# Approches variationnelles pour le traitement numérique de la géométrie

Pierre Alliez

# Variational Approaches to Digital Geometry Processing

Habilitation Thesis

Pierre Alliez
INRIA Sophia Antipolis - Méditerranée

This habilitation thesis has been defended June 2, 2009 at INRIA Sophia Antipolis - Méditerrané.

Reviewers:
- Leif Kobbelt (RWTH Aachen)
- Helmutt Pottmann (TU Vienna)
- Sylvain Petitjean (INRIA)

Committee members:
- Leif Kobbelt (RWTH Aachen)
- Peter Schröder (Caltech)
- Andre Lieutier (Dassault Systems)
- Claude Puech (INRIA Paris)
- Sylvain Petitjean (INRIA Lorraine)
- Jean-Daniel Boissonnat (INRIA Sophia Antipolis - Méditerrané)

# Acknowledgments

I first wish to thank the reviewers of this habilitation thesis (Leif Kobbelt, Helmut Pottmann and Sylvain Petijean) as well as the committee members (Leif Kobbelt, Peter Schröder, Andre Lieutier, Claude Puech, Sylvain Petitjean and Jean-Daniel Boissonnat) for their time and encouragements.

I wish to thank many people with whom I have spent some time during my life in research in chronological order:
– Olivier Devillers who advised me during a stimulating internship in 1996, and Jean-Daniel Boissonnat who was heading the Prisme project-team of INRIA.
– Francis Schmitt who was my Ph.D. advisor at Telecom Paris, and leaved us in October 2008. He advised me to be tenacious when digging new ideas and to always keep my enthusiasm.
– Henry Sanson and Nathalie Laurent who were my Ph.D. co-advisors at France Telecom R&D, and with whom I have started working on geometry compression.
– Mathieu Desbrun who had invited me to work in 2001 as a post-doc at the University of Southern California. We have been continuously collaborating since in a very friendly and fruitful relationship. It is very enjoyable to be able to chat at any time about research on Skype or during our cross visits. Also, many thanks for your time on reading this habilitation thesis!
– Jean-Daniel Boissonnat again for hosting me in his group since 2002. I really appreciate to find a constant support and time for discussing new ideas and projects.
– David Cohen-Steiner for the many fruitful discussions and joint projects.
– Mariette Yvinec for the many scientific discussions and collaborations on European and national projects.
– Andreas Fabri for collaboration on CGAL projects, and SIGGRAPH and industrial courses.
– Bruno Lévy for the many discussions and collaborations.
– Craig Gotsman for invitation at Technion and collaboration.
– Mario Botsch, Leif Kobbelt and Mark Pauly for collaborating on courses at SIGGRAPH and EUROGRAPHICS.

– My former Ph.D. students Marie Samozino and Jane Tournois for all their efforts and patience.
– All interns from IT Bombay: Lakulish Antani, Ankit Gupta, Amit Gupta, Saurabh Chakradeo and Rahul Srinivasan for their enthusiasm.

I also wish to thank my parents who always encouraged me to live my passion for research. Last but not least, I wish to thank Magali and our two lovely daughters Sophie and Julie for their support, patience and infinite love.

# Abstract

This habilitation thesis presents a series of contributions in the field of digital geometry processing. These contributions offer concepts and algorithms for surface reconstruction, surface approximation, quadrangle surface tiling and isotropic tetrahedron mesh generation. The narrative aims at highlighting the common feature shared among our contributions: we adopt a *variational methodology* throughout this document, in the sense that we tackle each digital geometric problem by casting it as an energy minimization so that low levels of these energies correspond to good solutions of the problem. The main motivation behind such formulations is a significantly increased quality and robustness, sometimes at the price of heavier computations than greedy algorithms. The data structures and concepts involved in our work lie between computational geometry, geometric computing, and numerical computing. A general summary also provides a vision of the many remaining challenges in the field.

# Contents

# Chapter 1

# Introduction

Scientific discoveries in the field of digital signal processing have led the way to the spectacular technological advances that brought on the digital revolution that we are now experiencing. The sweeping impacts on multimedia, numerical engineering and numerical medicine have been both deep, and broad. We can try measuring technological advances with respect to the norms, quality standards and other numbers provided by the multimedia industry: mp3, mega-pixels, HD television, to cite a few. It is however important to distinguish a real evolution from a superlative marketing argument.

Just like for sound, images and videos, the grand challenge for digital geometry processing consists in elaborating a foundational theory *and* practical algorithms. For multimedia, engineering or computer-aided medicine, geometry plays the special role of premier support to the digital models. Each model used for the simulation of physical phenomena requires defining a domain (a geometry); e.g., a mechanical part, a terrain, some organs. Computer-generated movies, special effects and multimedia applications also require modeling complex geometries. The impact for the engineer, the geologist, or the surgeon now corresponds to a real change of paradigm. Numerical engineering substitutes the physical prototype and experience by a digital model and simulations. The core activity stays the same (conceiving, anticipating the real), but the efficiency is multiplied tenfold. The geologist can now simulate with a digital model at the scale of the Mexican gulf. A surgeon can simulate at the scale of a whole digital patient digitized with a variety of imaging modalities.

Nevertheless, geometry is not an ordinary signal due to distinctive properties such as topology, lack of trivial parameterization, irregular structure and non-uniform sampling, to cite a few. For these reasons, simple extensions of concepts and algorithms from digital signal processing do not

apply, and a dedicated research effort on digital geometry processing is required. Geometry processing is now a growing area of research that uses concepts from applied mathematics, computer science and engineering to design algorithms for the acquisition, reconstruction, representation, analysis, manipulation, simulation and transmission of complex 3D models.

## 1.1 Motivations

Contributions to the field of digital geometry processing are motivated both by scientific objectives and by a variety of applications. Fundamental scientific questions are related to sampling theory, information theory and approximation for instance. Applications of geometry processing algorithms cover a wide range of areas from multimedia, entertainment and classical computer-aided design, to biomedical computing, reverse engineering and scientific computing.

Research themes in geometry processing, aside from the mere alteration of geometry, include studying transition mechanisms between physical (real), mathematical (abstract), and discrete (digital) representations of complex shapes. Going from physical to digital is commonly referred to as shape acquisition and reconstruction. Going from mathematical to discrete is commonly referred to as mesh generation. The transition from discrete to physical is referred to as machining or rapid prototyping through 3D printing, which is now commonplace in numerical engineering applications. One way to structure the algorithms concerned with geometry processing is to enumerate them throughout the geometry processing pipeline:

- Acquisition. There exists a variety of geometric measurement devices based on e.g., contact sensing, laser scanning and structured lighting. The output of such devices is often composed of 3D point sets, points with attributes such as color, time-varying point sets, line of sight, and oriented normal estimates to the observed surface. 3D imaging devices such as tomographs or echographs are commonly used in medicine, geophysics, astrophysics or industrial control. The output is commonly a gray level image from which isosurfaces can be extracted, or a set of non-parallel slices. Photogrammetry makes use of regular digital camera and stereo vision techniques to recover 3D feature point sets from the salient feature points of the images. The output is commonly a set of 3D points with color attributes.

- Registration. The output of shape acquisition devices is often a series of raw samples of the observed physical shape, such as point sets. When each point set corresponds to a different acquisition viewpoint

and the pose of the acquisition device is not captured, a registration is required so as to align the data in a common coordinate frame. While many approaches employ rigid transforms [RL01] and/or require manual selection of pairs of corresponding feature points, research is carried out to automatize the process [AMCO08] and to employ non-rigid transforms [LSP08, BR04] for compensating for calibration errors, noise, and irregular sampling.

- Reconstruction. Assuming a set of registered sampled data sets such as points or slices, the process of shape reconstruction amounts to recovering a surface or a solid from these samples. This problem is inherently ill-posed since infinitely many surfaces approximating the samples can potentially exist. The challenge is even greater as the data set has variable density due to occlusions and other acquisition conditions, as well as uncertainty (noise, outliers) due to various sources of noise in the acquisition device and complex interactions between, e.g., the laser beam and the physical shape material. As the surface reconstruction problem poses many theoretical and practical challenges, it has received a considerable attention for over 25 years. Key theoretical issues are related to sampling conditions required to guarantee faithful topological and geometrical reconstructions of the physical shape [Dey06], inference models and resilience to noise. Practical issues are related to the size of the data sets generated by modern acquisition devices [BKBH07], as it is now common to acquire more than 100M sample points on a statue.

- Reduction. 3D models automatically generated by either reconstruction or by automatic surface extraction are often overly complex, thus calling for a reduction phase that simplifies the data (e.g, a surface mesh) while preserving its main features and geometry. This fundamental problem, related to surface approximation theory, has been tackled with a variety of paradigms ranging from mesh refinement to decimation through optimization [LRC+02]. Key issues are related to the choice of error metric [GH98, GWH01], topology preservation versus simplification [WHDS03], guaranteed tolerances [BBVK04, BF05], levels of detail, and view-dependence [HSH09]. Note that data reduction is more general than surface mesh simplification and can be applied to other geometric data such as point sets, slices, terrains, or volumetric meshes.

- Analysis. The analysis of complex shapes is central for quality control in industrial applications, for interactive modeling and processing, and to measure a notion of shape similarity. Quality control requires measuring a digitized representation of a manufactured object. Interactive modeling and processing requires efficient analysis on the geometry

and topology of the digital model for feedback and automatic choice of parameters for subsequent processing. Common analyses include estimation of normals [MNG04b], curvature [CSM03] and other differential properties, and various numbers in topology such as number of connected components, boundaries, tunnels and handles [DLSCS08]. More sophisticated analyses require the extraction of features such as sharp creases, cusps and corners [DHOS07]. These features can then be requested to be preserved during processing. Key theoretical issues are related to the consistence of the estimation with respect to the discretization of the shape, the independence to noise and discretization artifacts, and the inference model used for feature extraction. Pushing the analysis further leads us to the segmentation of a shape into geometrically coherent segments [Sha08], a problem as delicate as the image segmentation problem because inherently ill-posed. Another more global aspect of the shape analysis problem is the recognition of symmetries [MGP06] or structural regularities [PMW+08] in complex shapes.

- Processing. We use processing to refer to all methods which alter a shape so as to best prepare it for subsequent operations along the pipeline. Processing applies to virtually any type of geometric representation: point sets, meshes, time-varying shapes, etc. Although the variety of such methods is almost infinite, we can refer to the most common operations which range from denoising to idealization through smoothing and fairing. Denoising suppresses the noise in the spatial and/or time domain [SBS07]. The noise can be related to the geometry as well as to the topology as reconstructed surfaces can contain spurious handles which are not part of the physical shape. A key issue for smoothing is related to the distinction between noise and real details of the shape. Outlier removal is also part of the denoising problem [SBS05]. Smoothing, which alters the shape so as to reduce its high frequency details, can be constrained to preserve sharp features [FDCO03]. Fairing is also a form of smoothing usually aiming at generating energy-minimizing surfaces where the energy is related to thin plate or higher-order (polyharmonic) functions [SKS01]. One step further is the idea of shape idealization which amounts to replace parts of the shape by a set of ideals which are, e.g., parts of canonical shapes such as planes, spheres, etc. In addition to smoothing, more general curvature-domain processing can be applied—such as pass-band or exaggeration filters. The idea consists of computing a mapping from the spatial domain to the curvature domain, manipulating in the curvature domain and inverting the mapping so as to reconstruct a modified shape which best matches the desired curvature domain [ME08].

- Editing. At first glance shape editing is a pure modeling task and hence may not seem to be part of the geometry processing pipeline. Nevertheless, the recent trend to use polygon meshes throughout the whole geometry processing requires elaborating upon the best representations and modeling metaphors to make the editing process both intuitive and interactive [Bot05]. Before, the only way for an artist to change the pose of a character was to first model its skeleton and attach a skin to it, before animating the skeleton and re-fitting the skin. Later came the idea of embedding the shape into a cage to allow the artist modeling through editing the cage. The recent advances in shape editing are spectacular in the sense that an artist can efficiently pose a complex triangle mesh by acting on a few modeling handles or even simply relocating a few mesh vertices. Key issues are related to the preservation of details during editing [BSPG06], and to proper scale selection of the modeling metaphors.

  Obviously, all possible shape editing operations are impossible to enumerate as they are potentially as rich as the expression power of an artist. We can however list the ones which are common in geometric modeling software. Hole filling involves patching holes with plausible surfaces [Lie03], often derived from smoothness or continuity assumption with respect to the hole boundaries. Shape completion involves the replacement of missing parts of a shape with plausible shape fragments taken from the shape itself [SACO04] or from examples [PMG+05]. In the same vein, shape detail transfer is related to the idea of transposing details from one shape to another. The notion of animation transfer consists of applying automatically a designed animation from one shape onto another [SP04]. This way, an artist can, e.g., efficiently copy-paste the carefully designed animation of a running cat onto an elephant. Morphing is the process of automatically deforming one shape onto another, with plausible intermediate shapes [SK04]. A recent trend is to enrich the set of operations with more global ones. These include shape symmetrization [MGP07] and non-homogeneous resizing [KSCOS08]. The latter is motivated by the fact that uniform scaling is limited in its applicability while straightforward non-uniform scaling can destroy features and lead to visual artifacts.

- Simulation. Although the simulation of physical phenomena using, e.g., finite elements, is not directly related to geometry processing, we position it in the pipeline to emphasis the fact that it requires the preparation of quality meshes. The needs for flexible simulations on, e.g., deformable domains with cracks, require careful design of convex as well as non-convex polyhedral finite elements [MKB+08]. Note also that mechanics is often said to be all about geometry.

- Visualization. Although visualization is not the primary concern for the geometry processing community, modern graphics hardware architectures require special care at organizing the data in a cache-oblivious manner so that memory locality is maximized [YLPM05]. Clever representation [SNCH08] and ordering and of mesh primitives for efficient rendering is one of the topics of interest.

- Protection. A watermarking method faces two competing goals. On the one hand, the watermark should not degrade the data, e.g. significantly alter its visual appearance. On the other hand, the watermark should be as robust as possible, i.e. the extraction of the signature should be fast and stable, even under malicious attacks on the watermarked data [CDSM04]. A pirate might modify the original data with the sole intent to destroy the watermark, for example, by applying filtering or resampling operations. A watermarking scheme is considered robust if the successful removal of a watermark by these attacks leads to a severe degradation of the data, i.e. renders it useless for most applications.

- Transmission and storage. The limited capacity of networks and storage devices requires compressing geometric data. We distinguish lossless compression from lossy compression, and progressive compression well suited to the transmission over networks [AG05]. Progressive compression is related to the geometric approximation problem as the goal is to optimize the rate-distortion trade-off [KSS00]. Finally, a special class of networks such as broadcast networks, requires compression techniques resilient to packet losses as well as to distortion [PKL05].

- Searching and browsing. As for other multimedia data, geometric data sets must be amenable to efficient searching and browsing. This requires automatic methods to summarize large data bases with compact descriptors through analysis (see above) as well as methods for content-based retrieval [FKMS05]. Efficient browsing also requires methods for the automatic generation of visual summaries and thumbnails with the "best" viewpoint [PPB+05]. Best herein may refer to visibility criteria as well as to natural poses such as upright orientation of man-made objects [FCODS08].

- Printing and Machining. The last step of the pipeline consists of going back from digital to a physical model by either printing for rapid prototyping or by controlling a machining process such as numerically controlled milling machines. As a prerequisite, geometry and topology of the model must often satisfy a series of constraints such as watertightness, bounded minimum feature size, and tool path accessibility [PK08]. Additional questions arise such as feasibility of the process

with respect to the constraints of machining tools and to machine dynamics.

## 1.2 Approaches and Trends

Many approaches have been proposed along the geometry processing pipeline to tackle the issues we enumerated. Among all the corresponding algorithms the following summary is an attempt at identifying the main goals and choices that guided their development. It is also an interesting exercise to understand the coupling with the recent trends coming either from the nature and size of the data or from the recent technological advances in terms of graphics hardware and multi-core architectures.

**Smooth vs discrete.** One of the main choices which distinguish the different approaches is the type of representation to represent complex shapes. One dilemma is the following: should we use smooth surfaces? or polygon meshes? or just point sets? should the discrete mimic the smooth? or just be discrete all the way to be self-consistent. In practice the gap between geometric computing on linear objects and algebraic problems for geometric computing is substantial. Smooth representations have their own advantages and weaknesses, and in practice only low degree algebraic objects are really practical [Pet07]. Triangle or polygonal meshes are simple enough to elaborate complex algorithms, and the recent advances have shown that we can use them efficiently and all the way along the pipeline [BPK+07]. The current assessment states that it is more efficient to process many simple primitives than few complex ones. The interest for subdivision surfaces clearly points to the fact that something in-between discrete and smooth is perhaps a good trade-off. For rendering applications we have witnessed the development of pure point-based approaches [BPCZ07, Pau03], the argument being that the number of polygons of complex shapes leads to a rendering size lower than a pixel.

**Automatic vs Interactive.** The quest for automatic algorithms with very few parameters is prevalent in the literature. Nevertheless a number of algorithms along the pipeline described above are either inherently ill-posed, or in essence, necessarily interactive as for editing. For this reason one thread of work focuses on producing reasonable solutions in a fully automatic fashion, while another thread delegates the handling of ambiguous cases to the user. For the latter case a key issue is then to propose new interaction metaphors and meaningful parameters for the user. This is the case, e.g., of

a recent reconstruction technique [SLS+07] which lets the user decide over the topology of the reconstructed shape.

**Global vs Local.** A majority of problems along the pipeline can not be solved solely at a local level. This is particularly true for surface reconstruction, registration, hole filling and other complex editing tasks. Somehow the global nature of some problems goes against the idea of elaborating upon an algorithm represented as a sequence of elementary tasks. We can distinguish roughly three approaches which i) rely on global geometric data structures and greedily apply elementary operations; ii) aim at bridging the gap between local and global through hierarchical or multi-resolution data structures; or iii) reformulate the problem numerically so that the solution emerges after global numeric computations. For surface reconstruction these explain the differences between, e.g., a ball pivoting [BMR+99], a shrink wrapping [KVLpS99] and a graph-cut approach [HK06a].

**Guarantees.** The quest for reliable computing motivates the elaboration of algorithms with theoretical and practical guarantees. A guarantee is obtained either by proving that the sequence of elementary steps of the algorithm effectively produces what it is supposed to, or simply by construction. An example of the first case is the proof of termination of a mesh generation algorithm [BO05], and an example of the second case is the absence of self-intersection in 3D tetrahedron mesh represented as a 3D Delaunay triangulation. Carrying on a theoretical analysis for an algorithm may require prior assumptions over the input data, which are not always in line with real data. This is the case of, e.g., Delaunay-based surface reconstruction approaches which guarantee a faithful geometric and topological reconstruction if and only if the shape is smooth, and the sampling is both noise-free and dense enough [Dey06]. These conditions are quite often not matched in practice.

**Acquired Data.** An increasing trend is to process data acquired from the physical world, such as laser scanned point sets for reverse engineering, seismic data in geology, 2D satellite images for dense photogrammetry and 3D images in medicine. Such data are often uncertain (noisy), of variable density (up to missing data) and may contain outliers. Uncertainty is due to the noise (electronic, optical) of the shape acquisition device cumulated with the complex interactions between, e.g., the laser beam and the material of the acquired shape. Variable density is due to the variability of an acquisition procedure (number of passes, distance to the object, orientation of a laser beam, occlusions, etc.). Outliers arise when, e.g., carrying on photogrammetry on mismatched feature points. Although it is tempting to

wait for technological advances which improve the acquisition devices so as to reduce noise and accuracy, here instead the trend is to choose acquisition devices with fewer constraints (contactless, based on ordinary cameras). We are witnessing two main threads of work: algorithms for repairing such raw, flawed data (see e.g., [BPK05]) or algorithms which are resilient to noise (both geometric and topological), outliers, missing data and variable density (see, e.g., [LPK09]).

**Massive Data.** Modern acquisition devices generate massive data sets which do not fit into memory and pose many other scalability issues in terms of storage and computation. One thread of work consists in trying to deal with all such data, by elaborating upon out-of-core algorithms, which load only small subsets of the data into main memory at any given time. Another thread of work, parsimonious, consists in considering only a small fraction of the data by, e.g., incorporating into a surface reconstruction algorithm the only points that brings new details to the surface [BC01]. One more general trend in signal and image processing, so-called compressive sensing, consists in sampling and simultaneously compressing the signal at a greatly reduced rate [Can06]. The key argument is that, in practice, the data are lossy compressed and simplified soon after sensing. The goal is thus to avoid such waste of valuable sensing resources.

**Dimension.** Another trend is to deal with data in higher dimensions. This is the case of time-varying geometric data sets, as well as point sets in high dimension for, e.g., shape recognition. The key questions that arise are establishing a theory and algorithms for analysis (manifold learning) and processing of such data.

**Parallel Computing.** The recent technological advances show an explosion of parallel architectures, both for personal computers and for supercomputers. Dual and quad-core CPUs with shared memory architecture are now common on desktop computers. Computational grids with thousands of CPUs and distributed memory are used to solve large-scale simulations. Moreover, modern graphics processing units (GPUs) are used to efficiently manipulate and display computer graphics. In addition, their highly parallel architecture makes them very effective for more general algorithms. This evolution requires considering new ways of conceiving geometry processing algorithms, with an eye towards hardware characteristics.

# Chapter 2

# Contributions

Before describing our collection of research contributions to the field of digital geometry processing, we provide a high-level picture that stresses out our underlying unified approach as well as the fundamental concepts behind the resulting algorithms.

## 2.1 Favored Approach

If we had to summarize our choice of approach in a few words, the terms "variational, discrete, and integral" would be most appropriate.

**Variational,** in this document, refers to an approach that cast a problem as a functional minimization (or extremization in general), such as low levels of the energy correspond to good solutions to the problem. The definition is left intentionally vague for the moment, and phrasing good solutions instead of optimal solutions alludes to the fact that globally optimal solutions are often a mirage for many problems of interest. Variational approaches are designed to maximize the quality of results; although this may come at the price of heavier computations than greedy algorithms, there are a number of applications such as mesh generation where quality supersedes computational cost.

