points

The inverse of the largest principal curvature w.r.t. the medial sphere center direction is an upper bound of the medial radius. At extraordinary vertices, the second derivatives are not continuous. At these points it would be necessary to find the largest directional curvature at the extraordinary vertex for every patch incident to this vertex. Another curvature-induced problem in our context is focal points. Especially for large curvature, our sphere shrinking process is very sensitive to errors in closest-point calculation when the medial sphere at this point only touches the subdivision surface at this point. The reason for this is that the center of the sphere is very sensitive to the location of a second touch point that is very close to the foot point of the sphere. This issue is very hard to fix and needs further investigation.

7.2.6 Broadening the scope

Finally, the approach could be broadened in scope. Extending it to other subdivision schemes, to surfaces with intentional sharp creases and open boundaries, and to robust handling of self-intersecting subdivision surfaces would widen its applicability to industrial and artist-authored assets. Given that our method yields a reliable, parameter-light MA and lfs on the smooth subdivision surface, it could also be a promising building block for lfs-aware reconstruction pipelines, where it could provide the unbiased reference geometry against which sampling and reconstruction algorithms are measured.

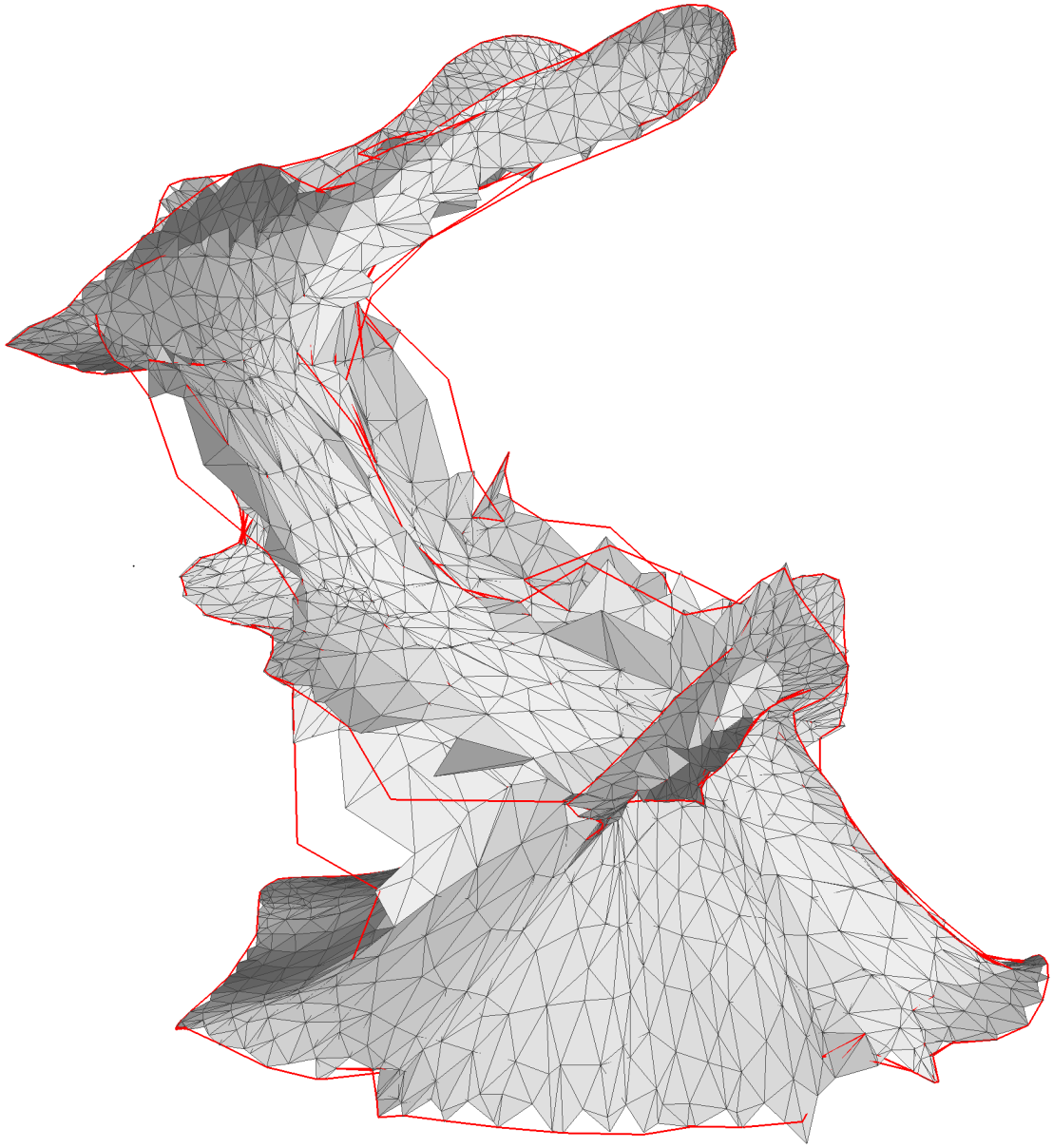

Figure 7.1: The inner MA triangulated with a regular triangulation that uses the squared medial radius as weight together with the boundary (red) of the inherited triangulation from the surface tesselation.

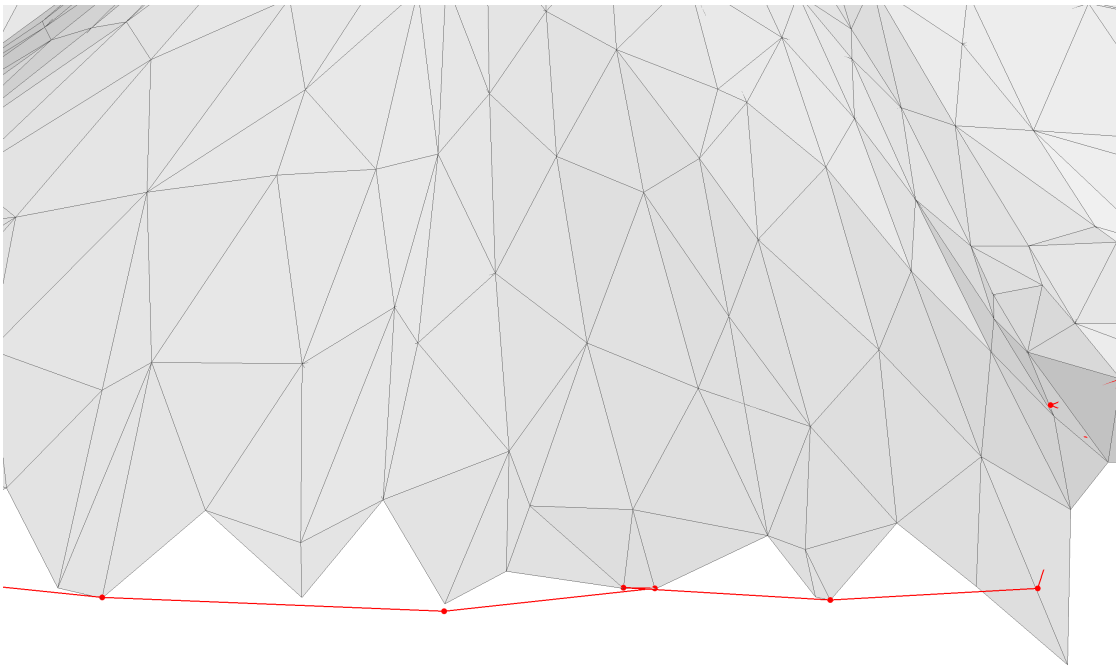

Figure 7.2: The inner MA triangulated with a regular triangulation that uses the squared medial radius as weight together with the boundary (red) of the inherited triangulation from the surface tesselation. Vertices of the boundary (red) are not present in the other triangle mesh.