**Discrete** refers to the fact that we are dealing with discrete data as inputs (point sets, triangle meshes, tetrahedron meshes, finite dimensional data) and that we are using discrete data and formulations all the way in the algorithms without resorting to smooth (high-order) geometric primitives approximating the input data.

**Integral** indicates a preference for computations through integration over domains instead of point-wise computations. This is motivated by the desire to obtain improved robustness of the algorithms (e.g., resilient to noise), reduced dependency on input sampling and self-consistence of the computations with respect to the discrete nature of the data structures.

The contributions presented in our narrative are located at the intersection of geometry processing, computational geometry and geometric computing. Geometric computing is purposely distinguished from computational geometry here in the sense that it is concerned with practical and robustness issues and real-world computers with finite, hence limited, capabilities. Geometry processing herein refers to the fact that the geometric content is altered or reconstructed, see the pipeline Section 1. The fundamental geometric data structures involved in the algorithms include Delaunay triangulations, Voronoi diagrams, and surface triangle meshes. Other auxiliary geometric data structures such as KD-trees or AABB-trees are used to improve efficiency. The fundamental concepts involved in the algorithms range from principal component analysis to clustering through harmonic one-forms and function interpolation. The numerical aspects involve solving linear systems as well as generalized eigenvalue problems.

We show in the next sections how these concepts are put to work in order to generate smooth surface reconstructions, faithful approximations, well-shaped quadrangle surface tilings and isotropic tetrahedron meshes. These four contributions have been selected as they provide a good representation of our favored approach. For each contribution we present a synthesis of the approach and some results. When available, we list the follow-ups by other researchers in the field, and we present our vision for future work.

## 2.2 Surface Reconstruction

In this section we focus on surface reconstruction from unorganized point sets. This problem is motivated by a number of CAGD, point-based graphics, and reverse engineering applications where scattered point samples of a surface need to be turned into a proper surface (see Section 1). Challenging are point sets generated by laser scanners and hand-held digitizers, as they are often noisy (due to the inherent uncertainty of measurements and merging of several scans), of variable density, and contain large holes due to occlusions during the acquisition process.

### 2.2.1 Related Work

Delaunay-based surface reconstruction techniques were initially designed to establish a plausible connectivity between points [Boi84]. One of the first reconstruction techniques that came with theoretical guarantees was proposed by Amenta and Bern [AB99]. The rationale behind their technique was that when a sampling is noise-free and dense enough, all Voronoi cells are elongated in the direction of the normal to the inferred surface. An analysis of the point set's Voronoi diagram can then be used to derive an interpolating reconstructed surface. This technique has stimulated many improvements and variants: we refer to [CG06] for a survey, and to [Dey06] for a comprehensive monograph. In practice however, most of these Voronoi-based techniques are interpolatory, thus not so adequate in the presence of noise.

Noise and sparseness in point sets have led to an approximating class of approaches, where an implicit function is computed so that one of its isosurfaces best fits the data points (see e.g., [OBA+03a]). These implicit approaches mostly differ in the norm used to match the point sets and in the isosurfacing. Examples of such implicit functions include approximations of signed distance functions to the inferred surface [HDD+92, CBC+01, BC02]. A reconstruction method for oriented point sets was introduced in which an implicit function $f$ is derived from a Poisson equation $\Delta f = div(\mathbf{n})$, providing the best $L^2$-match between the gradient of $f$ and the input oriented normals $\mathbf{n}$ [KBH06]. This algorithm scales well as it involves solving a sparse linear system, and its global $L^2$-minimization nature makes it resilient to noise. As for the Poisson approach, most current implicit techniques require a consistent orientation of the normals to perform correctly. Unfortunately, and unless reliable oriented normals are provided, finding such an orientation adds another ill-posed problem when the sampling is sparse and noisy. One of the approaches to normal orientation is through labeling a set of Voronoi poles; but it requires little or no noise and a dense-enough $\epsilon$-sampling to guarantee consistent results, two conditions rarely met in practice.

Given the intrinsic difficulty to estimate and orient normals, some work has polarized on handling raw point sets without attempting to locally estimate or orient the normals (e.g., [KSO04, WCS05, HK06b, PSQ06]). More recently, a spectral reconstruction method [KSO04] has been shown remarkably robust to outliers due to its reliance on graph partitioning algorithms [SM00]. However, this method is interpolatory, requiring post-smoothing for noisy point sets. Similarly, Hornung and Kobbelt [HK06b] propose a min-cut algorithm on a regular grid which attributes are derived from a probability density function. Finally, an approach based on eigen analysis

but derived purely from an optimization standpoint, offers an approximating reconstruction where smoothness can be controlled [WCS05] although various coefficients require careful adjustment to provide satisfactory results.

### 2.2.2 Approach

The key idea of the proposed approach consists of delegating the normal orientation to the implicit function solver. We first perform a robust Voronoi-PCA estimation of unoriented normals induced by the point set. This step results in a tensor field which encodes both the (unoriented) normal direction through its principal component and the confidence in the approximation through its anisotropy. Second, an implicit function is computed via solving a generalized eigenvalue problem so as to make its gradient best fit the principal components of the tensors, see Figure 2.1. As the solving step favors both a large aligned gradient of the function and smoothness tangentially to the inferred surface, a consistent orientation comes out as part of the solution.

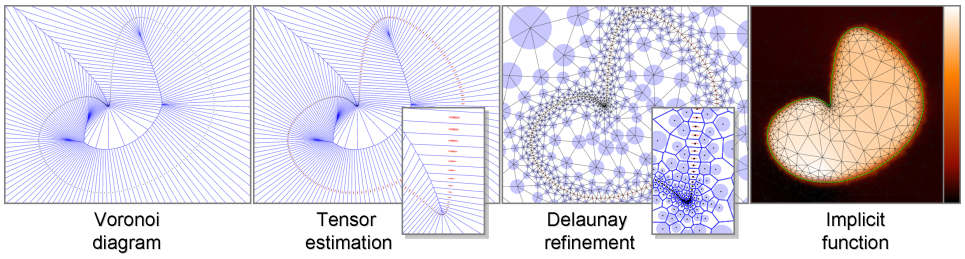



**Figure 2.1:** Reconstruction procedure. From left to right: input point set and its Voronoi diagram; covariance matrices of the cells shown as (rescaled) ellipses; Steiner points added through Delaunay refinement (isotropic tensors are assigned to Steiner points); piecewise linear function $f$ (solution of a generalized eigenvalue problem) that best fits the input data, with the reconstructed curve (iso-contouring of $f$).

#### 2.2.2.1 Normal Estimation

Our first goal is to estimate unoriented normals along with their reliability to the inferred surface from the input point set. We do not try inferring an orientation, as this global task is incumbent upon the second step of our approach. As a way of motivating our estimation, we briefly review a few closely related techniques.

Normal estimation from point sets has received a lot of attention in the past few years, see e.g., [PKKG03, MNG04a, DLS05, OF05, LP05,

HXMP05]. Various strategies have been proposed, guaranteeing high-order accuracies through, e.g., local fitting [CP03].

**Principal Component Analysis (PCA)** A conventional technique for estimating normal directions is through local principal component analysis. The main idea is to define a small neighborhood around each input point (e.g., the $k$-nearest neighbors [HDD+92]), compute the covariance matrix of the points in this neighborhood, and deduce the normal direction from the eigenvector associated to the smallest eigenvalue of the resulting covariance matrix. Variants have been proposed to improve resilience to noise, see for instance, [MNG04a].

**Voronoi Poles** Another common technique for estimating normal directions is more global by nature as it requires the construction of the Voronoi diagram of the input point set. A subset of Voronoi vertices called poles [AB99] is extracted, and used to estimate a normal direction at each sample point. In absence of noise and for dense-enough samples this method provides a faithful normal estimate even for irregular sampling, with convergence rates depending on the elongation of the Voronoi cells. A variant by Dey and Sun [DS05] provides resilience to noise.

We propose a normal approximation technique that combines both of their qualities. We begin by computing the 3D Voronoi diagram of the input point set after adding dummy sample points on a very large bounding sphere so as to bound the Voronoi cells. As the shape of these cells reflects the global distribution of the points in space, a key observation is that the covariance matrix of the Voronoi cell of a sample point provides not only an estimate of the normal direction, but also a measure of its reliability. Its principal component indicates the axis along which the cell is elongated (see inset), i.e., a good approximate of the normal direction if the samples lie on a manifold surface. Moreover, as described in [AB99], the confidence in the estimate is related to how long and thin the Voronoi cell is, i.e., in our context to the anisotropy of the covariance tensor. If the sampling is of good quality, this procedure will be very accurate as each Voronoi cell is long and skinny. However, as Figure 2.2(middle) illustrates, Voronoi cells can become small and/or isotropic if noise is present.

**Covariance Matrix of a Union of Voronoi Cells** To render our estimate robust to noisy point sets, we compute the covariance matrix of a

union of Voronoi cells[1]. As the Voronoi diagram of the point set partitions the whole domain, elongated cells are present beyond the noisy area, and accumulating enough neighbors will eventually render the union elongated enough (see Figure 2.2(right)). Although the idea of combining the influence from neighbors to promote noise resilience is commonplace, our technique is adaptive in the sense that we use as many neighbors as needed to find a reliable approximation as described next. This approach differs from [DS05] by integrating instead of searching for the most elongated Voronoi cell in a neighborhood.

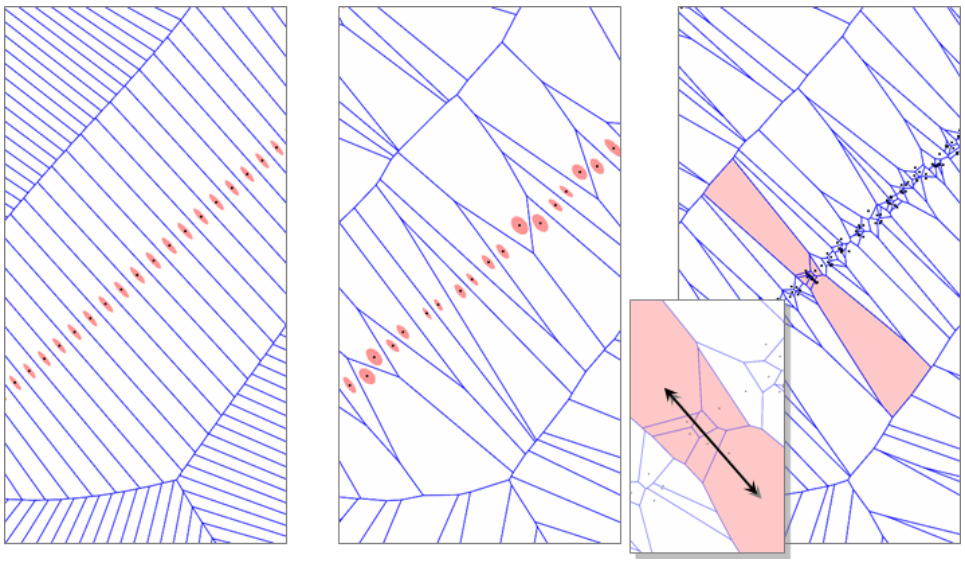


**Figure 2.2:** Voronoi cells of point sets. Left: Voronoi diagram of a point set without noise. Middle: with noise the cells become irregular. Right: a denser, noisy point set shows even more diverse cell shapes.

**Estimation Procedure** Given a sample point $p$, we first compute the covariance matrix of its Voronoi cell $V(p)$, and measure its anisotropy $\sigma \in [0,1]$ as $\sigma = 1 - \text{isotropy}(V)$, where $\text{isotropy}(V)$ is the ratio between the smallest and the largest eigenvalues. We then iterate over its $k$ nearest neighbor points $\{q_i\}$, and perform the same procedure for $V(p) \cup V(q_1)$, then for $V(p) \cup V(q_1) \cup V(q_2)$, etc., until either the anisotropy is sufficiently high, or we reach the maximum number of nearest neighbors specified by the user (typically $k = 50$). From these covariance matrices, the one with maximum anisotropy is returned (see Figure 2.3). When the sampling is dense and noise-free this evaluation procedure stops at a single Voronoi cell.

[1]The covariance matrix of a union of Voronoi cell is computed in closed form and requires tessellating all cells into tetrahedra. Details about such computations now available in the CGAL library are detailed in a technical report [GAP08].

There is indeed no incentive to add an extra neighborhood Voronoi cell as it would mainly thicken the integration domain, thus decreasing anisotropy. In presence of noise however the same procedure can also stop at a single Voronoi cell in the rare cases where the cell is very anisotropic yet inside the noise. We prevent this issue by narrowing the search interval through starting the procedure at $k = 5$.

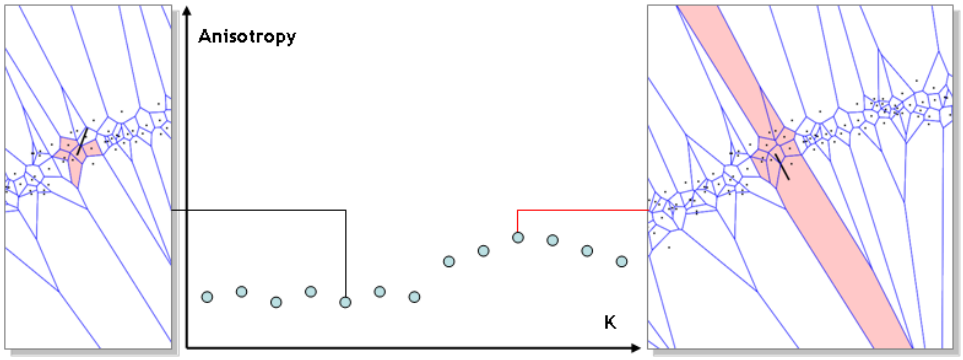



**Figure 2.3:** Tracking maximum anisotropy of unions of Voronoi cells. Left: for $k = 5$ Voronoi cells the anisotropy is low and the principal component of the covariance tensor is a bad normal estimate. Right: for $k = 5$ the anisotropy is maximum and the normal estimate is better. Further increasing of $k$ would mainly thicken the integration domain and reduce the anisotropy.

This procedure benefits from both the qualities of PCA (local analysis of a number of neighboring samples) and those of the Voronoi-based approach (global analysis of the sample repartition via the Voronoi diagram). Such Voronoi-PCA normal estimation technique can be seen as an integral PCA approach, similar to [PWY+06] for curvature estimation. Integration leads to more stable estimations compared to the pole approach which bases its estimation on the position of a single pole. In addition, it provides the surface reconstruction procedure described next with a local characterization of the sampling quality (see Figure 2.4). Such characterization is effective in the following sense: where the sampling is very dense the gradient of the implicit function is highly constrained to be aligned with the principal component of the tensors, while the function is favored to be smoother on areas where the sampling is sparse.

We illustrate how our integral-based normal estimation technique provides improved numerics compared to the usual pole-based approach by examining a very simple point set configuration, where 2D points are sampled uniformly along two parallel lines. When the bottom line is slowly shifted, the shape of the Voronoi cells evolves, triggering changes in the pole-based normal estimates. Because our approach relies on the integrated moment of the cells, it is significantly less sensitive to the shift as shown by the curve in Figure 2.5. Notice also that while a $k$-nearest neighbor PCA approximation

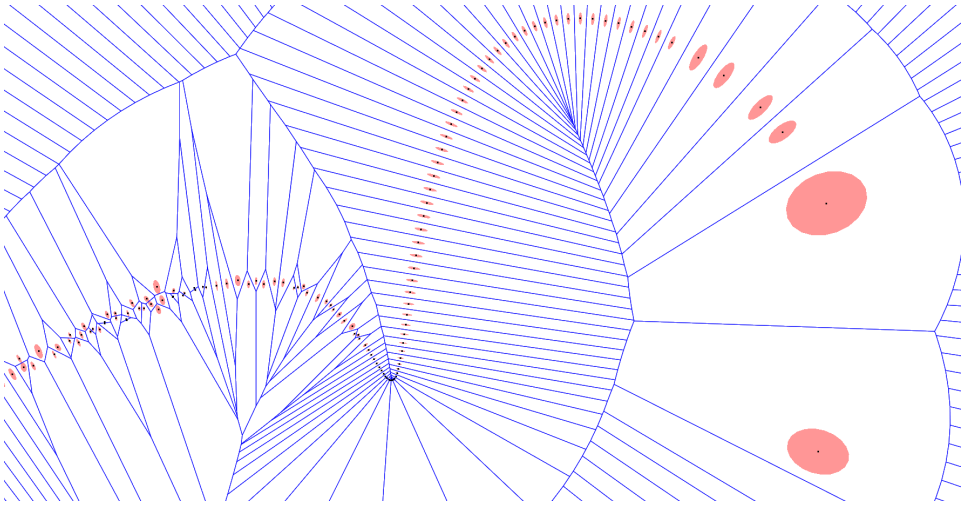


**Figure 2.4:** Variable sampling quality on a 2D curve. A 2D curve is sampled with noise (left), noise-free (middle), and with variable sampling (right).

loses accuracy when the (Euclidean-based) neighbors include points of both lines, the proposed approach benefits from the global nature of the Voronoi diagram, making it more robust to sparse sampling.

The next experiment compares the proposed normal estimation technique for 3D parametric surfaces with both pole and point-based PCA approaches. We sample a height field $z = sin(x)cos(y)$ (for which normals are known analytically) with different sampling criteria:

- Noise free. The height field is first sampled on a regular grid in parameter space in the interval $[-\pi, \pi]^2$, with rates ranging from $20 \times 20$ samples to $100 \times 100$ samples.
- Noise in parameter space. The height field is then sampled on a jittered grid in parameter space, with the same various densities as above. The noise is uniform and isotropic, with a maximum magnitude of half the grid spacing to make it scale with sampling density.
- Noise in embedding space. The samples of the first case (regular grid) are now jittered in 3D using an isotropic uniform noise of half the grid spacing.

In these three contexts, we measure the average angle deviation between normal estimates and true normals for each sample density. For the point-based PCA technique, we always use the 8 nearest neighbors as it leads to the best estimates. As Figure 2.6 indicates, our approach is, across all tests, either as good as or better than the two normal estimation techniques that

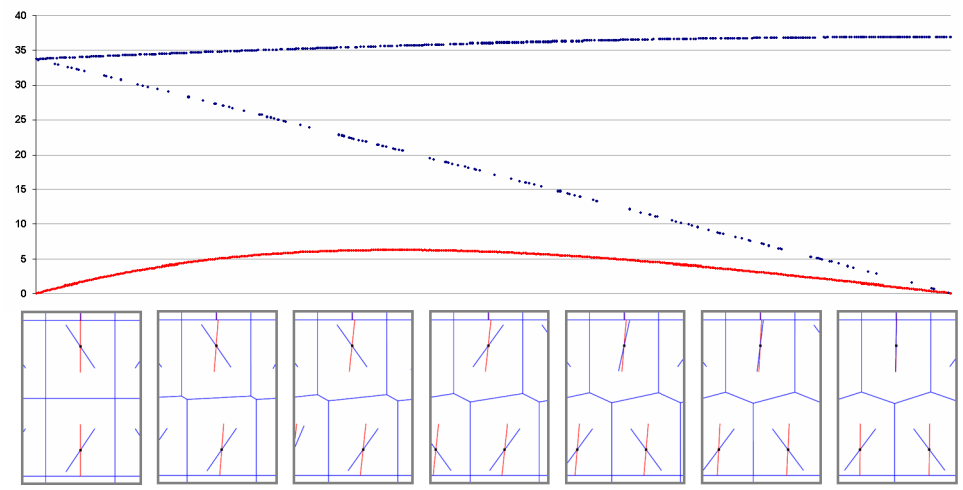



**Figure 2.5:** Poles vs. covariance matrices in 2D. A point set samples two parallel lines; (bottom) from left to right, we translate the samples of the bottom line slowly; pole-based normal estimates are depicted in blue, while our covariance-based normal estimates (deriving from the Voronoi cells displayed as well) are in red; (top) the two curves compare the pole-based results (blue, very discontinuous) with the covariance-based estimate (red) using the angle error (in degrees) as the bottom line of points is shifted.

it is built upon. Comparing various normal estimation techniques fairly is notoriously difficult as many parameters come into play, such as sampling density, sampling anisotropy, noise, so our tests are not intended to be extensive and conclusive. Nevertheless, we conducted several other experiments (including a comparison with [MNG04a]) that all show numerical relevance.

#### 2.2.2.2 Generalized Eigenvalue Problem

We now wish to compute an implicit function which gradient best aligns to the (unoriented) normal direction estimates. While a Poisson reconstruction [KBH06] can directly give us an implicit function so that its gradient best fits a normal field, we cannot have recourse to such a direct linear solve as we only have a direction field (i.e., unoriented normals). We need instead to find an implicit function $f$ whose gradient $\nabla f$ is everywhere best aligned to the the principal component of the tensor field $C$, where the notion of est alignment is weighted by the local confidence in the normal direction. We propose the following constrained maximization procedure to efficiently find such a function $f$:

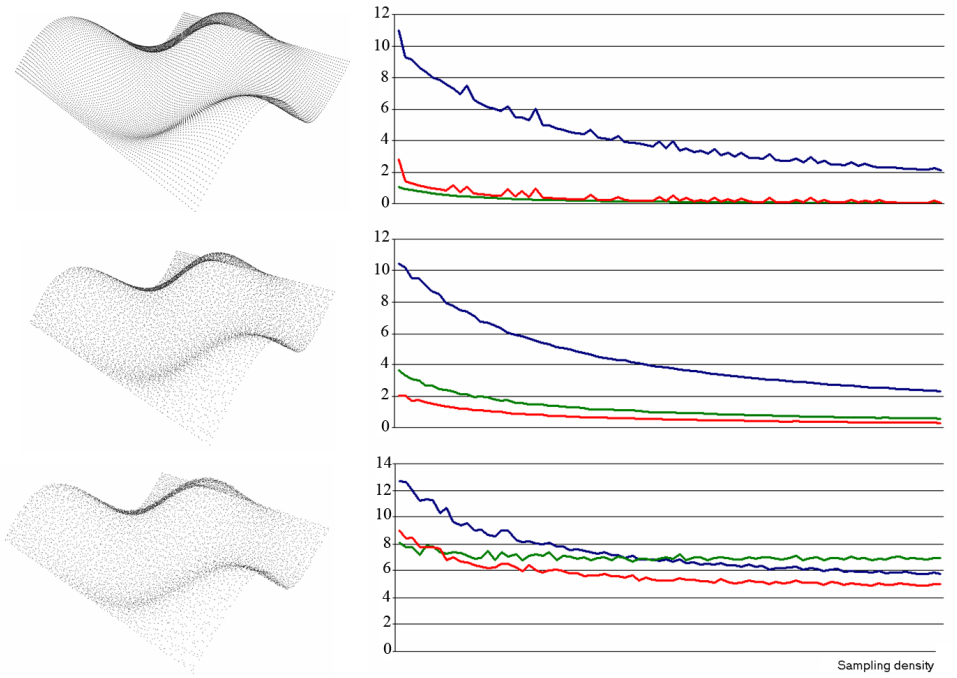



**Figure 2.6:** Normal estimation on a parametric surface. (top) noise-free sampling, (middle) noise added in parameter space, (bottom) noise added in embedding space. The plots show the average angle deviation in degrees against the sampling density. Pole-based estimation in blue, point-based PCA in green, and our covariance-based estimation with a single Voronoi cell in red.