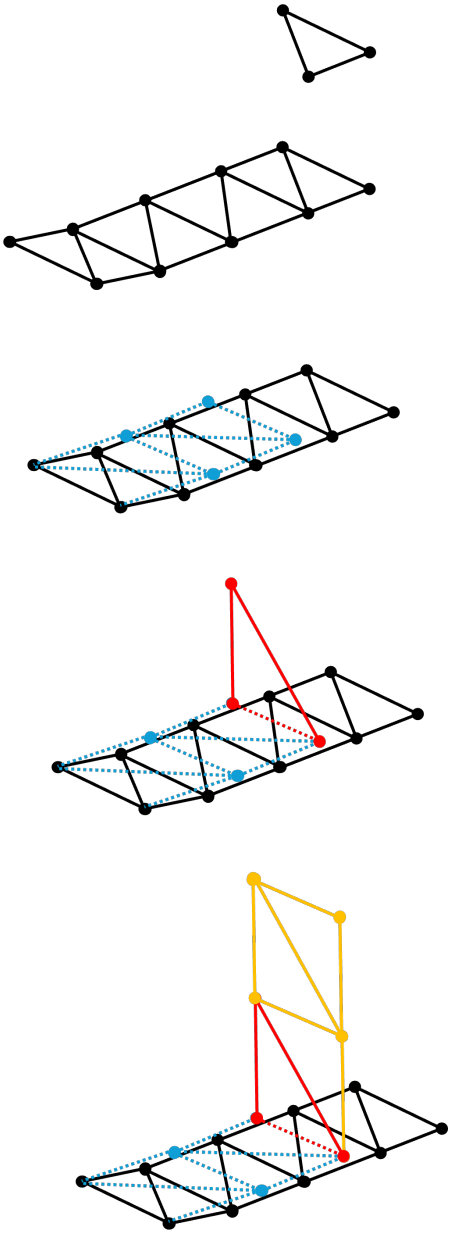

Figure 7.3: Process of creating a topologically correct MA mesh that is not self-intersecting from the mesh that inherits topology from the surface tesselation. Top to bottom: Start with a random triangle, then keep adding neighbours that do not intersect already added triangles (black). At some point, triangles that intersect the previous triangles are encountered (blue and dashed), which are discarded. If a non-intersecting triangle is encountered as a neighbor of a discarded triangle, it is a seam-triangle (red) and the shared edge is marked as a seam edge (blue/red dashed). Further triangles encountered after the seam triangle are orange and belong to another sheet.

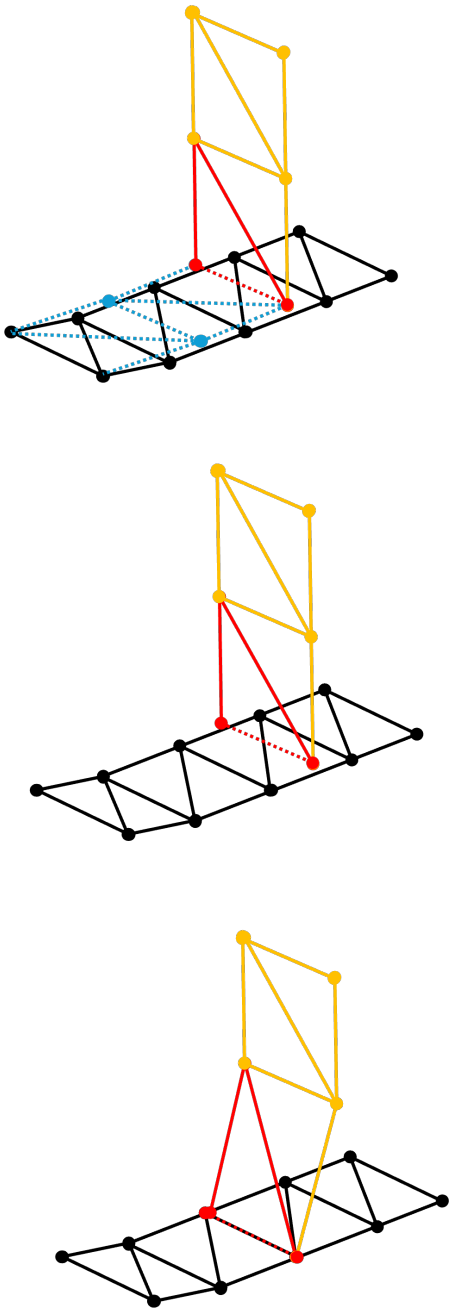

Figure 7.4: The stitching process for obtaining a topologically correct MA mesh. The intersecting triangles (blue, dashed) are removed. Then every seam edge (red, dashed) is merged with the closest edge

Overview of Generative AI Tools Used

The AI tools Claude Opus 4.8, Claude Sonnet 4.6, and MS Copilot Chat were used. For the implementation, these tools were used to find bugs, fix cross-platform issues, research numerical approaches, write scripts for analysis and debugging, and implementation of helper functions for data output, plots, and rendering. For the thesis, these tools were used to check spelling, wording, rephrasing, understandability of argumentation, translation, researching and verifying sources, as well as for resolving issues with LaTeXcommands or finding correct LaTeXsyntax.

Übersicht verwendeter Hilfsmittel

Enter your text here.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AAA+09] Oswin Aichholzer, Wolfgang Aigner, Franz Aurenhammer, Thomas Hackl, Bert Jüttler, and Michael Rabl. Medial axis computation for planar free-form shapes. *Computer-Aided Design*, 41(5):339–349, 2009.

[ABE09] Dominique Attali, Jean-Daniel Boissonnat, and Herbert Edelsbrunner. Stability and computation of medial axes: A state-of-the-art report. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pages 109–125. Springer, 2009.

[ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 415–421. ACM, 1998.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2–3):127–153, 2001.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Blu67] Harry Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

[BLZ00] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proc. 27th ann. conf. on Computer graphics and interactive techniques*, pages 113–120, 2000.

[BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

[Boo26] Boost Developers. Boost.math library. `https://boost.org`, 2026. Accessed: 2026-07-27.

[BP07] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics*, 26(3):72:1–72:8, 2007.

[Bre73] Richard P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[CD23] Mattéo Clémot and Julie Digne. Neural skeleton: Implicit neural representation away from the surface. *Comput. Graph.*, 114(C):368–378, 2023.

[CDRR04] Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Tathagata Ray. Sampling and meshing a surface with guaranteed topology and geometry. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 280–289. ACM, 2004.

[Che87] L Paul Chew. Constrained delaunay triangulations. In *Proc. 3rd annual symposium on Computational geometry*, pages 215–222, 1987.

[Che93] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 274–280. ACM, 1993.

[CL05] Frédéric Chazal and André Lieutier. The λ-medial axis. *Graphical Models*, 67(4):304–331, 2005.

[Dan80] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

[dBHR93] Carl de Boor, Klaus Hoellig, and Sherman Riemenschneider. *Box Splines*. Springer, New York, 1993.

[Dey07] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2007.

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, 1998.

[DLX+22] Zhiyang Dou, Cheng Lin, Rui Xu, Lei Yang, Shiqing Xin, Taku Komura, and Wenping Wang. Coverage axis: Inner point selection for 3d shape skeletonization. In *Computer Graphics Forum*, volume 41, pages 419–432, 2022.

[DZ04] Tamal K Dey and Wulue Zhao. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica*, 38(1):179–200, 2004.

[EH96] Matthias Eck and Hugues Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Proceedings of SIGGRAPH 1996*, pages 325–334. ACM, 1996.

[EM94] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[eot] edcorusa on thingiverse.com. Mother and child (simplified). `https://www.thingiverse.com/thing:456430`. Accessed: 2026-05-22.

[FHA24] Rao Fu, Kai Hormann, and Pierre Alliez. Lfs-aware surface reconstruction from unoriented 3d point clouds. *IEEE Transactions on Multimedia*, 26:11415–11427, 2024.

[GMPW09] Joachim Giesen, Balint Miklos, Mark Pauly, and Camille Wormser. The scale axis transform. In *SCG '09: Proceedings of the 25th Annual Symposium on Computational Geometry (SoCG)*, pages 106–115, 2009.

[Gre74] John A Gregory. Smooth interpolation without twist constraints. In *Computer aided geometric design*, pages 71–87. Elsevier, 1974.

[GWM24] Minghao Guo, Bohan Wang, and Wojciech Matusik. Medial skeletal diagram: A generalized medial axis approach for compact 3d shape representation. *ACM Transactions on Graphics (TOG)*, 43(6):1–23, 2024.