Given a tensor field $C$, find the *maximizer* $f$ of:

$$E_C^D(f) = \int_\Omega \nabla f^{\,t}\, C\, \nabla f \text{ subject to: } \int_\Omega \left[|\Delta f|^2 + \epsilon |f|^2\right] = 1,$$

where $\Omega$ is the domain, and $\Delta$ is the Laplacian operator.

The interpretation of this optimization problem is as follows. The energy function $E_C^D$, called anisotropic Dirichlet energy, directly measures the alignment of $\nabla f$ with the normal direction indicated by $C$. An isotropic tensor (i.e., unknown normal direction) has little influence on this energy, whereas an anisotropic tensor (i.e., high confidence in the normal direction) will play a major role, penalizing misalignment at reliable data point. We then add as a constraint that $E_C^D$ must be maximized over the unit ball defined by the biharmonic energy. Just like the Dirichlet (harmonic) energy is a measure of the smoothness $f$, the biharmonic energy measures the smoothness of $\nabla f$: therefore, this added constraint imposes a regularization of the maximizer $f$. A small amount of the $L^2$ norm of $f$ is added to avoid having to constraint values of $f$ (either on the boundary or inside the domain), as

well as to improve conditioning: the resulting constraint is a Sobolev-like norm $E^B$ on $f$. Solving for this constrained maximization amounts to carefully balance smoothness of $\nabla f$ vs. alignment of the gradient: it will align $\nabla f$ to $C$ if it is particularly rewarding for the anisotropic Dirichlet energy (i.e., when the normal direction is reliable); on areas where the tensor is isotropic, the solver will favor smoothness of the function gradient instead. This global balancing act implicitly induces a consistent orientation to the tensor field since flipping the sign of $\nabla f$ along the tangential direction of the inferred surface significantly increases the biharmonic energy. The normal orientation is this way delegated to the solver.

**Discrete Formulation** We now assume that we have a tetrahedral mesh of the 3D domain with $V$ vertices $\{v_i\}$ and $E$ edges $\{e_i\}$. Each edge $e_i$ is arbitrarily oriented. Given this mesh with a tensor $C_i$ at each vertex $i$, we wish to solve for a node-based, piecewise linear function $f$, i.e., to find a vector $F = (f_1, f_2, \ldots, f_V)^t$ that satisfies the aforementioned constrained maximization. The energies involved in the optimization are rather simple to express. Although various expressions of the anisotropic Dirichlet energy have been proposed in 2D in the context of quasi-harmonic parameterizations [Gus02, ZRS05], we construct our 3D energies via matrix assembly using the language of discrete forms. Thus, $E^D_C(F)$ is expressed as:

$$E^D_C(F) \approx F^t\ A\ F \quad \text{with} \quad A = d_0^t \star^1_C d_0$$

where $d_0$ is the transpose of the signed vertex/edge incidence matrix (size $E$x$V$) of the mesh, and $\star^1_C$ is the Hodge star operator for the metric induced by $C$. We approximate this latter operator by the Euclidean diagonal Hodge star $\star^1$ *modulated* by the tensor $C$, resulting in the following diagonal matrix:

$$\forall i = 1 \ldots E,\ (\star^1_C)_{ii} = \frac{e_i^t\, C\, e_i}{e_i^t\, e_i}\ (\star^1)_{ii} \qquad \text{with:} \quad (\star^1)_{ii} = \frac{|e_i^*|}{|e_i|},$$

where $e_i$ is the $i^{\text{th}}$ (oriented) edge, $|e_i|$ is its length, and $|e_i^*|$ is the area of its dual Voronoi face. The value of the tensor $C$ on an edge is computed by averaging the tensor values at each vertex of the edge.

**Discrete Biharmonic Energy** For the biharmonic energy, the following (simplified) discretization performs adequately in practice (regularization will be added later through a data fitting term):

$$E^B(f) \approx F^t\ B\ F \quad \text{with} \quad B = (d_0^t \star^1 d_0)^2$$

where we used the same notations as above.

**Optimization Procedure** Using a Lagrange multiplier $\lambda$, we can now rewrite the constrained optimization as a maximization of the following functional:

$$E = F^t AF + \lambda(1 - F^t BF).$$

A necessary condition of optimality is $\partial E/\partial F = 0$, yielding:

$$AF = \lambda BF.$$

This expression expresses what is known as a generalized eigenvalue problem (GEP), where $F$ is an eigenvector of this GEP, and $\lambda$ is its corresponding eigenvalue. The solution to such constrained maximization is the eigenvector of the GEP corresponding to the largest eigenvalue.

*Proof:* Since the eigenvectors of a GEP form a basis, we can write a function $F$ as:

$$F = \sum a_\lambda F_\lambda,$$

where $F_\lambda$ is the eigenvector corresponding to the eigenvalue $\lambda$ (i.e., $AF_\lambda = \lambda BF_\lambda$) so that $F_\lambda^t BF_\lambda = 1$ and, for $\lambda_1 \neq \lambda_2$, $F_{\lambda_1}^t BF_{\lambda_2} = 0$. Therefore:

$$E_C^D = (\sum_\lambda a_\lambda F_\lambda)^t B(\sum_\lambda \lambda a_\lambda F_\lambda) = \sum_\lambda \lambda a_\lambda^2 \leq \sum_\lambda \lambda_{\max} a_\lambda^2 = \lambda_{\max}.$$

Since the energy is bounded by the max eigenvalue $\lambda_{\max}$ and this value is attained by $E_C^D(F_{\lambda_{\max}})$, we get: $F = F_{\lambda_{\max}}$.

So far our Voronoi-based variational approach requires no parameter-tweaking to provide a reconstruction of a point set. We can, however, extend this approach as follows: the matrix $B$ that contains the discrete biLaplacian can be modified in various ways to allow for more control over smoothness and/or interpolation: i) Data fitting: we can change the optimization results by adding a term that controls data fitting. We favor a value of 0 on the input points by adding to the constraint a fitting factor times the sum of the squares of the function values at input points. Changing the fitting factor will provide a controllable data fitting effect to our procedure; and ii) Splines-under-tension energy: instead of only using the biLaplacian, we can constrain the optimization over a unit ball defined by a linear combination of the Dirichlet and the biharmonic energies. It will allow for a better tradeoff between smoothness of the results vs. fitting of the normal directions, as it is tantamount to a splines-under-tension energy [SW90]. The matrix $B$ needs to be modified in order to implement these two generalizations as we detail next.

**Implementation** After reading in the input point set, we compute its Voronoi diagram and refine its dual Delaunay tetrahedral mesh by Delaunay refinement [CGA09, RY06] so as to obtain well-shaped tetrahedra. At each Steiner point added we set the tensor $C_i$ to the 3x3 identity matrix to remain agnostic as to the normal direction there. We then solve the generalized eigenvalue problem $AF = \lambda BF$ by turning it into a standard eigenvalue problem. This is achieved by precomputing a Cholesky factorization of $B$ using the TAUCS library [TCR05]; this results in a lower triangular matrix $L$ so that $B = LL^t$. We now rewrite the GEP as:

$$AF = \lambda LL^t F \Leftrightarrow L^{-1}AL^{-t}L^t F = \lambda L^t F \Leftrightarrow \begin{cases} L^{-1}AL^{-t}G = \lambda G \\ G = L^t F \end{cases}$$

We then employ the implicitly restarted Arnoldi iteration method from the ARPACK++ library [GS], with $L^{-1}AL^{-t}$ as the Arnoldi operator and by requesting the maximum eigenvalue. After convergence, we get the solution $F$ by solving $L^t F = G$, as it corresponds to the eigenvector corresponding to the maximum eigenvalue of the original GEP. This eigenvector defines a piecewise linear function over the tetrahedron mesh: we are now ready for contouring. To find which iso-contour to extract, we first evaluate the resulting function at all input points (not Steiner points), and pick the median value for contouring. In practice the median is more robust than the average value. For the final iso-contouring we use a surface mesh generation algorithm based on Delaunay refinement and filtering [BO05]. The latter is preferred to the accustomed marching cubes as it generates meshes with fewer triangles and a guaranteed quality over the shape of these triangles. Figure 2.7 illustrates outputs of the Delaunay-based mesh generation algorithm with three uniform sizing criteria.

### 2.2.3 Results

We first exemplify the similarities and differences between ours and the Poisson-based method performed on the same simplicial mesh (we implemented the Poisson equation of [KBH06] on a triangle mesh). As Figure 2.8 illustrates, the first point set (dense, noise-free) is correctly oriented using pole labeling (red/blue dots are poles), which allows us using the Poisson equation for reconstruction. Both methods lead to very similar reconstructed curves. When the sampling is sparser, the pole-based orienting process fails to provide the proper normal orientation, and the Poisson reconstruction reflects this, while the implicit function remains unchanged.

We illustrate in 2D the effects of tuning the parameters. $\mu_\Delta$ allows controlling the smoothness (Figure 2.9, top). $\mu_{\text{fit}}$ controls the fitting of the input points, and hence the separation of two components (Figure 2.9, middle). It also provides a way to increase the contrast of the implicit function

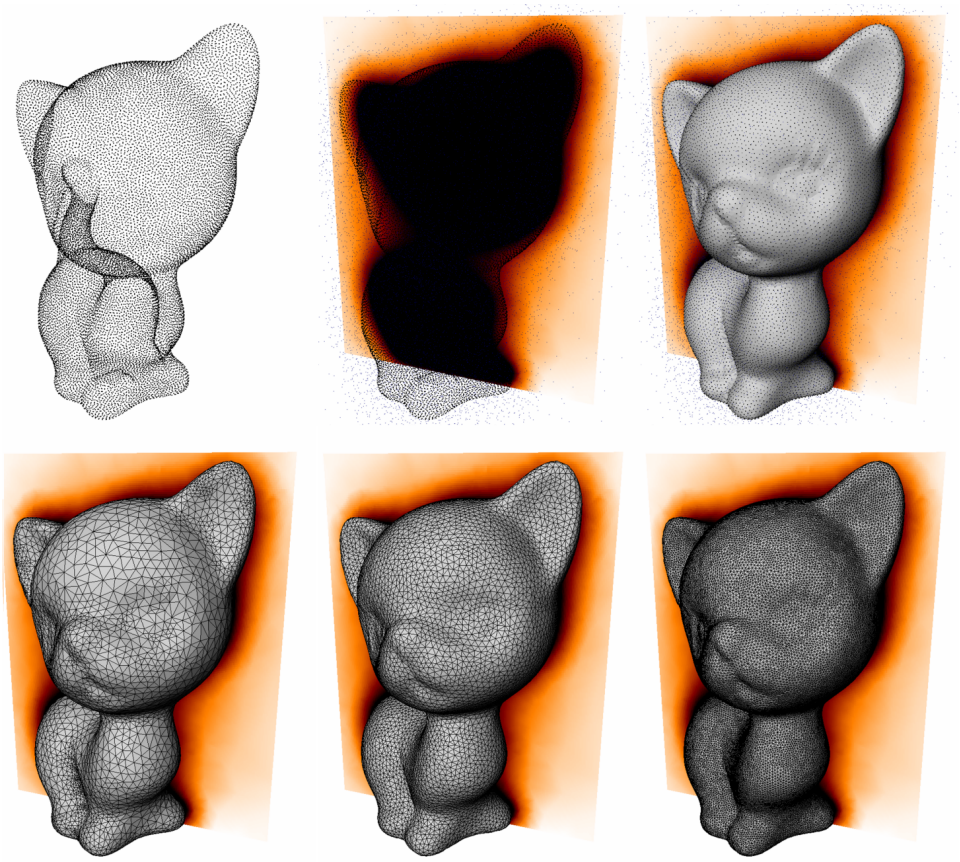


**Figure 2.7:** Kitten. Top: 20K input point set, Steiner points and implicit function, and shaded reconstructed surface obtained by marching-tetrahedra. Bottom: three output meshes at increasing resolutions.

for nested components (Figure 2.9, bottom left). A shape completion example from a sparse dataset, obtained with the splines-under-tension energy, is also shown (Figure 2.9, bottom/middle). Our last 2D example illustrates (Figure 2.10) the resilience to both sparsity and noise of the point set.

We also processed a number of 3D point sets issued from (laser range) scanners. We show in Fig.2.11 that our method applied to a raw, unoriented point set (206K points) recovers a similar surface to the Poisson reconstruction from [KBH06] for which additional normal information was provided. We note that even at octree depth 11, the Poisson-based mesh is comparatively over smoothed, mostly due to the interpolation of the normals onto the octree leaves. If such an over smoothing is desirable, we can either smooth the tensor field around the input points, or increase the smoothness by tuning two parameters (as demonstrated in Figure 2.9). Figure 2.12 illustrates the reconstruction from 250K points with noise and missing data.

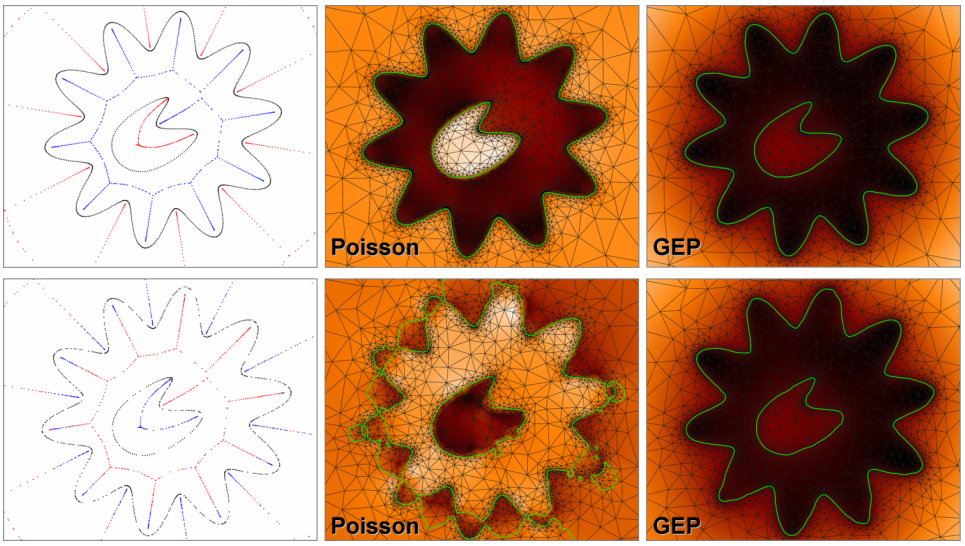



**Figure 2.8:** Comparison with Poisson reconstruction in 2D. (top) our approach is very similar to a Poisson reconstruction for dense point sets as the normal orientations can reliably be deduced; (bottom) for sparser/noisier unoriented datasets, local orientation becomes prone to errors, while our variational approach remains valid.

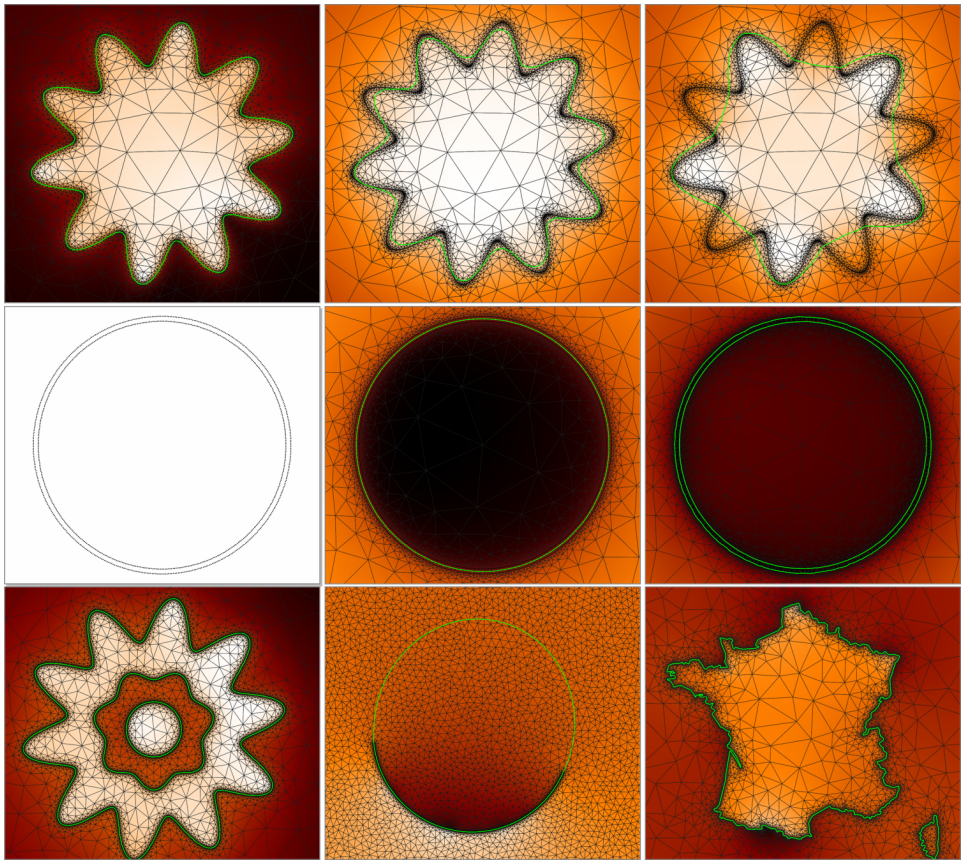


**Figure 2.9:** Reconstruction Parameters. (top) adjusting $\mu_\Delta$ (left to right) permits easy tuning of the resulting smoothness, (middle) while $\mu_{\text{fit}}$ controls the fit to the input point (middle & right), allowing to accurately separate and capture fine, nearby surface layers. (bottom) data fitting allows the reconstruction of nested components (left); smooth completion of sparse dataset is achieved using the splines-under-tension energy (middle); separate components of different geometric complexity can also be captured accurately.

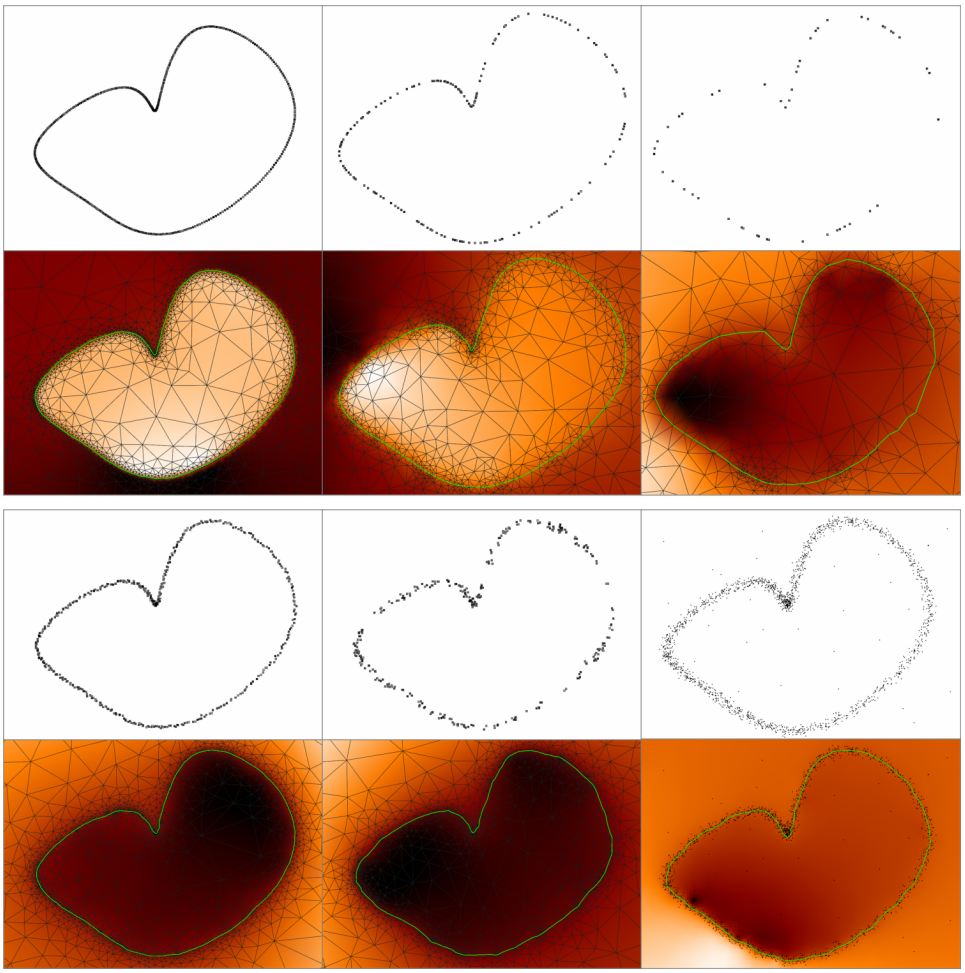


**Figure 2.10:** Noise and Sparsity Resilience. (top) our approach can handle from dense (left) to sparse (right) raw datasets without any parameter tweaking; (bottom) similarly, results degrade gracefully with noise.