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH 1994*, pages 295–302. ACM, 1994.

[HKTB24] Qijia Huang, Pierre Kraemer, Sylvain Thery, and Dominique Bechmann. Dynamic skeletonization via variational medial axis sampling. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[HWQ+19] Jianwei Hu, Bin Wang, Lihui Qian, Yiling Pan, Xiaohu Guo, Lingjie Liu, and Wenping Wang. Mat-net: Medial axis transform network for 3d object recognition. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 774–781. International Joint Conferences on Artificial Intelligence Organization, 2019.

[JG+10] Benoît Jacob, Gaël Guennebaud, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[joT] johnny6 on Thingiverse. Low poly stanford bunny. `https://www.thingiverse.com/thing:151081`. Accessed: 2026-05-22.

[JST15] Andrei C Jalba, Andre Sobiecki, and Alexandru C Telea. An unified multiscale framework for planar, surface, and curve skeletonization. *IEEE trans. on PAMI*, 38(1):30–45, 2015.

[KBSS01] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 57–66. ACM, 2001.

[KG13] Kishor B. Kale and B. Gurumoorthy. Profile tolerance verification for free-form surfaces using medial axis transform. *Procedia CIRP*, 10:133–141, 2013.

[KL96] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of SIGGRAPH 1996*, pages 313–324. ACM, 1996.

[Lev44] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[Lie04] André Lieutier. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. *CAD*, 36(11):1029–1046, 2004.

[Loo87] Charles T. Loop. Smooth subdivision surfaces based on triangles. In *Master's thesis, University of Utah*, 1987.

[LSNCn09] Charles Loop, Scott Schaefer, Tianyun Ni, and Ignacio Castaño. Approximating subdivision surfaces with gregory patches for hardware tessellation. *ACM Trans. Graph.*, 28(5):1–9, 2009.

[LWS+15] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG)*, 35(1):1–16, 2015.

[MK05] Martin Marinov and Leif Kobbelt. Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models*, 67(5):452–473, 2005.

[MWK+08] Miriah Meyer, Ross T. Whitaker, Robert M. Kirby, Christian Ledergerber, and Hanspeter Pfister. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1546, 2008.

[NMM08] Yu Nishiyama, Masayuki Morioka, and Takashi Maekawa. Loop subdivision surface fitting by geometric algorithms. In *Poster Proceedings of Pacific Graphics 2008*, 2008.

[Ohr] Stefan Ohrhallinger. Illustrations. Private communication. 2026-06-08.

[OMW16] Stefan Ohrhallinger, Scott A Mitchell, and Michael Wimmer. Curve reconstruction with many fewer samples. In *Computer Graphics Forum*, volume 35, pages 167–176. Wiley Online Library, 2016.

[OPM23] Stefan Ohrhallinger, Amal Dev Parakkat, and Pooran Memari. Feature-sized sampling for vector line art. In *Pacific Graphics*, 2023.

[ow] Enalya on www.turbosquid.com. Hand - lowpoly. https://www.turbosquid.com/3d-models/3d-hand-basemesh-1504798. Accessed: 2026-05-22.

[Pixa] Pixar. Opensubdiv bfr documentation. https://opensubdiv.org/docs/bfr_overview.html#bfr-navlink-parameterization. Accessed: 2026-05-22.

[Pixb] Pixar. Opensubdiv release notes 3.5.0. https://opensubdiv.org/docs/release_notes.html#release-3-5-0-sep-2022, https://opensubdiv.org/docs/release_34.html. Accessed: 2026-06-08.

[PK99] Kálmán Palágyi and Attila Kuba. A parallel 3d 12-subiteration thinning algorithm. *Graph. Models and Image Proc.*, 61(4):199–221, 1999.

[PR08] Jörg Peters and Ulrich Reif. *Subdivision Surfaces*. Springer, 2008.

[pria] printable_models. Farm dog v2 3d model. https://free3d.com/3d-model/farm-dog-v2--499533.html. Accessed: 2026-05-22.

[prib] printable_models. Fish v1 3d model. https://free3d.com/3d-model/fish-v1--996288.html. Accessed: 2026-05-22.