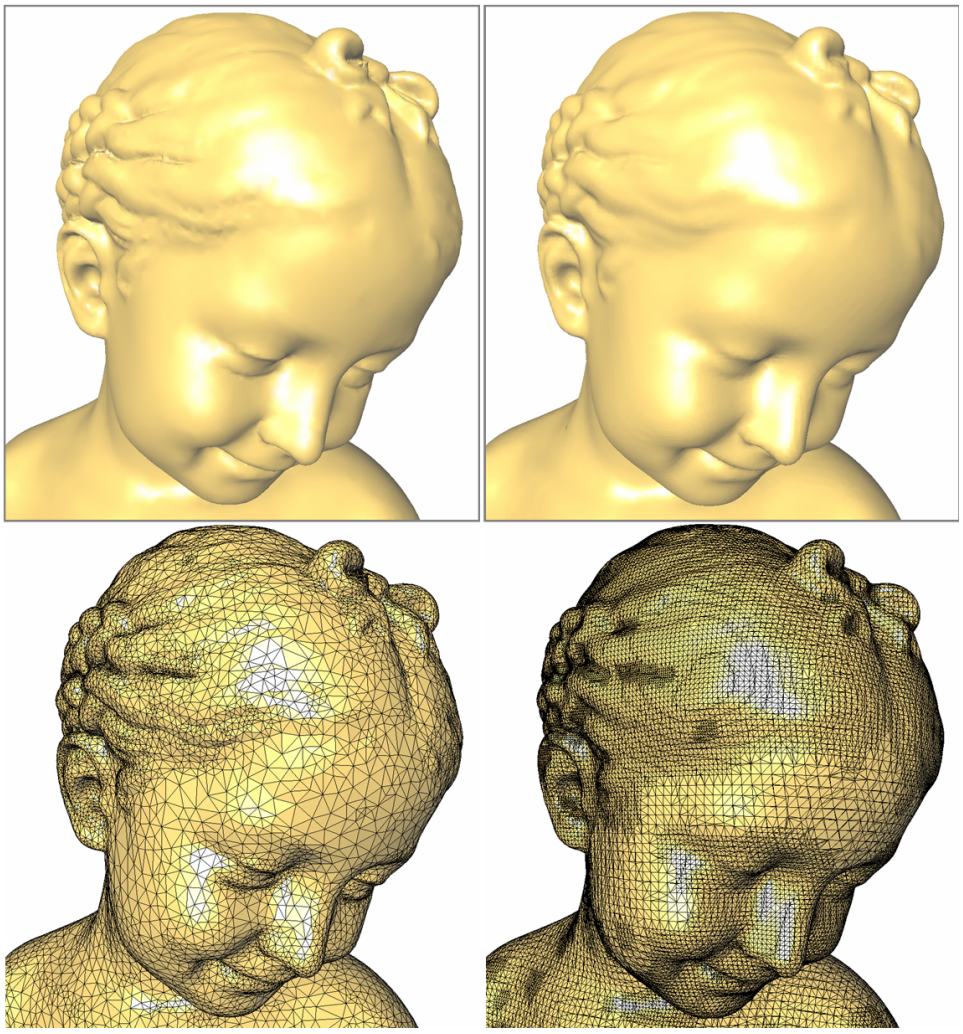


**Figure 2.11:** Comparison with Poisson reconstruction in 3D. Without any normal information, our method (left) results in a similar shape (albeit less over smoothed) to a Poisson reconstruction (octree depth 11) for which oriented normals were provided.

**Figure 2.12:** Sforza. Reconstruction from 250K points (marching-te reconstructed surface shown).

### 2.2.4 Summary

Our surface reconstruction approach is a mix between Voronoi-based methods (to estimate normals via PCA), implicit-based approaches (as an implicit function is globally optimized to allow for smooth approximation through iso-contouring), and spectral techniques as our optimization procedure involves solving a generalized eigenvalue problem. The main idea is to characterize the sampling quality through computing tensors as covariance matrix of unions of Voronoi cells, and to compute a scalar function which gradient best matches the principal component of these tensors. As the function is requested to be both aligned with the principal components and smooth, the function gradient is consistently oriented.

#### 2.2.4.1 Strengths

The key added value to this surface reconstruction approach is that it matches the results of oriented point set reconstruction techniques such as Poisson reconstruction, without the need for normal orientation. Since the normal orientation is delicate (if not impossible) on sparse or noisy point clouds, the latter property is a major advantage. Other distinctive features include the use of tensors in a generalized eigenvalue problem to balance fitting and smoothness based on data confidence, as well as the ability to trade data fitting for smoothness.

#### 2.2.4.2 Weaknesses

One important limitation of the normal estimation procedure is its limited resilience to outliers. Additionally, the weakness of the reconstruction algorithm is mostly a scalability issue, the bottleneck being the Cholesky factor as its memory requirement is high (e.g., 20M non-zero elements for the kitten). Out-of-core factorization is a viable option that we use for large models like *Bimba* and *Sforza*, but the overall timings consequently suffer (up to 25 minutes for 250K input points), and the super-nodal ordering still requires in-core execution. For larger datasets a 64-bit machine is currently indispensable to address these memory issues. We are searching for alternate scalable solutions to solve the generalized eigenvalue problem.

#### 2.2.4.3 Follow-ups

In a separate work not detailed here, we proved the convergence analysis of the Voronoi-PCA normal estimation scheme in 2D and for the noise-free case. By convergence, we mean proving that when the sampling increases the

principal component of the covariance matrix converges to the true normal to the curve. Such an analysis draws upon closed-form covariance matrices of Voronoi cells, the theory of $\epsilon$-sampling and the Gershgorin circle theorem for symmetric matrices. The key idea behind the proof is to exploit the known fact that the aspect ratio of the Voronoi cells increases while the sampling density increases [Dey06]. We then derive, from this aspect ratio and the Gershgorin circle theorem, a ratio between the eigenvalues of the covariance matrix and a decreasing angular deviation between the principal component of the covariance matrix and the true normal.

In a recent work Merigot et al. [MOG09] show that a small modification of the Voronoi-PCA approach is amenable to convergence proofs in presence of noise and when the points are sampled from a compact set (and not only from a smooth surface). In addition, it is shown that the whole curvature tensor, not just the normal direction, can be derived from the covariance tensor.

#### 2.2.4.4 Future Work

As future work we wish to study how the proposed normal estimation technique can be applied to dimension detection, along the line of [DGGZ03]. For applications where outliers are numerous we wish to elaborate upon a resilient normal estimation technique. As for surface reconstruction, we acknowledge the fact that the $L^2$ norm used here and in Poisson reconstruction is not ideal, as sharp transitions are overly penalized. This prevents reconstructing both sharp indicator functions and piecewise smooth surfaces. An $L^1$ norm would be worth investigating. As of today, the main remaining challenge is certainly the reconstruction of piecewise smooth surfaces. A common sequential approach consists of extracting the features first before reconstruction [DHOS07]. We believe that delegating the feature extraction to a global solving stage instead would be more robust as our approach already shows for normal orientation.

## 2.3 Surface Mesh Approximation

In this section we focus on the approximation of triangle surface meshes. Finding a concise, yet geometrically-faithful digital representation of a surface is at the core of several research themes in graphics. Given the verbosity of many 3D datasets (and in particular, of scanned meshes), reducing the number of mesh elements of a surface mesh while maintaining its geometric fidelity as best as possible is crucial for subsequent geometry processing. Ideally, each element should be made as efficient as possible by stretching it locally in order to fit a large area of the shape we wish to approximate, without introducing significant geometric error. This quest for geometric efficiency naturally raises the following question: given a 3D surface, a target number of face elements, and an error metric, what is the best geometric approximation of the object that one can find with this face budget? Or similarly, given a distortion tolerance, what is the smallest polygonal mesh approximant with a distortion lesser than the tolerance? Despite the fundamental aspects of this problem, its NP-hard nature has mostly thwarted the search for practical heuristics for finding such optimal meshes.

### 2.3.1 Related Work

We start with a background on approximation theory applied to functions, height fields and surfaces, and motivate an approach to shape approximation through variational partitioning.

**Functional Setting** Given a class of functions and a metric (usually $\mathcal{L}^p$ or $\mathcal{L}^\infty$), approximation theory has provided strong results on the best approximations with $n$ elements, be them piecewise-constant elements or higher order ones. Such results have given rise, for example, to optimal image encoders that give the Kolmogorov entropy bounds of the problem at hand [CDDD01]. However, most of these results cannot be easily extended to surfaces: the functional setting relies on a parameterization when comparing two functions. In the general case of two arbitrary surfaces, with no mapping known from one to the other, the functional metrics cannot be used directly.

**Height Fields** For the special case of height fields (where a trivial parameterization can readily be used), a few results are known about the optimality of piecewise-linear approximation at the asymptotic limit when the areas of the approximating elements (typically, triangles) vanish. It has been proven that with respect to the $\mathcal{L}^2$ metric, the triangulation that minimizes the

error of piecewise linear interpolation for a given large number of triangles must have an optimal triangles orientation aligned with the eigenvectors of the Hessian of the function, and an optimal size in each principal direction given by the reciprocal square root of the absolute value of the corresponding eigenvalue [Nad86]. Note that there is a subtle twist on hyperbolic regions, where there is not a unique optimal shape and direction, but a whole family of it; we will come back to this impediment in Section 2.3.2.4. Such an alignment and stretching of the triangles optimizes the efficacy of the mesh, i.e., minimizes the error per surface area. Results basically identical are also proven for an arbitrary $\mathcal{L}^p$ metric [Sim94]. A few results are also known for optimal approximation of the gradient error [DS91], or for bilinear approximation [D'A00], but again, they are only asymptotically valid. These different results are fairly narrow in scope: first, they are restricted to height fields; second, the triangulations are assumed to be only interpolating the height field at the vertices; and third, the asymptotic case does not help in designing a concrete surface approximation for a small number of triangles. Concrete bounds for the interpolation error and the gradient interpolation error for a non-infinitesimal triangle [She02a] offers much better insights, but still does not provide, to date, practical mesh generators with guaranteed approximation quality for a given number of elements. It is known that finding the piecewise-linear triangulation with a given number of vertices that optimally approximates a height field with respect to the $\mathcal{L}^\infty$ metric is a NP-hard problem [AS98].

**Arbitrary Geometry** Aside from the asymptotic results mentioned above, theoretical knowledge on optimal piecewise linear approximation of arbitrary surfaces is mostly unchartered territory despite the considerable amount of practical work on digital geometry. This lack of foundations and the intrinsic complexity of this problem explains the overwhelming usage of greedy algorithms, that can reduce the number of triangles but at the expense of an uncontrollable approximation error, or conversely, can guarantee a given approximation error criterion but at the expense of an uncontrollable number of triangles (with the noticeable exception of computational geometry papers proposing algorithms for convex and bivariate surfaces (see [AS98]), or about optimally-sparse $\epsilon$-sampling for accurate surface reconstruction [AB99, BO05]). The notion of error distance between two surfaces is, however, routinely used. Probably the most used metric in graphics, the $\mathcal{L}^p$ distance between a surface $X$ and an approximating surface $Y$ is the extension of the traditional $\mathcal{L}^p$ metric for the functional setting, and is often defined as:

$$\mathcal{L}^p(X,Y) = \left( \frac{1}{|X|} \iint_{x \in X} \|d(x,Y)\|^p dx \right)^{\frac{1}{p}}$$

$$\text{with: } d(x,Y) = \inf_{y \in Y} \|x - y\|$$

where $\|.\|$ is the Euclidean distance, while $|.|$ is the surface area. The extension of the $\mathcal{L}^\infty$ metric, called the Hausdorff distance, is naturally expressed as: $\mathcal{H}(X,Y) = \max_{x \in X} d(x,Y)$, but can be quite delicate to compute [ASCE02]. Notice that these definitions are sided: a true distance measure should add the symmetric version. However, in the context of surface approximation, the symmetric counterpart increases the complexity significantly as it contains an integral over the surface that we are basically looking for. Thus, it is discarded in practice (for instance in [HDD+93]).

**Partitioning** Many techniques have been specifically designed to exploit an object's local planarity, symmetry and features in order to optimize its geometric representation. While most simplification approaches try to provide an $\epsilon$-approximation with respect to various metrics, rare are the methods that systematically target a minimum distortion error for a given budget of linear mesh elements. A powerful solution to mesh simplification is to greedily cluster geometric elements, creating in effect a partition of the original object. Mesh decimation provides an elegant approach to such a partitioning, through greedy and repeated collapsing of mesh elements [Hop96, KLS96, GH98, LT98]. However, and although some of the metrics used for clustering can be proven asymptotically optimal (i.e., for infinitesimal triangles) for the $\mathcal{L}^2$ metric [HG99], the greedy nature of decimation is prone to result in suboptimal meshes. A similar statement is true for another (almost dual) family of approaches [MYV93, IIY+99, She01, SSGH01, GWH01, LPRM02, MPS+04] which gather faces in a set of characteristic regions to provide a succinct, higher-level description of the geometry.

**Global optimization** Contrasting with the previous greedy techniques, Hoppe et al. [HDD+93] proposed to cast mesh simplification as an optimization problem. With an energy functional measuring deviation from the input mesh, they show that optimizing the number of vertices as well as their positions and their connectivity captures the curvature variations and features of the original geometry. Although their functional is mostly a point-to-surface Euclidean distance, they report excellent results for mesh simplification. Despite a spring force restricting the anisotropy of the results, such optimization techniques often result in irregular meshes for which geometric efficiency (i.e., how many faces are needed to capture geometry) is particularly good. While other methods use some form of local mesh optimization (see, for instance, [BVL02, OBP03]), this subject remains marginally studied to date, most certainly because of the difficulty of the task at hand: the mere size of the search space seems to hamper efficiency. Although good practical approaches for shape approximation have been proposed in the past, only

marginal work has been devoted to global minimization of approximation error with respect to a chosen metric.

### 2.3.2 Approach

Our strategy to design succinct meshes is entirely error-driven and uses a discrete, variational shape approximation method: we distribute mesh elements over the original geometry by targeting both a best local fit and a least global error, without resorting to any estimation of differential quantities nor parameterization. To achieve these goals, we define geometric proxies as a best-fit geometric surrogate to effectively shed topological issues; and we define proper shape error metrics to measure how well a proxy fits a piece of geometry. Finally, we cast the approximation problem as a variational partitioning (see overview Figure 2.13).

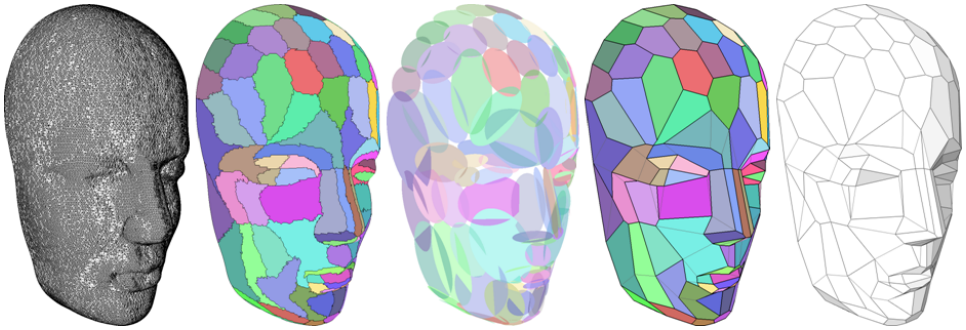


**Figure 2.13:** Variational Shape Approximation: Through repeated error-driven partitioning (left), we find a set of geometric proxies (represented as ellipses, center) providing a concise geometric description of an input surface (62K triangles) by capturing the anisotropy of the initial model; notice the presence of disks on near-spherical regions, and stretched ellipses on near-parabolic regions. These proxies are then used to construct an approximating polygonal mesh (right).

#### 2.3.2.1 Shape Approximation

We reformulate the problem of surface approximation by introducing the notions of shape proxies and variational partitions.

#### 2.3.2.2 Variational Partitioning and Proxies

Agarwal and Suri [AS98] mentioned that the problem of functional approximation can be cast as a geometric partitioning one. This idea of clustering points or faces of a 3D objects into a partition to help approximate the

geometry has already been used in graphics [HUHJ01, PGK02], and particularly for mechanical parts [IIY+99, She01], where clear-cut features make the partitioning easier. After all, an approximating face is nothing but a surrogate, linear approximant for a set of original clustered faces that share, on average, similar geometric characteristics. Therefore, clustering faces into a partition with $k$ regions appears to be a natural way to resample geometry (see Figure 2.14). Although we base the geometric approximation on partitioning through clustering too, we iteratively seek a partition that minimizes a given error metric. We now define some terminology.

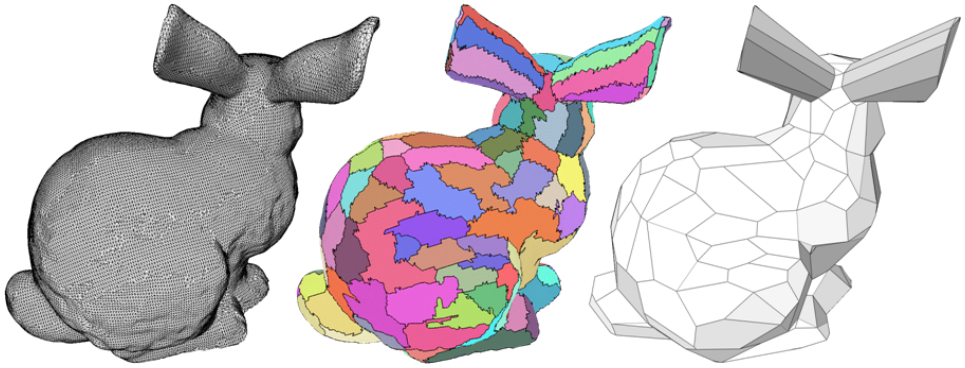


**Figure 2.14:** Bunny: (left and center) $\mathcal{L}^{2,1}$-optimized geometric partitioning; (right) Anisotropic polygonal mesh deduced from the partition. Notice the anisotropy on the ears.

**Partition and Proxies** Each region $\mathcal{R}_i$ of a partition $\mathcal{R}$ can be summarily represented to first order as an "average" point $X_i$ and an "average" normal $\mathbf{N}_i$ (the word average is here used in a broad sense; it will be made clear later when we define a metric with respect to which these averages will represent the best local linear fit). We will denote such a local representative pair $P_i = (X_i, \mathbf{N}_i)$ a *shape proxy* of the associated region. Thus, for any given partition of a surface in $k$ regions, we associate a set $P$ of shape proxies $P = \{P_i\}_{i=1..k}$ that coarsely approximate the whole geometry. At this point, it is worthwhile to point out that a $k$-partition, in effect, defines a dual meta-mesh of the original: the proxies define $k$ dual meta-faces (obtained through clustering of original faces), and the connectivity of the $k$ regions of the partition induces the topology of this dual mesh. Now, for this approximant to be relevant, we need to evaluate the quality of the partition in order to find a partition with optimal quality.

#### 2.3.2.3 Metrics on Proxies

Defining an appropriate error metric is a key ingredient in approximation. As mentioned earlier, the $\mathcal{L}^2$ or Hausdorff metrics are often used when comparing two triangulated surfaces. In our case, we want to measure the geometric relevance of a proxy set for a given surface. It allows us scoring a partition in terms of how well it approximates a surface.

#### 2.3.2.4 $\mathcal{L}^2$ Metric for Proxies

We can extend the notion of $\mathcal{L}^2$ distance. Given a region $\mathcal{R}_i$ and its associated proxy $P_i = (X_i, \mathbf{N}_i)$, we denote $\Pi_i(.)$ the orthogonal projection of the argument on the "proxy" plane passing through $X_i$ and of normal $\mathbf{N}_i$; the $\mathcal{L}^2$ metric is then:

$$\mathcal{L}^2(\mathcal{R}_i, P_i) = \iint_{x \in X} \|x - \Pi_i(x)\|^2 dx \tag{2.1}$$

This formula (square root and area normalization removed) measures the integral of the squared error between the region $\mathcal{R}_i$ and its linear proxy $P_i$. Notice that we integrate the $\mathcal{L}^2$ distance over the surface so as to make the optimization robust to irregular sampling rate of the input geometry (see Figure 2.15).

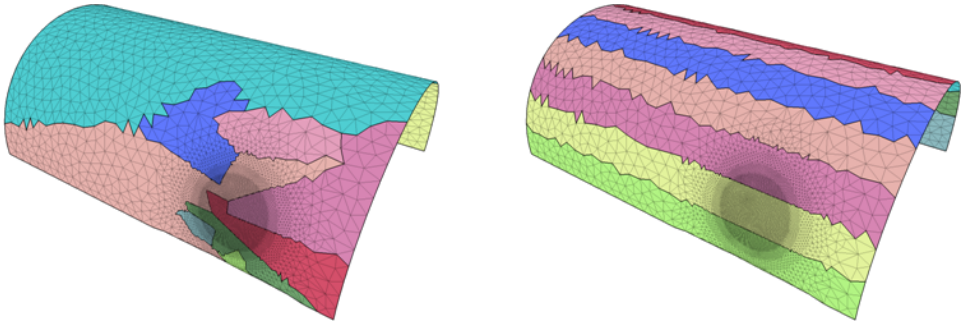


**Figure 2.15:** $\mathcal{L}^2$-optimized partition for a highly non-uniform input mesh (notice the disk-shaped region with refined triangles). The sampling irregularity severely distorts the partitioning if point-based covariance matrices are used (left), while the triangle-based covariance matrices (right) provide the expected polygonal approximation, capturing the true geometry.

As proven for elliptic areas in the asymptotic limit [Nad86], a $\mathcal{L}^2$-optimal approximation of a surface tends to create elements taking advantage of local anisotropy by being stretched in the minimum curvature direction with an aspect ratio of $\sqrt{|\kappa_{\max}/\kappa_{\min}|}$. This stretching along the minimum curvature

direction makes very good use of the local shape of the object. However, in the hyperbolic case, there is no unique optimal shape and alignment. Since we are targeting a variational approach, this non-unique optimality is worrisome: a minimization algorithm can randomly jump around in the null space of the functional, resulting in either endless oscillations, or in technically optimal but noise-prone results. To circumvent this issue, we look for another metric next.

**Introducing $\mathcal{L}^{2,1}$ as a Shape Metric** The $\mathcal{L}^2$ metric tries matching geometry through approximation of the geometric position of the object in space. However, the normal field is fundamental in the way the visual system interprets the object's shape: normals govern lighting effects such as diffusion, specularity, as well as curvature lines and silhouettes; a smooth normal field defines a smooth shape, while normal discontinuities indicate features. Moreover, there is evidence that our visual perception is actually more sensitive to changes in normals rather than in changes in positions: this property has been used in compression for instance, where quantization noise can be hidden in the low-frequency errors as it induces less visual impact [SCOT03]. In the functional setting, Shewchuk [She02a] advocates that one should focus on getting good bounds on the gradient interpolation error, as these are more difficult to control: the functional interpolation errors can always be improved through refinement, whereas such a refinement may not improve the gradient interpolation quality. In fact, approximating a function well does not mean that its gradient will also get approximated [Fu93]: there are well known examples (Schwarz's Chinese lantern for instance) of triangulated surfaces converging to a smooth surface for the Hausdorff metric, but with a surface area and normals diverging. However, as hinted by the Poincaré-Wirtinger-Sobolev inequality, controlling the upper bound of the norm of the gradient interpolation error allows to also bound the norm of the interpolation error. Given the cogent body of evidence in favor of a normal-based measure of distortion, we introduce a shape metric that we denote $\mathcal{L}^{2,1}$, as it is based on a $\mathcal{L}^2$ measure of the normal field:

$$\mathcal{L}^{2,1}(\mathcal{R}_i, P_i) = \iint_{x \in X} \|\mathbf{n}(x) - \mathbf{n}_i\|^2 dx \tag{2.2}$$

We show in Section 2.3.2.9 that this metric is numerically superior to $\mathcal{L}^2$ in several ways: i) The anisotropy of the surface is better exploited, since the asymptotic aspect ratio of an optimal element is in $|\kappa_{\max}/\kappa_{\min}|$, therefore largely superior to the asymptotic $\mathcal{L}^2$ behavior. Moreover, we show that there is a unique optimal shape and alignment in the limit for all (non-isotropic) surface types, be it parabolic, elliptic, or hyperbolic. The difference in results with the $\mathcal{L}^2$ metric is noticeable (see Figure 2.16); and

ii) Finding the best normal proxy is as simple as averaging the normals over the associated region. We do not have to compute a covariance matrix and thus save a significant amount of computations compared to $\mathcal{L}^2$. Finally, notice that the asymptotic results are in agreement with the optimal case of the gradient approximation mentioned in [She02a, DS91].

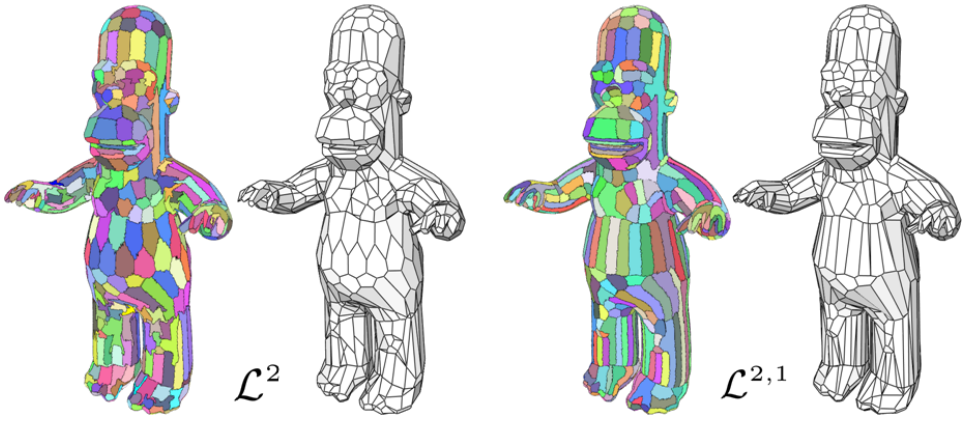



**Figure 2.16:** Homer: This character illustrates the effect that an error metric can have on approximation. While $\mathcal{L}^2$ (left) and $\mathcal{L}^{2,1}$ (right) behave similarly on near-spherical regions such as the top of the head, the belly and mouth regions are very different in each case.

**Optimal Shape Proxies** We can define what we mean by an optimal partitioning of an arbitrary surface: Given an error metric $E$ (either $\mathcal{L}^2$ or $\mathcal{L}^{2,1}$), a desired number $k$ of proxies, and an input surface $S$, we call optimal shape proxies a set $P$ of proxies $P_i$ associated to the regions $R_i$ of a partition $\mathcal{R}$ of $S$ that minimizes the total distortion: $E(\mathcal{R}, P) = \sum_{i=1..k} E(\mathcal{R}_i, P_i)$. In other words, the set of proxies is optimal with respect to an error metric if it minimizes the total approximation error over the possible sets of proxies of same cardinality. Of course, in practice we cannot hope to find the global minimum in a reasonable time. However, we set up our shape approximation as a discrete, variational partitioning of the initial faces so that we can apply a simple discrete clustering algorithm.

#### 2.3.2.5 Optimizing Shape Proxies

Given an error metric $E$, a number $k$ of proxies, and an input geometry $S$ of arbitrary size and topology, we wish to find a partitioning $\mathcal{R}$ of $S$ in $k$ disjoint, connected regions and its respective set $P$ of optimal proxies that minimizes (or nearly minimizes) $E(\mathcal{R}, P)$. Because in practice the input geometry is triangulated, we can consider this mesh as a discrete collection

of faces: the problem is then cast into optimal discrete clustering, for which simple algorithms exist.

**Background on Lloyd's Clustering Algorithm** Clustering a set of discrete points involves dividing them into non-overlapping regions (or clusters), where points belonging to a region are closer by some measure of proximity to one another than to points in other clusters. Every regions can be characterized by a single, "average" center, and the set of all $k$ regions is called a $k$-partition. The Lloyd algorithm is a deterministic, fixed point iteration that provides such a partitioning [Llo82]. The idea is as follows: after defining $k$ random centers, all the data points are partitioned into $k$ regions by assigning each point to its nearest center. Then, the algorithm updates the centers to be the barycenters (centroids) of their associated regions before starting a new partition with these new centers. This process is repeated until a stopping criterion is met. It can be proven that such an algorithm (sometimes referred to as k-means clustering) tries minimizing a cost function $E$ based on how tightly each region is packed and how well separated the different clusters are: the functional $E$ defined by a set of $N$ points $\{X_j\}$ and $k$ centers $\{c_i\}$ is: $E = \sum_{i \in 1..k} \sum_{X_j \in \mathcal{R}_i} \|X_j - c_i\|^2$. For such a functional, Lloyd's algorithm always converges in a finite number of steps, since each step reduces the energy $E$: the partitioning stage minimizes $E$ for a fixed set of centers $c_i$, while the fitting stage minimizes $E$ for a fixed partition. Lloyd's algorithm is widely used as it manages to find very good minima (even if sometimes not global) despite its simplicity and ease of implementation [Hau01, KMN+02, OBA+03b, KT03, SAG03, SWG+03]. Therefore, if we are able to adapt this algorithm to our context, we should be able to quickly produce a low-distortion partitioning and a set of geometric proxies that closely approximate any input geometry.

**Algorithm At a Glance** Lloyd's method hinges on the two phases of *partitioning* and *fitting*, repeated alternately to drive the total energy down. Paralleling this process, we present a simple extension of Lloyd's algorithm to variational, geometry-driven partitioning that includes the following steps: i) Geometry Partitioning: In order to create a partition of an arbitrary non-flat triangulation, we use a error-minimizing region growing algorithm that will segment the object in non-overlapping connected regions; ii) Proxy Fitting: Once a partition is found, we compute for each region an optimal local representative, the proxy (see Section 2.3.2.2). These geometric proxies, that minimize the distortion error for a given partition, are nothing else but the equivalent of the centroids in Lloyd's algorithm.

**Nomenclature** We now refer to the input surface as $S$, its current partition as $\mathcal{R}$, its $k$ regions as $\mathcal{R}_i$, and their current respective proxy as $P_i = (X_i, \mathbf{N}_i)$. The distortion error is referred to as $E$, and can represent either the $\mathcal{L}^2$- or $\mathcal{L}^{2,1}$-based error defined in Section 2.3.2.3.

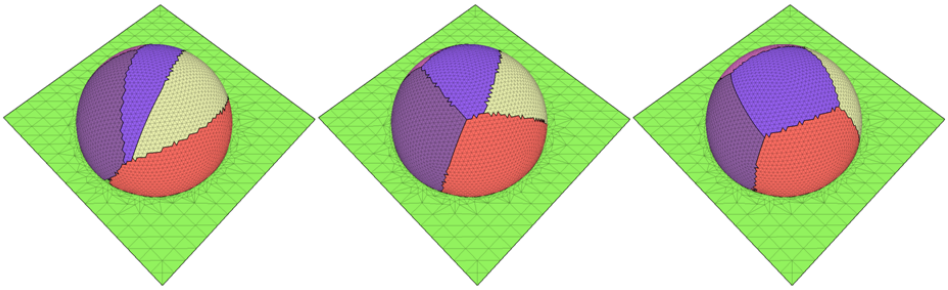


**Figure 2.17:** Half-sphere on plane: (left) random initialization of a 6-partitioning; (center) after one iteration of optimization, the regions self-organize; (right) after 5 iterations, the regions settle.

#### 2.3.2.6 Geometry Partitioning

Knowing a current set of proxies $P$, we wish to update the partition $\mathcal{R}$ while trying to minimize the distortion error $E(\mathcal{R}, P)$ in the process. We perform this k-proxy clustering as follows.

**Initial Seeding** For each region of the previous partition, we first find the triangle $T_i$ of $S$ that is the most similar to its associated proxy. This is easily achieved by visiting each current partition region $\mathcal{R}_i$, and by going once through all its triangles to find the one with the smallest distortion error $E(T_i, P_i)$. In order to bootstrap the algorithm, we add one region at a time, perform a partitioning, then proceed by adding a new region at the triangle of maximum error with respect to the region it belongs to (this is reminiscent of the farthest point strategy); no fitting between two floodings is necessary, as the proxy values are directly picked from the seed triangles' barycenters and normals.

**Distortion-minimizing Flooding** Once these seed triangles are found, we wish to grow a region out from them, in order to find a new, better partition. Just like in Lloyd's algorithm, we wish to cluster together only faces that are close (i.e., with a low error distortion) to the proxy. Therefore, for each seed triangle $T_i$, we insert its three adjacent triangles $T_j$ in a priority queue, with a priority equal to their respective distortion error $E(T_j, P_i)$, and we add an additional tag indicating the label $i$ of the proxy they are

being tested against (a triangle can therefore appear up to three times in the queue, with different tags and priorities). The region-growing process is then performed by repeatedly popping triangles with least priority until the priority queue is empty. For each triangle popped out from the queue, we check its proxy assignment: if it has already been assigned to a proxy, we do nothing and go to the next triangle in the queue; otherwise, we assign it to region of the proxy indicated by the tag, and push the (up to two) unlabeled incident triangles in the queue along with the same tag. When the priority queue has been emptied, each triangle has been assigned to a proxy: we therefore have a new partition. The use of a global priority queue ensures that the regions will have grown anisotropically, in the sense that the direction of growth of each region will be the one with smaller local slope. Such growing process ensures connected and non-overlapping regions as required, and is rapid ($Nlog(N)$ complexity).

#### 2.3.2.7 Proxy Fitting

Once we have found a new partition $\mathcal{R}$ over the surface $S$, we now wish to update the respective proxies $P_i = (X_i, \mathbf{N}_i)$ in order for them to be the best representative of their associated region $\mathcal{R}_i$ freshly updated (iterative partitioning is exemplified by Figure 2.17). Notice that, for the given partition $\mathcal{R}$, this procedure will find the set of proxies that minimizes the total distortion error $E(\mathcal{R}, P)$. This minimization is done as follows. For the $\mathcal{L}^2$ metric, $X_i$ is the barycenter of the region $\mathcal{R}_i$ while $\mathbf{N}_i$ is the direction (the sign does not matter) indicated by the eigenvector associated with the smallest eigenvalue of the covariance matrix of the region—i.e., the proxy is the least-square fitting plane traditionally found with Principal Component Analysis. For the $\mathcal{L}^{2,1}$ metric, proxy normal is the area-weighted average of the triangles' normals of the region; the point $X_i$ is chosen to be the barycenter of the region.

In order to make the variational partitioning more flexible, we let the user not only pick the desired number of proxies at any time, but we also allow interactive, incremental insertion and deletion of proxies. The insertion is done by finding the current region with maximum total distortion, and within it, we pick the triangle with worst distortion error as the initial seed for the next flooding (this is yet another farthest-point sampling heuristic); this will add a new region and proxy in the most needed part of the object. Similarly, we allow the incremental deletion of a region. We select the region to be deleted as follows: for each pair of adjacent regions, we simulate a merging of the two regions and compute the resulting distortion with the

new best fitting proxy; the pair of regions achieving the smallest distortion when merged are then replaced with a single one, deleting a proxy in effect.

**Teleportation** In the course of finding a better distortion minimum, the algorithm can find itself stuck in a local minimum. Typically, this can happen on a flat region: if a region happens to be encircled by other regions with similar proxies, it may be locally stuck in this minimum configuration as this position prevents it to roam on the surface and find better positions. Yet it is clearly a waste of efficiency to leave this region as is. We have therefore implemented a region teleportation procedure to give a region the chance to tunnel out of a local minimum, similar in spirit to [BH96, LT00]. At regular intervals during the clustering process, we simply force a region deletion as described above, immediately followed by a region insertion: the effect of this simple two-step operation is to remove a region stuck in a local minimum, and teleport it where it is most needed. In practice, it is better to first test if this operation is worth it: we use a heuristic that test whether the error added by a (simulated) deletion is smaller than half of the error of the worst region. If this test fails, no teleportation is necessary.

**Convergence** We cannot guarantee global convergence of this variational approach, although we observe in practice a very good behavior: the proxies start settling down after a few iterations, or oscillate around extremely similar distortion errors (see the error as a function of the number of iterations in Figure 2.18). Convergence is, however, guaranteed for convex objects for the $\mathcal{L}^{2,1}$ norm, since it amounts to the well-known k-means (area-weighted) clustering of the discrete normals on the image of the Gauss map. Furthermore, convergence would also be guaranteed for arbitrary surfaces if one was to relax the connectedness of the regions in the partition; however, having proxies that correspond to disconnected patches of surface is not very relevant.

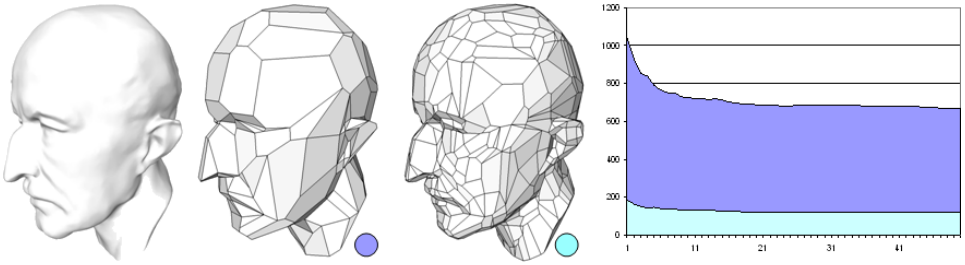



**Figure 2.18:** Max Planck: For the two optimized approximations (130 and 300 proxies resp.), we show the associated curves of the $\mathcal{L}^{2,1}$-distortion error as a function of the number of Lloyd's iterations; as expected, a few iterations suffice to reach a much reduced distortion error.

#### 2.3.2.8 Meshing the Proxies

Now that we found a nearly-optimal partitioning, its proxies capture the essence of the input geometry. Additionally, the adjacency graph of the partition defines the connectivity of a mesh as well. We create an anchor vertex at every original vertex where three or more regions meet. The spatial position of these anchor vertices is determined as follows: for each neighboring proxy of an anchor, we compute the projection of the associated vertex from which the anchor was created onto the proxy (i.e., its ideal position for this proxy); and we average these projections. We can now add edges between the anchor vertices by simply visiting each region boundary. Although these so-constructed edges and vertices are topologically sufficient, they may approximate the region's boundary rather coarsely. We thus have recourse to a recursive chord-length subdividing algorithm. If **a** and **b** are during recursion two anchor vertices linked by an edge separating proxy $P_i$ and $P_j$, we visit all the original vertices of the associated boundary arc, find the largest distance $d$ from these vertices to the edge $(\mathbf{a}, \mathbf{b})$, and add an anchor vertex there.

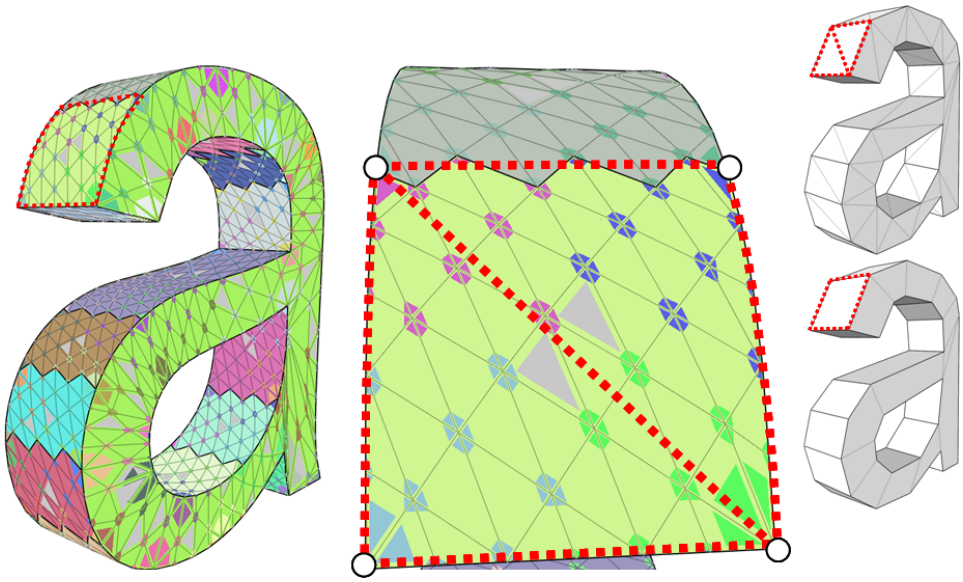


**Figure 2.19:** Discrete Constrained Delaunay Triangulation: Flooding the mesh from the anchor vertices (solid dots) creates triangles (light grey, left and center) whose three corners have different colors. Each of these triangles generates a meta-triangle during meshing. A final edge-removal pass provides a $\mathcal{L}^{2,1}$-polygonal model (right, bottom).

With the anchor vertices and edges defined, we already have a polygonal mesh. However, the polygons have no guarantee of being flat nor convex. The triangulation of this initial graph is done through a "discrete" Con-

strained Delaunay triangulation (CDT) to render the process robust: we create Delaunay-like triangles within each region, while constraining the existing anchor-based edges to be part of the final triangulation. To achieve this goal, we have recourse to a flooding algorithm, very similar to the multi-source Djisktra's shortest path algorithm with an edge weight equal to its length, and for which the sources are the anchor vertices. In a first step, we only flood the boundary of a region so that every vertex on it is colored depending on the closest anchor vertex: this enforces the constrained triangulation by forcing the boundary to be in it. We then start a flooding of the interior of the region, coloring the vertices also according to their closest anchor vertex. The extraction of the final triangles is now straightforward. We look at every triangle of the input mesh whose three vertices have distinct colors: each of them corresponds to a triangle in the final triangulation, emanating from the anchor vertices indicated by the three colors. A summary of this process is depicted in Figure 2.19.

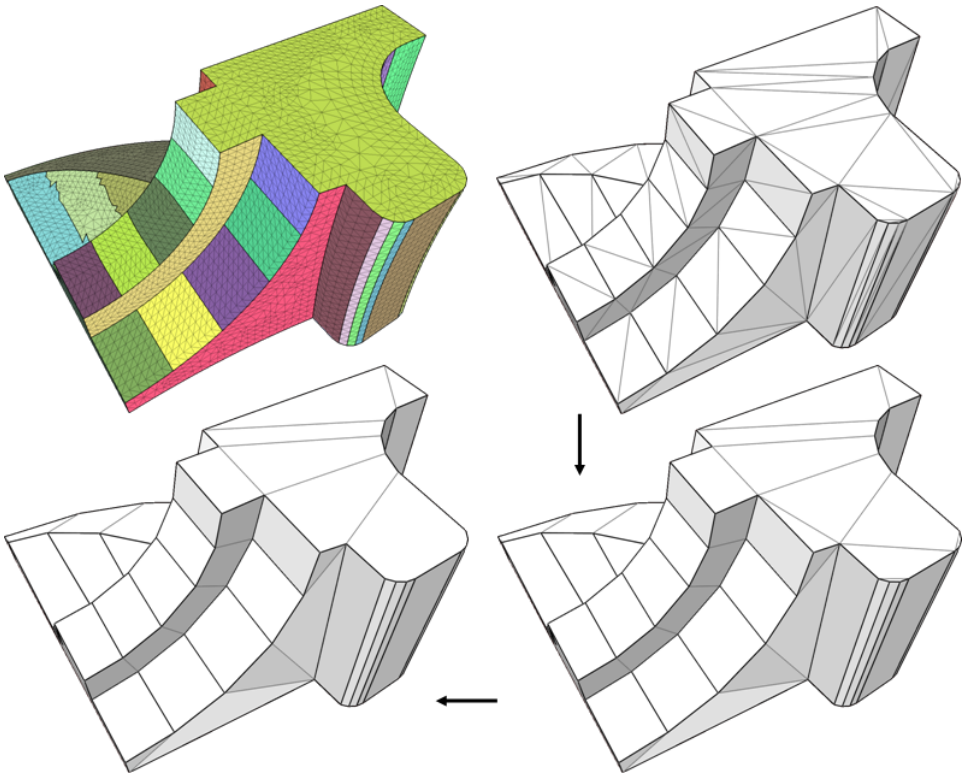
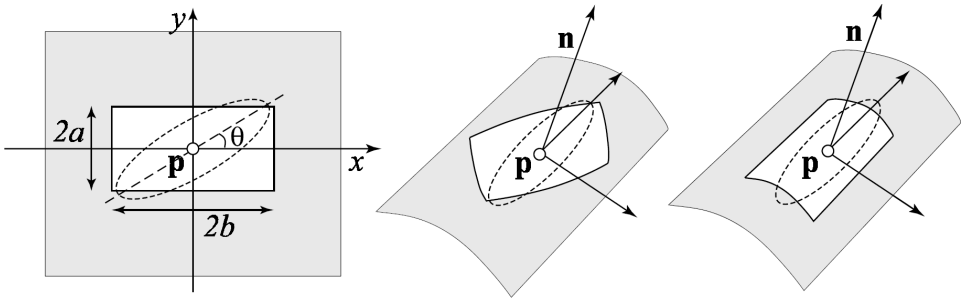


**Figure 2.20:** Generation of a polygonal model: triangle mesh obtained via CDT (top-right); creation of well-shaped quads (bottom-right), then polygons (bottom-left) by iterative edge removal.