[Proa] The CGAL Projec. Cgal 6.1. https://www.cgal.org/2025/10/01/cgal61. Accessed: 2026-06-08.

[Prob] The CGAL Projec. Cgal documentationcgal - 2d triangulations. https://doc.cgal.org/latest/Triangulation_2/classCGAL_1_1Constrained__Delaunay__triangulation__2.html. Accessed: 2026-06-08.

[PSL+25] Antonio Pepe, Richard Schussnig, Jianning Li, Christina Gsaxner, Dieter Schmalstieg, and Jan Egger. Deep medial voxels: learned medial axis approximations for anatomical shape modeling. *IEEE Transactions on Medical Imaging*, 44(12):4775–4786, 2025.

[PT97] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer, 2nd edition, 1997.

[PWG+19] Yiling Pan, Bin Wang, Xiaohu Guo, Hua Zeng, Yuexin Ma, and Wenping Wang. Q-mat+: An error-controllable and feature-sensitive simplification algorithm for medial axis transform. *CAGD*, 71:16–29, 2019.

[Rei95] Ulrich Reif. A unified approach to subdivision algorithms near extraordinary vertices. *CAGD*, 12(2):153–174, 1995.

[RG03] M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of planar domains with curved boundaries. *Computer-Aided Design*, 35(7):619–632, 2003.

[Ric05] Christophe Riccio. OpenGL Mathematics (GLM). `https://github.com/g-truc/glm`, 2005. Version 1.0.1, accessed 2026-07-31.

[RO26] Jörg Christian Reiher and Stefan Ohrhallinger. Epsilon-sampling on loop subdivision surfaces, 2026. Git repository.

[Rup95] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

[RW11] Alexander Rand and Noel J. Walkington. Delaunay refinement algorithms for estimating local feature size in 2d and 3d. *International Journal of Computational Geometry & Applications*, 21(5):507–543, 2011.

[Sch96] Jean E. Schweitzer. *Analysis and Application of Subdivision Surfaces*. Ph.d. thesis, University of Washington, 1996.

[SCYW16] Feng Sun, Yi-King Choi, Yizhou Yu, and Wenping Wang. Medial meshes – a compact and accurate representation of the medial axis transform. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1278–1290, 2016.

[SP08] Kaleem Siddiqi and Stephen Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 2008.

[SPSP02] Dirk Selle, Bernhard Preim, Andrea Schenk, and Heinz-Otto Peitgen. Analysis of vasculature for liver surgical planning. *IEEE Transactions on Medical Imaging*, 21(11):1344–1357, 2002.

[Sta98] Jos Stam. Evaluation of loop subdivision surfaces. In *SIGGRAPH'98 CDROM Proceedings*. Citeseer, 1998.

[Stu25a] Pixar Animation Studios. Opensubdiv - open-source subdivision surface library. `https://github.com/PixarAnimationStudios/OpenSubdiv`, 2025. Accessed: 2025-01-10.

[Stu25b] Pixar Animation Studios. Opensubdiv - open-source subdivision surface library. `https://www.opensubdiv.org/docs/images/bfr_tess_tri_order.png`, 2025. Accessed: 2025-01-10.

[TDS+16] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3d skeletons: A state-of-the-art report. *Computer Graphics Forum*, 35(2):573–597, 2016.

[TvW02] Alexandru Telea and Jarke J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the Symposium on Data Visualisation 2002*, pages 251–259, 2002.

[WHS+24] Ningna Wang, Hui Huang, Shibo Song, Bin Wang, Wenping Wang, and Xiaohu Guo. Mattopo: Topology-preserving medial axis transform with restricted power diagram. *arXiv preprint arXiv:2403.18761*, 2024.

[WSK+15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[YBM04] Y. Yang, O. Brock, and R. N. Moll. Efficient and robust computation of an approximated medial axis. In Gershon Elber, Nicholas Patrikalakis, and Pere Brunet, editors, *Solid Modeling*. The Eurographics Association, 2004.

[YLJ18] Yajie Yan, David Letscher, and Tao Ju. Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[Zor00] Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes, ACM, 2000.

[ZS84] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.