We perform a final pass over the triangulation to remove the edges that do not contribute to the shape. First, we try to make as many nicely-

shaped quads as possible: we look at edges that can be safely removed (i.e., that produce no normal flips); we sort them by a score linked to the well-shapedness of every candidate quad [Peb02]; finally, we go down the list and remove the edges creating the best quads first, until the list is empty. Second, we perform a second pass in order to create convex polygons when possible. A close-up on the remeshed Fandisk model in Figure 2.20 exhibits the type of polygon mesh obtained.

#### 2.3.2.9 Asymptotic Behavior of the $\mathcal{L}^{2,1}$ Metric




(A.1) (A.2) (A.3)

Consider an arbitrary surface $\mathcal{S}$. Let $\mathcal{R}$ be a small rectangle of dimension $2a \times 2b = |\mathcal{R}|$, and such as $\mathcal{R}$ is tangent in its center to the surface $\mathcal{S}$ at a point $\mathbf{p}$. The normal $\mathbf{n_p}$ at $\mathbf{p}$ is therefore also normal to $\mathcal{R}$. The only parameters that are not determined are $a$, $b$, and the angle $\theta$ between the minimum curvature direction and the side of $\mathcal{R}$ (see Figures A.1 and A.2). Then we have [Gra98]:

$$\mathbf{n}(x,y) \simeq \mathbf{n_p} + \mathbf{H}\begin{pmatrix} x \\ y \end{pmatrix}$$

.

$\mathbf{H}$ is the (symmetrical) Hessian matrix. The average normal $\mathbf{N}$ is therefore $\mathbf{N} = \mathbf{n_p}$, and the $\mathcal{L}^{2,1}$-based error $E$ is expressed as:

$$\begin{aligned} E &= \iint_{\mathcal{R}} \|\mathbf{n}(x,y) - \mathbf{n_p}\|^2 \mathrm{d}x\ \mathrm{d}y = \iint_{\mathcal{R}} (x\ y)\ \mathbf{H}^t\ \mathbf{H} \begin{pmatrix} x \\ y \end{pmatrix} \mathrm{d}x\ \mathrm{d}y \\ &= \iint_{\mathcal{R}} (x\ y)\ \mathbf{Q} \begin{pmatrix} x \\ y \end{pmatrix} \mathrm{d}x\ \mathrm{d}y = \int_{-a}^{+a}\int_{-b}^{+b} (x\ y) \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \mathrm{d}x\ \mathrm{d}y \\ &= \frac{4ab}{3}(Q_{11}\ a^2 + Q_{22}\ b^2) \end{aligned}$$

Notice that $\mathbf{Q} = \mathbf{H}^t\mathbf{H} = \mathbf{H}^2$ is by definition always symmetric positive, even if $\mathbf{H}$ is not positive (like in hyperbolic regions). We now define the efficiency $f = E/|\mathcal{R}|$ as the ratio of error covered by area unit [Sim94]. Obviously, we wish $f$ to be minimum. In our case, we can rewrite:

$$f = \frac{1}{3}(Q_{11}\, a^2 + Q_{22}\, b^2).$$

Now if we try to optimize, using a Lagrange multiplier $\lambda$, the efficiency as a function of $a$ and $b$ under the constraint that the area $ab$ is constant, we get the following linear system:

$$\frac{2}{3}\begin{pmatrix} Q_{11}\, a \\ Q_{22}\, b \end{pmatrix} = \lambda \begin{pmatrix} b \\ a \end{pmatrix}.$$

We then find that the optimal dimensions of $\mathcal{R}$ (to which our minimization will converge to in the limit) is: $a = \mu/\sqrt{|Q_{11}|}$, $b = \mu/\sqrt{|Q_{22}|}$, $\mu$ constant. For this optimal rectangle, we have: $f = \frac{2|\mathcal{R}|}{12}\sqrt{Q_{11}Q_{22}}$. However, notice that $det\ \mathbf{Q} = Q_{11}Q_{22} - Q_{12}^2 \geq 0$ for any $\theta$. The efficiency $f$ is therefore best when $Q_{12} = 0$: $\mathbf{Q}$ is then diagonal, which means that $\mathbf{H}$ is also diagonal and thus, $\theta = 0$. As a consequence, we will converge towards a quadrangle that is aligned with the principal curvature (since $\theta = 0$ - see Figure A.3); and has a side ratio of $a/b = \sqrt{\frac{Q_{11}}{Q_{22}}} = |\frac{H_{11}}{H_{22}}|$, i.e., of ratio $|\kappa_2/\kappa_1|$ since $\mathbf{H}$ is diagonal in the optimal configuration.

### 2.3.3 Results

Although the $\mathcal{L}^2$ metric provides good approximations in general, the $\mathcal{L}^{2,1}$ results are in agreement with what we would have expected from a good segmentation of geometry, and often capture more details (see Figure 2.21). It also compares favorably with the QEM approach [GH98], see Figure 2.23. Figure 2.22 shows a more complex example on a 150K triangle mesh and 800 proxies.

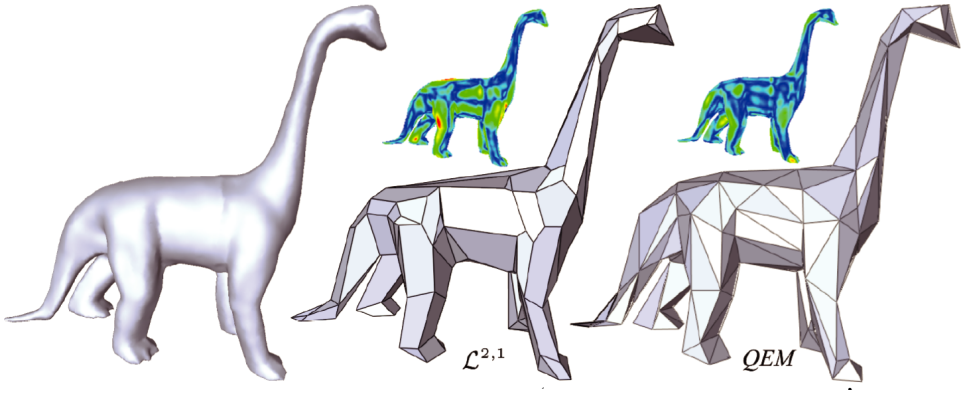



**Figure 2.21:** (Left) Dinosaur model; (Center) $\mathcal{L}^{2,1}$-approximation, with a false color version of the Hausdorff distance to the original mesh. (Right) Results for QEM [GH98] with same number of edges. Notice that our approach based on normals reproduces the "highlights" (see the neck), with a symmetric Hausdorff error 18% smaller (as measured by [ASCE02]).

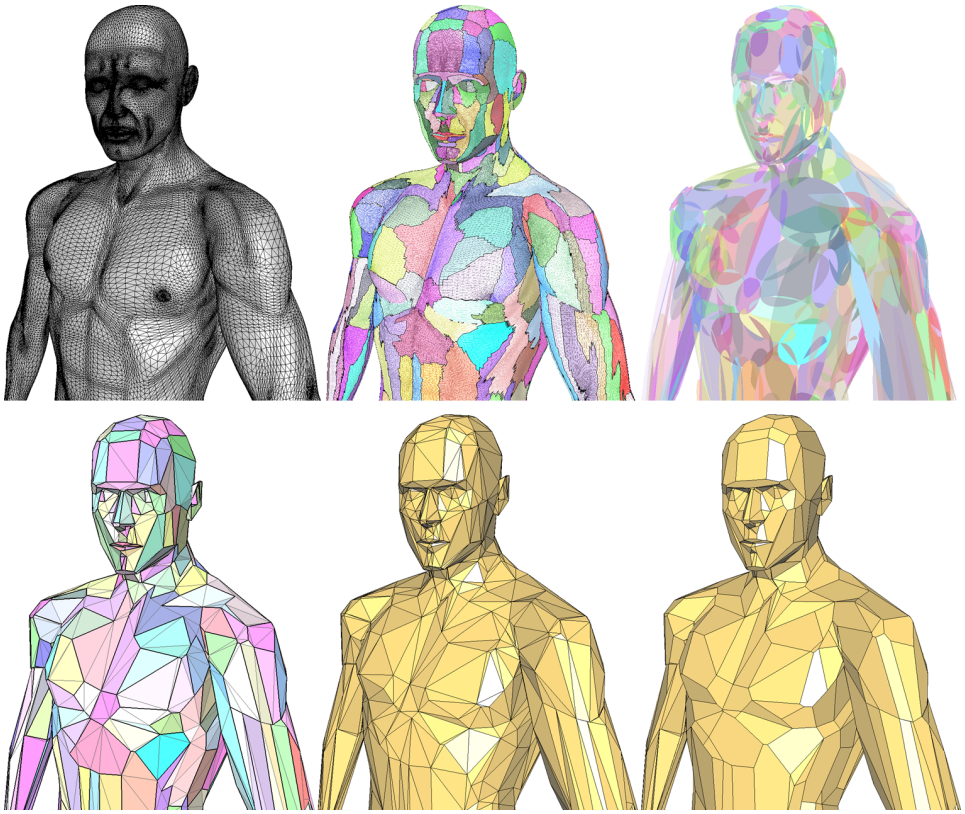


**Figure 2.22:** Man. Top: input surface mesh (150K triangles), optimized partition with 800 proxies, and the corresponding planar proxies. Bottom: triangle mesh extracted with color attributes from the optimized partition, triangle mesh extracted without color attributes and final polygon mesh.

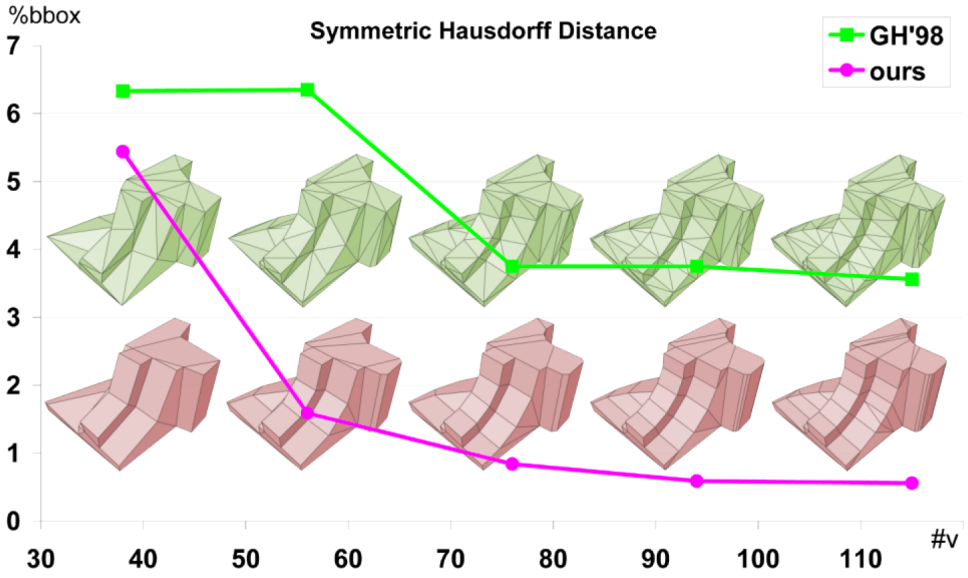



**Figure 2.23:** Comparison of the Hausdorff error for QEM [GH98] and the $\mathcal{L}^{2,1}$ technique, for equal number of vertices (a comparison using equal number of edges leads to a similar curve).

### 2.3.4 Summary

We proposed a variational shape approximation approach that breaks away from the common approximation paradigm consisting in directly optimizing a piecewise-linear approximant of an original surface. Through mutual and repeated error-driven optimizations of a partition and a set of local proxies, the proposed method ends up providing concise geometric representations in the form of either local best-fit proxies or of polygonal meshes.

#### 2.3.4.1 Strengths

The variational aspect of the approach we presented improves over greedy techniques, albeit at the price of higher computations. The symmetries are quickly found, the anisotropy is automatically detected and adapted to, and the regions line up well with the features. The $\mathcal{L}^{2,1}$ shape metric turns out to capture more subtle details than the common $\mathcal{L}^2$ metric, especially for mechanical parts. Another added value is its generality, as other error metrics and proxies can be put to work within the algorithm. The numerics in the algorithm are based on integration in the sense that the metric is computed in closed-form over the triangles of each region. This provides better independence to the input tessellation, albeit only for the fitting phase. Finally, the algorithm is amenable to the generation of polygon meshes with nearly planar polygons, a desirable feature for some applications.

#### 2.3.4.2 Weaknesses

Being based on iterative optimization, our technique cannot compete with greedy methods such as [GH98] in terms of computational time: our algorithm can be ten to twenty times slower. One important weakness lies within the partitioning phase. The error-driven priority queue used for region growing is noise sensitive and depends on the input tessellation, contrary to the fitting phase. Devising a robust partitioning phase, resilient to noise and input tessellation, would be be relevant for measurement data. At the algorithmic level, the final mesh generation technique does not guarantee absence of self-intersections which can happen, e.g., for two close surface sheets.

#### 2.3.4.3 Follow-ups

The simplicity of the proposed approach has inspired several follow-ups[2]. In [WK05] our approach is taken a step further by allowing for proxies other

[2]the paper [CSAD04] is cited 187 times according to Google Scholar.

than simple planes, such as spheres, cylinders, and rolling-ball blends. Apart from requiring fewer primitives to achieve a given fitting approximation quality, this method can also recover the semantic structure of an input model to some extend. In [JKS05] a similar idea is used to decompose the input mesh into nearly developable segments. Another extension of this algorithm to handle general quadric proxies has been elaborated by Yan et al. [YLW06]. Faithful approximations are obtained using fewer proxies than when using planar or CAD proxies, see Figure 2.24. Canonical proxies such as cylinders and spheres can be identified and even favored within the same framework.

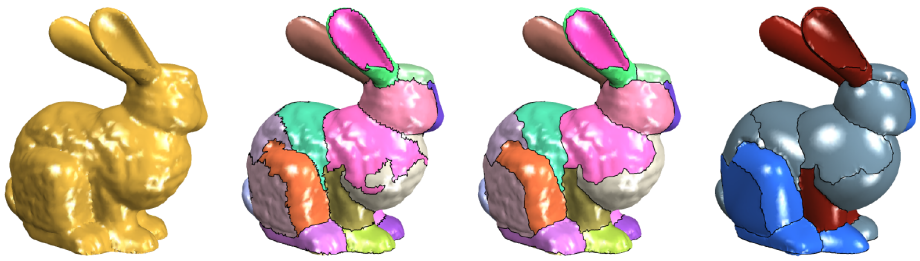


**Figure 2.24:** Stanford bunny approximated by 28 quadric proxies. Figure taken from [YLW06].

Another extension to shape approximation with spheres has been proposed [WZS+06], with a variety of error metrics including total outside volume, shadowing fidelity, and proximity measurement. Finally, an extension to symmetry preserving remeshing has been proposed by Podolak et al. [PGR07], where several instances of symmetric proxies are instantiated for the optimization. The output meshes are shown to preserve and enhance the symmetries of the input meshes.

#### 2.3.4.4 Future Work

Future work include the investigation of a Sobolev metric ($H^1$), a linear combination of $\mathcal{L}^2$ and $\mathcal{L}^{2,1}$ energies, that would require a low-order polynomial root solver to compute the best fit. Recent advances in improved initializations for k-means clustering [AV07] could also be used to ameliorate the practical convergence rate and energy minimum of the algorithm.

Our initial reason for not using higher order proxies such as quadrics was the lack of closed form solutions for both fitting and partitioning phases. The follow up by Yan et al. [YLW06] shows that even approximations can provide satisfactory results. A collaboration with experts on low degree algebraic objects would be required to push these ideas further [Pet07]. Another issue in fitting general quadrics is the generation of a final representation,

especially in the presence of intricate boundaries between regions: should we try meshing the regions? or fit parametric surface patches instead?

Finally, one important problem for industrial applications is the simplification of complex geometries with two guarantees: remaining intersection-free and within a tolerance volume. Although several approaches address the problem of volume tolerance [BBVK04, BF05], there is a surprisingly few number of techniques that address the intersection free guarantee [GBK03]. In our approach we are using planes first, before assembling surface meshes in a perfectible manner; switching to a fully volumetric approach with volumetric proxies may be relevant to satisfy both guarantees simultaneously.

## 2.4 Quadrangle Surface Tiling

In this section we focus on quadrangle surface tiling of triangle surface meshes. Partitioning a surface into quadrilateral regions is a common requirement in computer graphics, computer aided geometric design and reverse engineering. Such quad tilings are amenable to a variety of subsequent applications due to their tensor-product nature, such as B-spline fitting, finite elements, texture atlasing, and generation of modulation maps. Automatically converting a triangulated surface (issued from a 3D scanner for instance) into a quad mesh is, however, challenging. Stringent topological conditions make quadrangle tiling a rather complex and global problem [Ede00]. Application-dependent meshing requirements such as edge orthogonality, orientation and alignment of the elements with the geometry, sizing, and mesh regularity add further hurdles.

### 2.4.1 Related Work

Comprehensive reviews [ACSD+03, BMRJ04, AUGA07, DKG05], hint at a need for algorithms offering more control on the mesh regularity, as well as on the shape, size, direction and alignment of the mesh elements with geometric or semantic features.

One technique proposes to use holomorphic discrete 1-forms [GY03] to generate fully regular quadrilateral meshes (except along a seam). Unfortunately, the holomorphic requirement leaves little control over the local alignment of the mesh elements and creates potentially large area distortion, even after optimization [JWYG04]. Another technique introduces a radically different approach to conformal parameterization with arbitrary cone singularities [KSS06]: distortion is concentrated at carefully chosen places so as to allow better, global control of area distortion and hence a good control over mesh sizing.

An approach proposed in [ACSD+03, MK04] consists in tracing curvature lines, thereby enforcing proper alignment of the mesh edges while creating quad-dominant tilings. The placement of these lines are based on local decisions, resulting in "hanging" lines all over the mesh: T-junctions and poor regularity of the mesh ensue. When targeting higher mesh regularity, a better approach defines these lines as isolevels and steepest descents of a global potential [DKG05]. As a result of this type of contouring, the lines are either closed curves (so-called isoparametric flow lines), or streamlines (gradient flow lines) obtained by numerical integration, leading to less T-junctions and a better regularity of the final quad mesh. This method also allows some design control through user-defined selection of a number of local extrema of the potential (be they points, or even polylines). How-

ever, each local extremum corresponds to an index 1 in the gradient field of the potential; because of the Hopf-Poincaré index theorem, this means that a number of other singularities (most likely saddles, of index $-1$) will be consequently created too, as the indices of all singularities of the vector field must sum up to the Euler characteristic of the surface. Therefore, design control does not scale nicely as each additional constraint increases shape distortion of tiles all over the rest of the surface.

Ray et al. [RLL+06] introduces another contouring technique performing a non-linear optimization of periodic parameters to best align directions along two given orthogonal vector fields, offering more freedom on the type of singularities than any previous approach. In particular, indices of type $1/2$ and $1/4$ are allowed, providing a satisfactory balance between area distortion and alignment control. In addition, a curl-correction step modulating the norm of the vector field is devised to minimize the number of point singularities. A more recent work focuses on the design of direction fields [RVAL09].

Finally, a Morse-theoretic approach [DBG+06] has also been devised, but once again, providing little control over design and resulting in singularities at conspicuous places and elements of arbitrary shapes. Nevertheless this approach was recently improved [HZM+08] so as to control direction, alignment as well as sizing.

### 2.4.2 Approach

The use of isolines as a basis for surface tiling is appealing from a practical point of view: it naturally privileges regularity of the resulting quad mesh, and is numerically more robust than the use of streamlines as it alleviates the need for numerical integration. Isovalues can eventually be changed in order to adapt spacing between isolines. Therefore, we propose to design an algorithm which computes two piecewise continuous and discretely harmonic scalar functions, so that their respective contouring provides the final tiling with no T-junctions (see Figure 2.25).

Based on discrete differential forms, our approach provides control not only on the position of singularities and their (possibly fractional) indices, but also on the way these singularities are interconnected in the final tiling. This information on the topological structure of the output mesh is encoded in what we call a *singularity graph*, specified by the user. This singularity graph is also a way to control directions for the tiles depending on their locations. The core of our algorithm relies on extensions of the well-known cotangent formula that enriches the space of discrete harmonic functions:

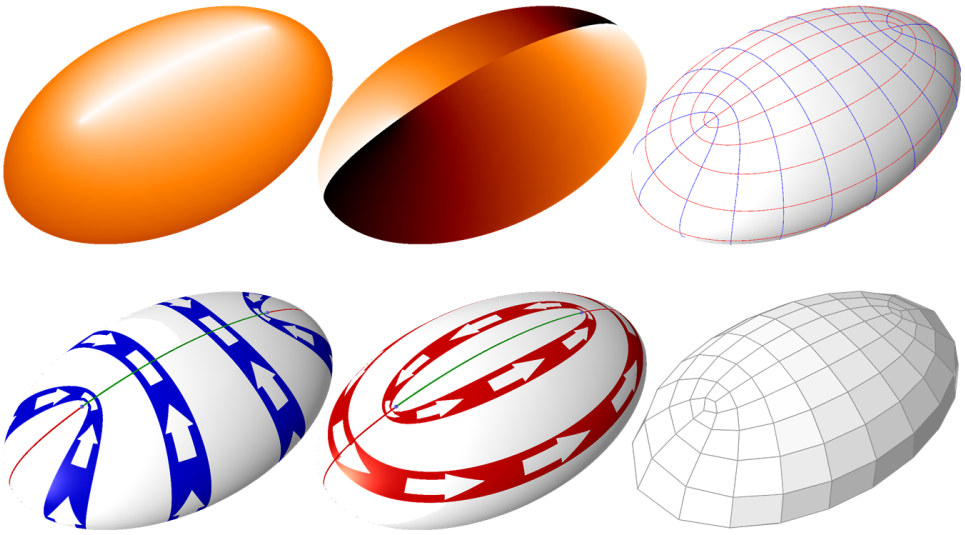


**Figure 2.25:** Quadrangle surface tiling. Left: two harmonic scalar functions and their gradient depicted with arrows. Right: iso-contouring lines of the scalar functions and resulting quadrangle tiling with direction control.

we thus stay within the framework of linear algebra, avoiding non-linear minimization required in [RLL+06, KSS06] that can impair scalability.

#### 2.4.2.1 Local Quadrangulation as Contouring

We start by using a reverse argument. Suppose that we already have a small surface patch composed of well-shaped quadrangles. From this tiling, we can first set a local $(u, v)$ coordinate system (with directions $e_u$ and $e_v$) of the surface to be aligned with the edges of the tiles. We can then define a metric $\langle\,,\rangle$ so that $\langle e_u, e_v\rangle{=}0$ everywhere, and so that lengths of each quad edge are unit. Thus, the mesh is locally defined by integer $u$ and $v$ isovalues. In addition the gradients of the two parameters $\nabla u$ and $\nabla v$ are orthogonal in the prescribed metric. The way we have defined the metric also guarantees that we must have the magnitudes of the gradients equal to each other. The two conditions together are known as the Cauchy-Riemann equations for the parameters $u$ and $v$ of this patch:

$$\langle \nabla u, \nabla v\rangle = 0 \qquad \text{and} \qquad \langle \nabla u, \nabla u\rangle = \langle \nabla v, \nabla v\rangle.$$

These two equations can be formulated using the differentials of $u$ and $v$, as well as the Hodge star induced by our metric; the two 0-forms $u$ and $v$ satisfy:

$$du = \star dv$$

Notice that we can deduce (by applying $d$ and $d\star$ to the previous equation) that $d \star dv = d \star du = 0$, hence $du$ and $dv$ are both co-closed. Since

$d \circ d = 0$, both are also closed. Therefore, $du$ and $dv$ must be harmonic. In more traditional notation, both gradient fields are curl- and divergence-free. Another consequence of the co-closedness of the two differentials is that both $u$ and $v$ are also harmonic, i.e., their Laplacian vanishes. These properties explain the popularity of harmonic functions in Euclidean space, where orthogonality means $\pi/2$ angles, hence leading to well-shaped quads [DKG05, GY03].

#### 2.4.2.2 Towards Global Contouring

To extend the basic principle explained in the previous section from a local to a global quadrangle tiling, we must overcome a number of issues.

**Necessity of discontinuities** First, globally continuous harmonic scalar potentials are too restrictive for our purposes. For the case of a genus-0 closed manifold, there are no globally continuous harmonic potentials other than the constant ones, of little worth. A way to deal with this problem is to add point singularities, which amounts to piercing point holes (poles) at various locations on the surface. Let us begin with a trivial topology and pierce a sphere once at the top and once at the bottom; what remains is a continuous harmonic potential $u$, with extrema at the two poles, thus with flow lines defining longitudes (see Figure 2.26).

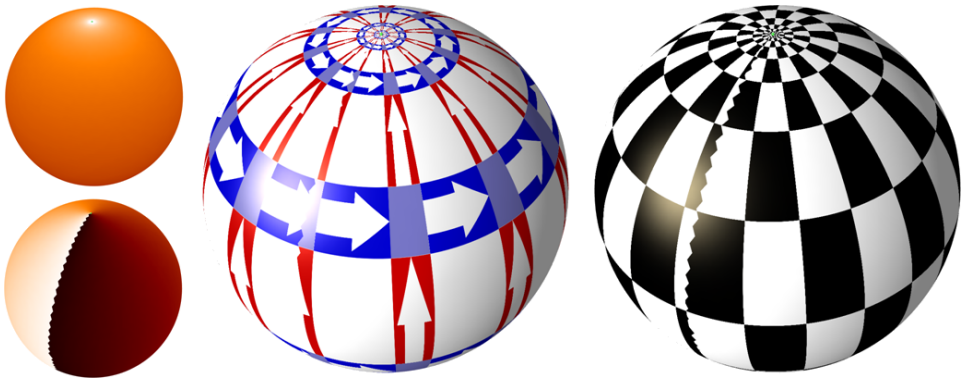


**Figure 2.26:** Two pole singularities on a sphere. Left: two harmonic functions $u$ and $v$. $u$ is continuous. $v$ is discontinuous with a constant jump along the singularity line joining the poles. Middle: derivatives $du$ and $dv$ depicted with arrow textures. Both $du$ and $du$ are continuous. $du$ goes from the south to the north pole while $dv$ winds around the poles. Right: tiling obtained through contouring $u$ and $v$ with integer iso-contouring lines. The discontinuity is visible.

However, the corresponding $v$ potential cannot be globally continuous since its derivative $dv$ has closed flow lines, namely latitudes. Therefore, the

only hope to extend the contouring approach is to allow the potentials to be piecewise continuous, i.e., only continuous inside non-overlapping patches of the manifold akin to the notion of charts [YZ04]. We may find a potential $v$ that is continuous everywhere except on a line joining the two poles, along which the jump of the $v$-value is constant due to the harmonicity property. In this case $v$ is discontinuous while $dv$ is not. Quadrangle tiling through integer-contouring requires that the jump (or equivalently the circulation around the poles) is an integer number. This requirement is easily matched by scaling the whole one-form $dv$ so that the jump is rounded to its nearest integer. This way, the jump is invisible when contouring, see Figure 2.27.

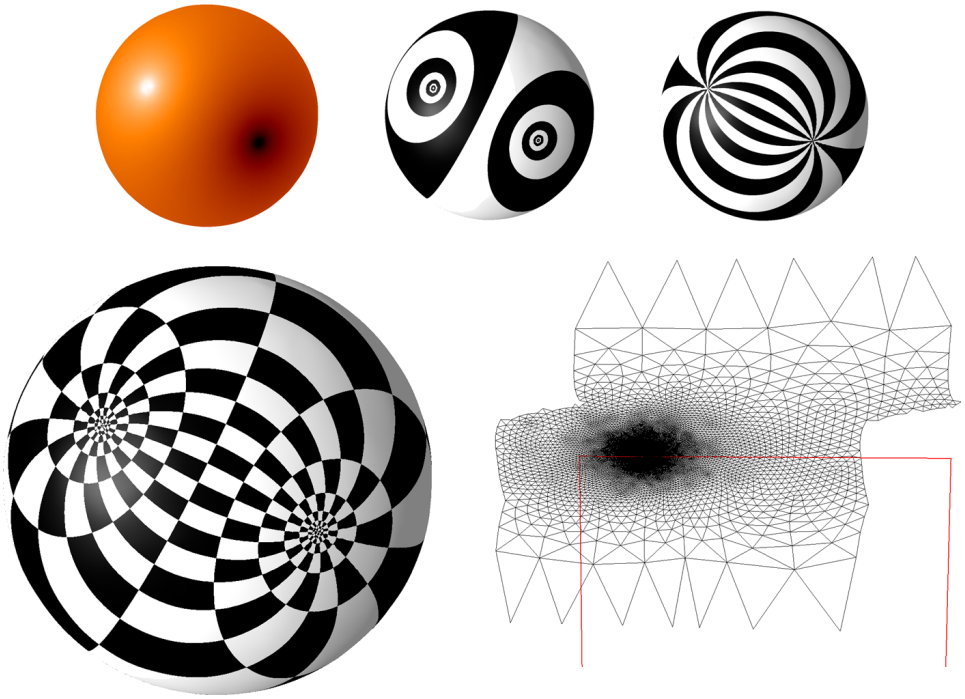


**Figure 2.27:** Seamless tiling with two poles. Top left: harmonic function $u$ with two pole singularities. Top middle and right: iso-contouring $u$ and $v$. $v$ is obtained by integrating a scaled version of $dv$ so that the jump of $v$ along a line joining the two poles is integer (not shown). Bottom left: tiling through iso-contouring of $u$ and $v$. The discontinuity is invisible. Bottom right: $u - v$ parameterization of the input mesh cut open at the poles and along the jump line. The red lines delineate the unit parameter square.

A pole leads to high distortion in the final tiling, as many contouring lines converge to it. One way to alleviate this issue is to stretch it into a line, so as to redirect many contouring lines across the line instead of to a single pole. In essence such a line can be seen as a pole (singularity of index one) turned into two singularities of index one half, so-called wedges. The example depicted in Figure 2.28 illustrates how the iso-lines of $u$ are

constrained to be tangential to the singularity lines. Although this provides a nice way to control alignment of edges in the final tiling, the iso-lines of $v$ are discontinuous across the lines as such a singularity is equivalent to piercing a hole.

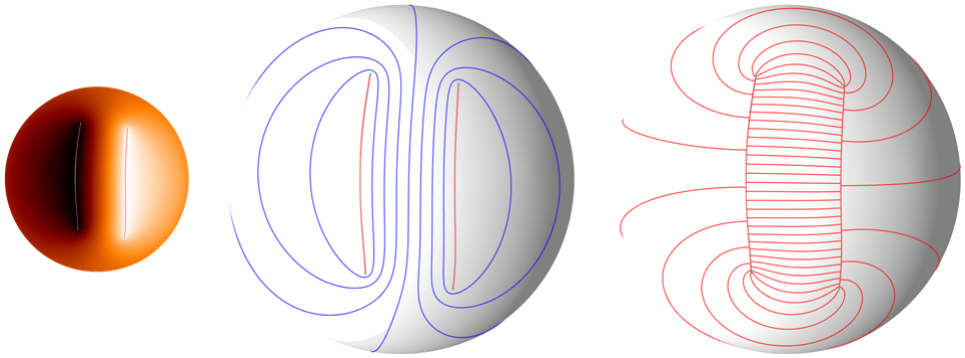


**Figure 2.28:** Line singularity. Two line are defined, along which $u$ is constrained to 0 and 1 respectively. Left: $u$, continuous. Middle: iso-contouring lines of $u$. Right: iso-contouring lines of $v$, discontinuous across the lines. This translates into T-junctions in the final tiling.

For arbitrary topology and number of pole singularities the complication is substantial, as there is a whole basis of harmonic forms (one form per cohomology class). We refer to [TACSD06](Appendix) for details about the computation of such basis, and how we can apply the same integer-rounding idea as described above to avoid T-junctions, for an arbitrary number of singularities. When put to work in the design of surface tilings, such continuous harmonic 1-forms are not ideal. In addition to high distortion (especially around poles), they inevitably generates saddles when the total index of the singularities is higher than the Euler characteristic of the surface. Figure 2.29 depicts four poles place on a sphere, and six lines placed on the mannequin head model. To be able to generate fractional singularities, one needs to allow for certain types of discontinuities of 1-forms.

**Compatibility conditions** We now assume that the 0-forms (the potentials $u$ and $v$) can contain singularities, i.e., jumps along certain edges. Similarly, their differentials (the 1-forms $du$ and $dv$, akin to the gradients of each potential) may have singularities at the same locations, i.e., the vector fields representing these 1-forms may jump across patch boundaries. We will denote such a boundary between two continuous patches a singularity line. We will set simple compatibility conditions on the jumps of potentials and of their differentials that will guarantee that a global contouring of $u$ and $v$ results in a proper tiling. More precisely, linear constraints of continuity on the two potentials can ensure continuity of the isolines: if we trace all isolines

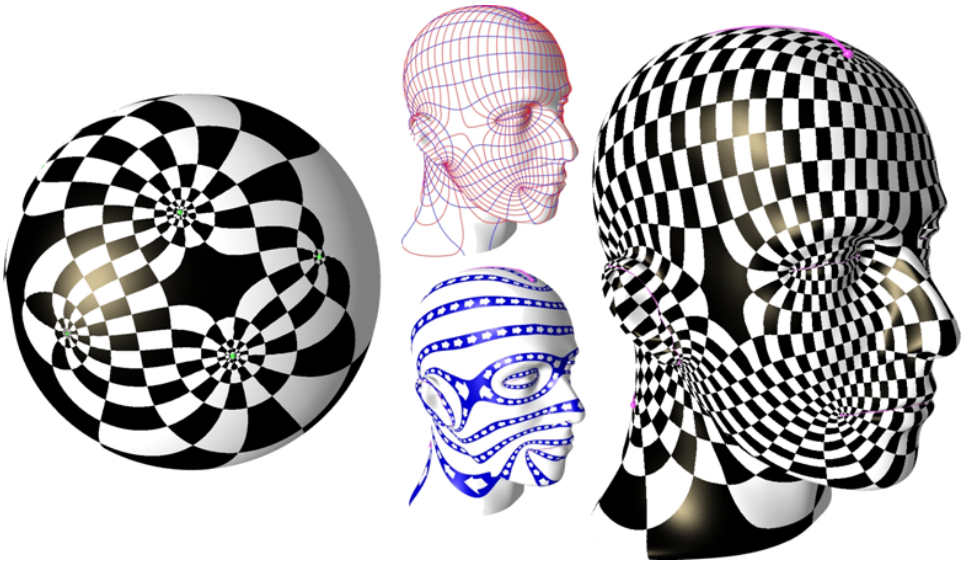


**Figure 2.29:** Undesirable Singularities. Left: more than two poles (each of index 1) on a genus-0 surface inevitably create singularities with negative index (saddles), creating large and distorted n-gons. Right: six lines placed on a genus-0 surface to better control alignment also create saddles.

with integer values, then the necessary and sufficient compatibility conditions are that the jumps of the potentials should be integer. On the other hand, the smoothness of the isolines will be ensured by a (tweaked, yet still linear) condition of harmonicity of the two potentials at patch boundaries. This last condition is, in fact, a continuity condition for 1-forms across the singularity line. We thus call this condition *singular continuity* to convey the notion of smoothness modulo the presence of a singularity.

**Singular Continuity of Discrete Forms** As mentioned above, obtaining a quad-dominant tiling on a disk-like patch through contouring two 0-forms $u$ and $v$ is rather easy. However, enforcing a proper tiling throughout the surface requires strong compatibility conditions at each singular line. Fortunately, only three different types of singular continuity across two neighboring patches can happen (see Figure 2.30): *regular* (when both $u$ and $v$ directions individually match between the two patches), *reverse* (when both $u$ and $v$ directions change their orientations across the boundary), and *switch* (when the $u$ and $v$ directions are switched on the shared boundary). Only then can we get a globally consistent tiling of the surface.

#### 2.4.2.3 Enforcing Singular Continuity of Forms

We now go over the various cases of continuity. As we use two linear equations per vertex, we describe the different vertex types that we can encounter on a mesh: a vertex can be strictly within a patch, or on a particular type of singularity line.

**Free Vertices** When a vertex $i$ is within a patch, i.e., not on any singularity line, we simply wish to enforce harmonicity of both 0-forms $u$ and $v$. Consequently, the common harmonicity condition [PP93] is imposed on this vertex, yielding:

$$\sum_{j \in \mathcal{N}(i)} w_{ij} \begin{pmatrix} u_i - u_j \\ v_i - v_j \end{pmatrix} = 0$$

where the index $j$ goes through all the immediate neighboring vertices of $i$, $u_k$ (resp., $v_k$) represents the value of $u$ (resp., $v$) at vertex indexed $k$. For the Euclidean metric, the weights $w_{ij}$ are the well-known sum of cotangents of angles opposite to edge $ij$.

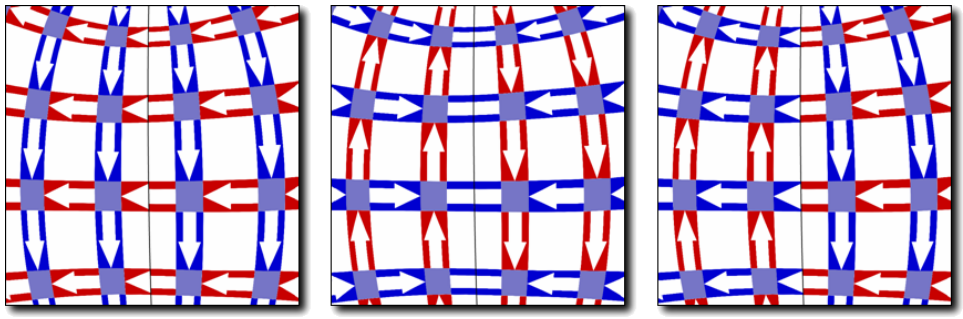


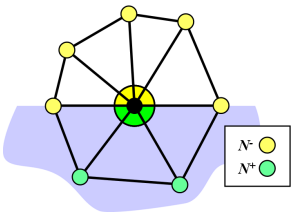
**Figure 2.30:** Singular continuity. Three different types of continuity through a singularity line: regular, opposite and switch. Blue/red arrows are along isolines of $u/v$.

**Vertices with Regular Continuity** When a vertex is on a *regular* singularity line between two patches, we assume that the fields $u$ and $v$ are smooth across the patch boundary modulo a constant offset. That is, if we call $u^-$ (resp., $v^-$) the potential $u$ of this vertex using its value from one of the patches, and $u^+$ (resp., $v^+$) the value at the same vertex but considering its value from the other patch, we wish to have:

$$u^- - u^+ = P_1 \qquad v^- - v^+ = P_2, \tag{2.3}$$

where $P_1$ and $P_2$ are two arbitrary integer constants associated with this particular patch boundary (we will discuss how to choose their values adequately later on). Obviously, enforcing this equality modulo offset will

guarantee that integer isolines of $u$ and $v$ do match up at the boundary. Notice also that it corresponds to guaranteeing continuity of the 1-forms $du$ and $dv$, as $d(u^+ - u^-) = du^+ - du^- = 0$. Finally, to ensure smoothness of these isolines, we enforce harmonicity of the two potentials taking the jump into account (see inset for conventions used):




$$\sum_{j\in\mathcal{N}^-(i)} w_{ij} \begin{pmatrix} u_i^- - u_j \\ v_i^- - v_j \end{pmatrix} + \sum_{j\in\mathcal{N}^+(i)} w_{ij} \begin{pmatrix} u_i^+ - u_j \\ v_i^+ - v_j \end{pmatrix} = 0$$

The above conditions can be rewritten using only one value of $u$ and one value of $v$ for the boundary vertex $i$, therefore alleviating the need for storing two different values, one on each side of the singularity line. Indeed, if we assume $u_i \equiv u_i^-$, and thanks to Eq. (2.3):

$$\sum_{j\in\mathcal{N}(i)} w_{ij} \begin{pmatrix} u_i - u_j \\ v_i - v_j \end{pmatrix} = \sum_{j\in\mathcal{N}^+(i)} w_{ij} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix}.$$

Notice that this equation is a variant of the former case, modifying the right hand side to impose the correct conditions on each side of the boundary.

**Vertices with Reverse Continuity** This time, we want the 0-forms $u$ and $v$ to change orientation when crossing the patch boundary. That is, we wish to have $du^+ = -du^-$, and $dv^+ = -dv^-$. These constraints are enforced by defining:

$$u^+ + u^- = Q_1 \qquad v^+ + v^- = Q_2,$$

where $Q_1$ and $Q_2$ are two integer constants associated to the boundary on which the vertex lies. We now enforce harmonicity of the two potentials at $i$ modulo the reversal:

$$\sum_{j\in\mathcal{N}^-(i)} w_{ij} \begin{pmatrix} u_i^- - u_j \\ v_i^- - v_j \end{pmatrix} + \sum_{j\in\mathcal{N}^+(i)} w_{ij} \begin{pmatrix} u_j - u_i^+ \\ v_j - v_i^+ \end{pmatrix} = 0$$

Again, one notices that a simpler expression using only one value for vertex $i$ and a non-zero right-hand side, is:

$$\sum_{j\in\mathcal{N}^-(i)} w_{ij} \begin{pmatrix} u_i - u_j \\ v_i - v_j \end{pmatrix} + \sum_{j\in\mathcal{N}^+(i)} w_{ij} \begin{pmatrix} u_i + u_j \\ v_i + v_j \end{pmatrix} = \sum_{j\in\mathcal{N}^+(i)} w_{ij} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}.$$

This last expression preserves the symmetric nature of the Laplacian matrix. This is a practical feature as state-of-the-art linear solvers scale well on symmetric linear systems [TCR05, BBK05].

**Vertices with Switch Continuity** Finally, for vertices on a singularity line on which we want $u$ and $v$ to switch, we simply enforce that $du^+ = dv^-$ and $dv^+ = -du^-$. Notice the extra minus sign, because switching $u$ and $v$ reverses one of the two directions. Again, these conditions are satisfied if:

$$v^- - u^+ = R_1 \qquad v^+ + u^- = R_2,$$

Finally, to ensure smoothness of these isolines, we enforce harmonicity of both potentials given this discontinuity through:

$$\sum_{j \in \mathcal{N}^-(i)} w_{ij} \begin{pmatrix} u_i^- - u_j \\ v_i^- - v_j \end{pmatrix} + \sum_{j \in \mathcal{N}^+(i)} w_{ij} \begin{pmatrix} v_j - v_i^+ \\ u_i^+ - u_j \end{pmatrix} = 0$$

The resulting symmetric expression, using only one value for the vertex $i$ and a non-zero right-hand side, is now:

$$\sum_{j \in \mathcal{N}^-(i)} w_{ij} \begin{pmatrix} u_i - u_j \\ v_i - v_j \end{pmatrix} + \sum_{j \in \mathcal{N}^+(i)} w_{ij} \begin{pmatrix} u_i + v_j \\ v_i - u_j \end{pmatrix} = \sum_{j \in \mathcal{N}^+(i)} w_{ij} \begin{pmatrix} R_2 \\ R_1 \end{pmatrix}.$$

Finally there is an analogous formula for what we could call reverse-switch continuity vertices, when we switch $u$ and $-v$.

#### 2.4.2.4 Properties of Singular Continuity

The four cases discussed above are enough to provide a rich repertoire of singularities. In particular, the previously mentioned case of a genus-0 object with two poles can be handled quite simply by linking the two poles with a singularity line: this "virtual" cut on the sphere creates one single patch touching itself along a regular continuity boundary. Now, the two potentials $u$ and $v$ can be computed per vertex by solving a modified Laplace equation, with vertices along the singularity line having different coefficients and non-zero right-hand sides. One remarkable property of the previous equations is that the exact position of the various boundaries between patches does not affect the final result: any boundary line in the same homology class as the original one will result in the same quad mesh. Although the 0-forms will be different (since their jumps will be located at distinct locations), their contouring will stay the same: only the local sign of their gradients will be affected in the reverse continuity case, while the gradient of $u$ will become the gradient of $v$ in the switch continuity case. Therefore, the only real parameters are the set of constants, chosen for each boundary (that we called $P_1, P_2, Q_1, Q_2, R_1$, and $R_2$ previously). This is quite convenient as no special effort needs to be spent on getting smooth singularity lines (see Figure 2.31, right). In other words, only the topology of the patches is needed.

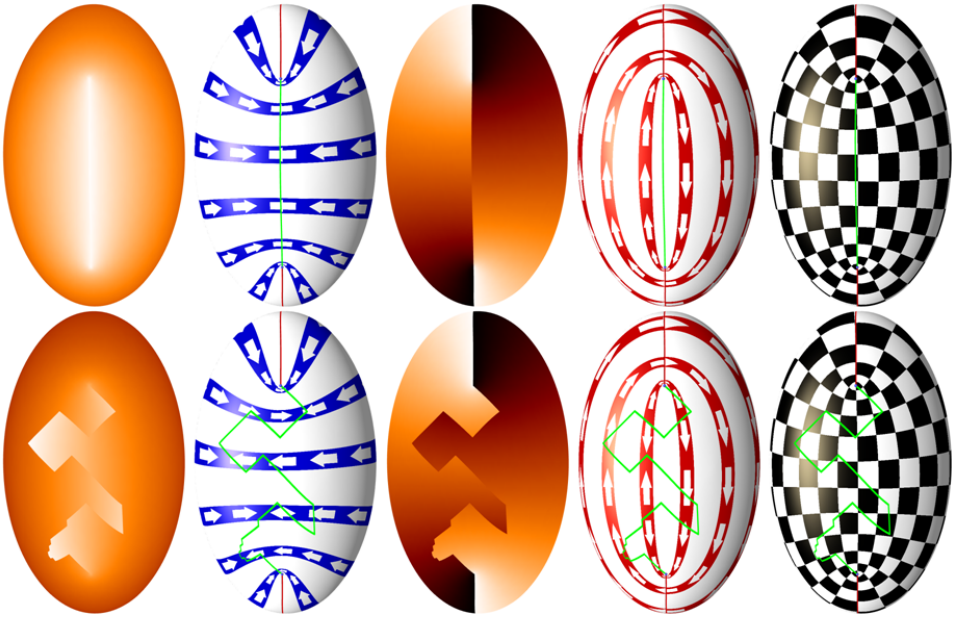


**Figure 2.31:** Line Singularity. Top line, from left to right: Piecewise-continuous harmonic potentials $u$ and $v$ (color-shaded); Red and blue arrows depict the direction of the potential gradients; a checkerboard is mapped onto the ellipsoid using $(u, v)$ as texture coordinates. Bottom line: when the singularity line is wiggly, the two potential functions change, but their isolines remain identical to the previous case.

**Other Typical Singularities** Other singularities can be achieved by designing a proper choice of boundary continuity between various patches. For instance, a trisector singularity is obtained by assembling three concurrent lines, all of continuity type reverse (see Figure 2.32).

A square singularity, i.e., four index-1/4 poles forming a square-shaped index-1 singularity, is assembled from four lines in the shape of a square, with type regular, switch, reverse, and switch in cyclic order (see Figure 2.33). Notice that these cases create little distortion and by design no T-junctions. In our approach a singularity graph is the main backbone behind the final quadrangle tiling design.

#### 2.4.2.5 Design

Allowing quad mesh design flexibility requires the use of a potentially large set of patches: the more patches we define, the better we can control the local alignment of edges as well as the local area distortion. Thus, we propose a user-guided way to create this patch layout through the definition of a singularity graph, representing a topological template linking the sin-

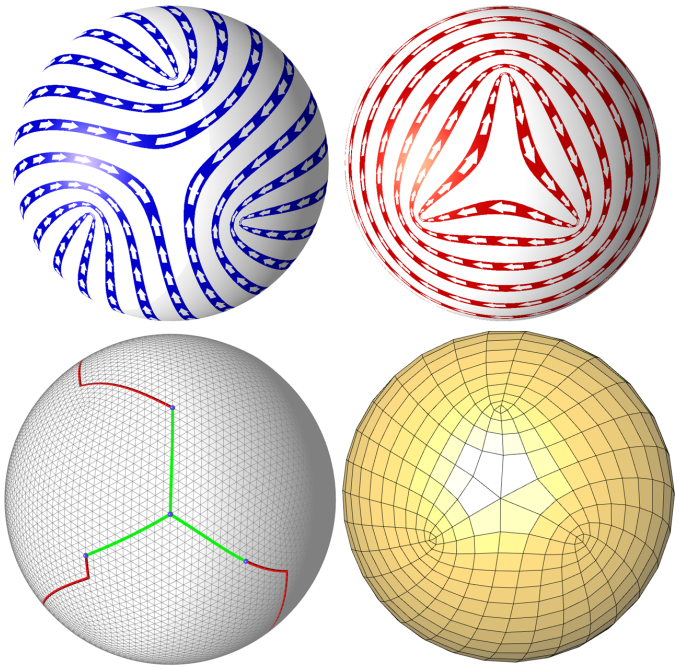


**Figure 2.32:** A trisector singularity is obtained by assembling line singularities of type regular and reverse.

gularities. We call a singularity graph a meta-mesh whose meta-faces are non-overlapping patches of the original mesh, and whose meta-edges are assigned one of the singularity continuity conditions described above. The vertices of the singularity graph (called meta-vertices to avoid ambiguity) are a subset of the vertices of the input triangle mesh. They should be thought of as salient points of the manifold, as they will live at the intersection of several regular patches in the final quadrangle tiling. Connectivity between meta-vertices define the meta-edges of the singularity graph. Each of these meta-edges are made out of two half-edges, oppositely oriented. Finally, every cycle of half meta-edges defines a (meta-)face of the singular graph. Such a face corresponds to a patch in which our 0-forms will be smooth and continuous. In this section, we will call $F$ (resp., $E$) the number of meta-faces (resp., meta-edges).

**Determining Types of Singular Continuity** Given a singular graph, we must assign to each meta-edge a particular singular continuity type. These assignments can be automatically obtained if we first tag each meta-halfedge as $u$, $-u$, $v$, or $-v$ according to their alignment with increasing or decreasing directions of the parameter. Indeed, we would ideally like to map each meta-face of the graph to an orthogonal polygon in the parameter

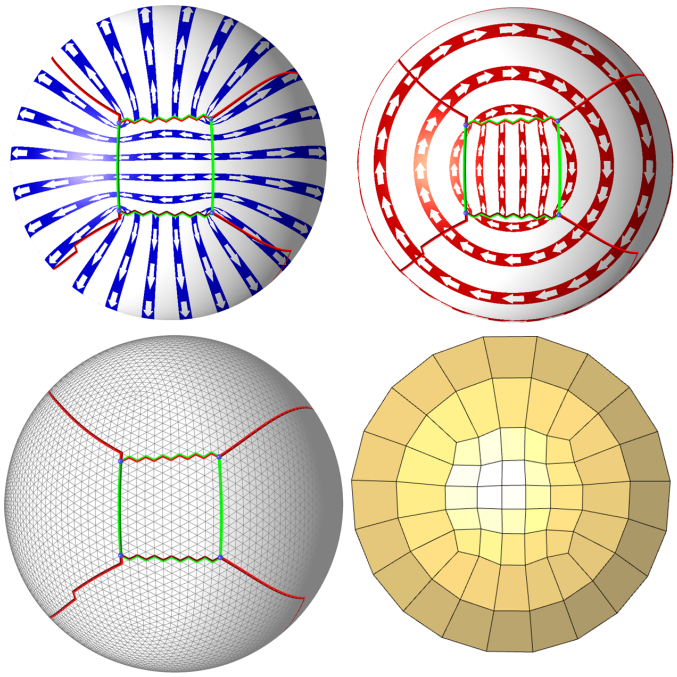
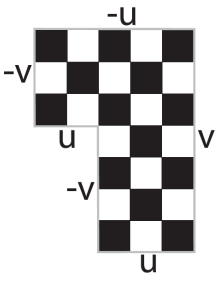


**Figure 2.33:** A square singularity is obtained by assembling line singularities of type regular, reverse and switch.

domain to guarantee the existence of a regular quadrangulation inside this patch. That is, the face should conceptually look like a simple polygon in the $(u, v)$ parameter plane with only angles multiple of $\pi/2$ (see inset)—in fact, the choice of which edge is aligned with $u$ vs. $v$ does not affect the final quad mesh, so this polygon can be arbitrarily rotated by multiples of $\pi/2$ too. This condition imposes a constraint on the half-edge assignments, and we will provide an automatic procedure to enforce it on each face. After such an assignment is provided, the corresponding singular continuity types for all meta-edges becomes simple, as, for each pair of half-edge assignments, correspond the following continuity types:




- Regular: $\{u, -u\}$, $\{-u, u\}$, $\{v, -v\}$, $\{-v, v\}$;
- Reverse: $\{u, u\}$, $\{v, v\}$, $\{-u, -u\}$, $\{-v, -v\}$;
- Switch: $\{u, v\}$, $\{v, -u\}$, $\{-u, -v\}$, $\{-v, u\}$, and $\{-u, v\}$, $\{-v, -u\}$, $\{u, -v\}$, $\{v, -u\}$.

**Designing a Tiling** Designing the final quadrangle tiling amounts to deciding on an assignment of $(u, v)$ values at each corner of every meta-face. These are, indeed, the only constraints that need to be fixed for the modified Laplace equation to be solvable: once these meta-parameters are known, all the constants defined per patch-boundary in the continuity equations are determined. Thus, we begin by computing these values by solving a small meta-system. We will then inject the resulting meta-parameters into the system of (modified) Laplace equations for the final solve.

The constraints on these parameters depend only on the differences of $u$ and $v$ values on each meta-edge. Let $Du$ (resp., $Dv$) be such a difference in $u$ (resp., $v$) value over a meta-edge. In each meta-face, the sum of all the $Du$'s of half-edges tagged as $u$ must equal the sum of the $Du$'s for those tagged as $-u$: the meta-face is a closed polygon in parameter space. The same argument applies for the sum of $Dv$ of $v$ edges and that of $-v$ edges (in the language of differential forms, this states that $du$ and $dv$ must be closed on the meta-mesh too). Similarly, as we wish to have isolines stitching properly across meta-faces, we must have equal differences for each half-edge of a meta-edge. Thus, these differences are $E$ coefficients that need to be set, and there are $2F$ linear constraints on them (one for $u$ and one for $v$ per meta-face). We now set up in a small linear system these $2F$ constraint equations for $E$ variables. However, the constraints can (and will often) be redundant. We thus use Gauss elimination to find the independent equations. This process is extremely fast since the graph contains typically three orders of magnitude fewer edges than the original mesh, and all the coefficients in this meta linear system are either 1 or -1 (since $Du$ (or $Dv$) is computed as the simple difference between two parameter values). Additionally, notice that the $2F$ constraint equations sum to zero: we can thus guarantee that there will be at least $E$-$2F$+1 number of independent variables. Since a meta-face is homeomorphic to an orthogonal polygon, each meta-face has at least 4 edges. Therefore, $E \geq 2F$, and there is always at least one degree of freedom. The user is requested to enter values based on the number of isolines desired on those meta-edges (this is, indeed the geometric meaning of $Du$).

Once the meta-parameters are set, we can assemble the global linear system for the 0-forms $u$ and $v$ of the original mesh. The system is created by assembling two linear equations per vertex $v_i$ (depending if it is free or on a line singularity), and none for the vertices on corners of meta-faces, as they are already determined by the meta-parameters. Whenever one of these linear equations involves a meta-face corner, its value (i.e., one of the meta-parameters) is constrained and therefore substituted and moved to the right-hand side of the system. The matrix of this system is sparse

and symmetric. The final mesh is extracted through integer contouring: we walk in the triangulation from an integer intersection of $u$ and $v$ to the next. Note that we may cross a line singularity; in that case, we account for the singularity type to be able to resume the walk on the other side and find the next intersection.

### 2.4.3 Results

Figure 2.34 illustrates how the user can finely control the alignment of the mesh with features or semantically relevant directions. Note how a quarter-pole is obtained by a regular line and a switch line meeting at a meta-vertex of the singularity graph.

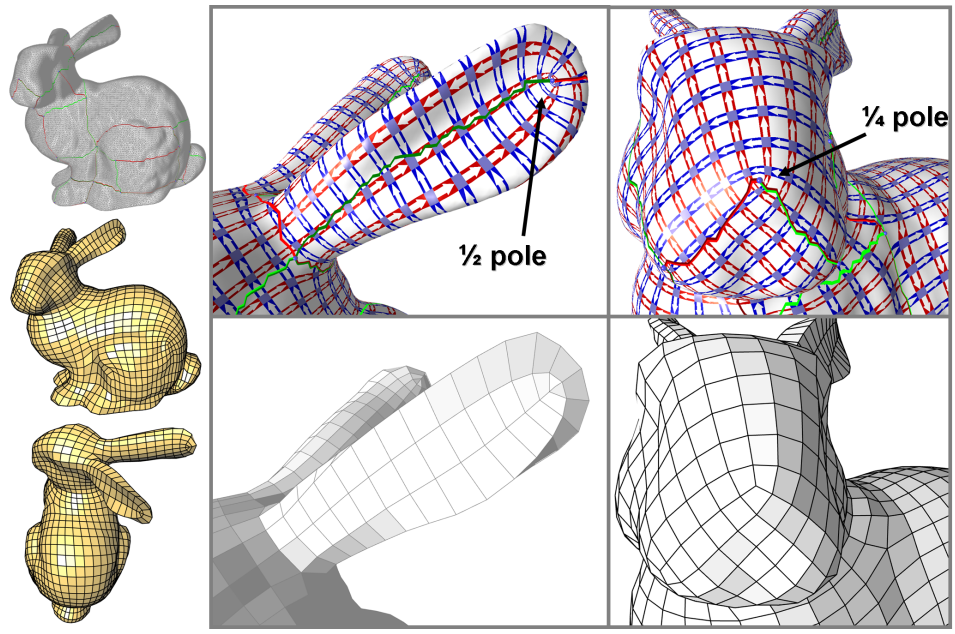



**Figure 2.34:** Stanford Bunny. Left: Singularity graph and quadrangle tiling. Middle: detail of a half-pole. Right: detail of a quarter-pole (using a switch and a regular line incident to the singularity). The half-pole becomes a degree-2 vertex, incident to two quads, with two nearly collinear edges. The degree-2 vertex can optionally be removed by merging its two incident quads into one.

Figure 2.35 illustrates the robustness of the results to an unprocessed input mesh, here very irregular and non-uniform.

Models of non-trivial topology are handled by designing singularity graphs whose edges cover the homology generators. Figure 2.36 depicts a genus-1 cup model with a small handle. The number of isolines per meta-edge of the singularity is chosen so as to generate a nearly uniform tiling. Figure 2.37

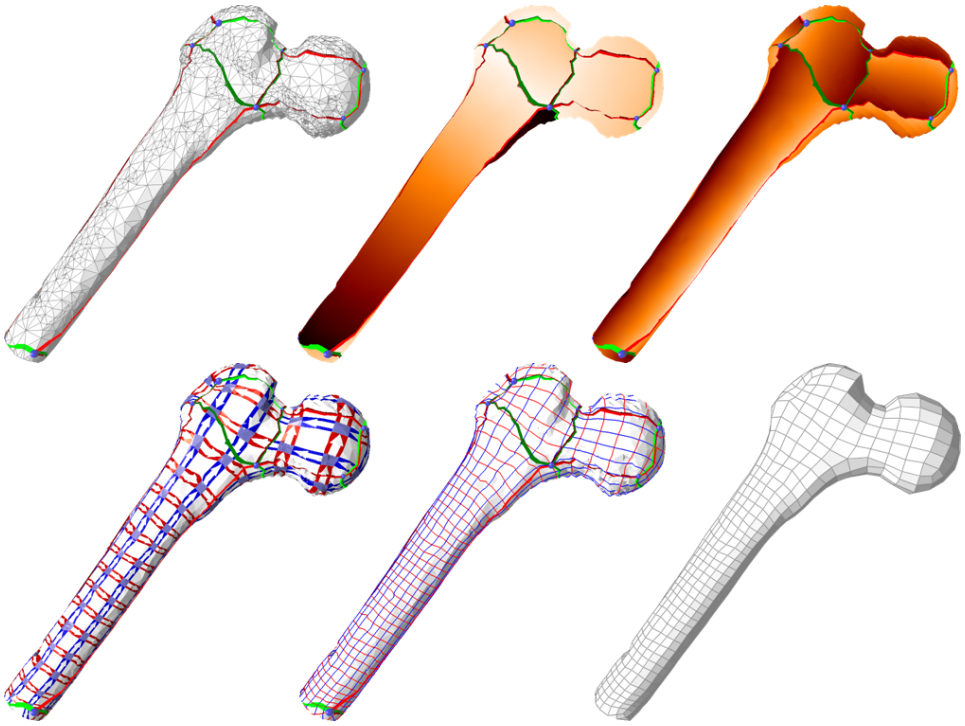


**Figure 2.35:** Femur. Top: input mesh and singularity graph, $u$ and $v$. Note the irregularity and non-uniformity of the mesh. Bottom: $du$ and $dv$ depicted with texture arrows, isocontouring lines and final quadrangle tiling.

depicts a genus-2 model. Note that although a saddle is present the final tiling is purely made out of quadrangles.

While the presence of saddles is sometimes unavoidable on certain manifold (as stated by the Hopf-Poincaré theorem), the user can minimize their distortion by altering the singularity graph: adding a meta-face over a saddle will split it into four lower-index saddles, reducing the effective distortion quite significantly at the price of three additional irregular vertices (see Figure 2.38).

Boundaries are handled as follows. Once the assignments of $(u, v)$ at each corner of the meta-faces are done, we go through the edges tagged as boundary between two such corners, and force the boundary values to be linearly interpolating the two corner values. This forces the isovalue of the potentials to follow the boundaries (see Figures 2.39 and 2.40).

**Figure 2.36:** Genus-1 model. Left: input triangle surface mesh with singularity graph covering the homology generators, and the final quadrangle tiling. Right, top: $u$, $v$. Right, middle: $du$ and $dv$. Right: bottom: iso-contouring lines, and $du - dv$.

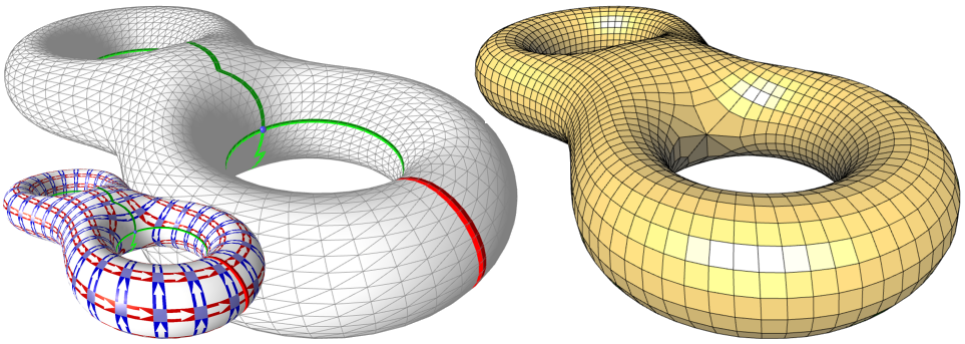


**Figure 2.37:** Genus-2 model: a saddle (imposed by Hopf-Poincaré theorem) is present on a meta-vertex. The final mesh is still pure quad. The singularity graph is defined by homology generators.

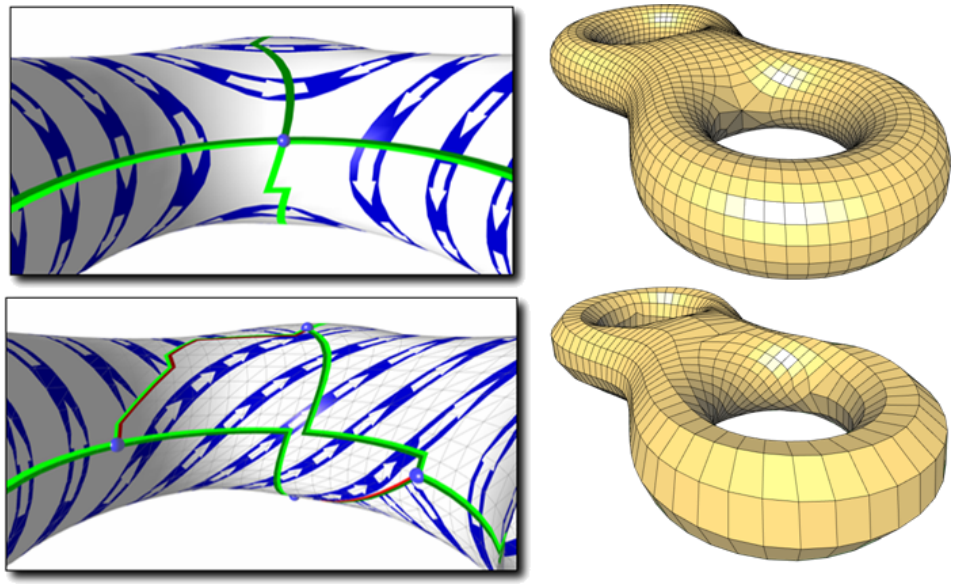


**Figure 2.38:** Splitting a saddle. A saddle (top) is turned into four lower-index saddles by altering the singularity graph.

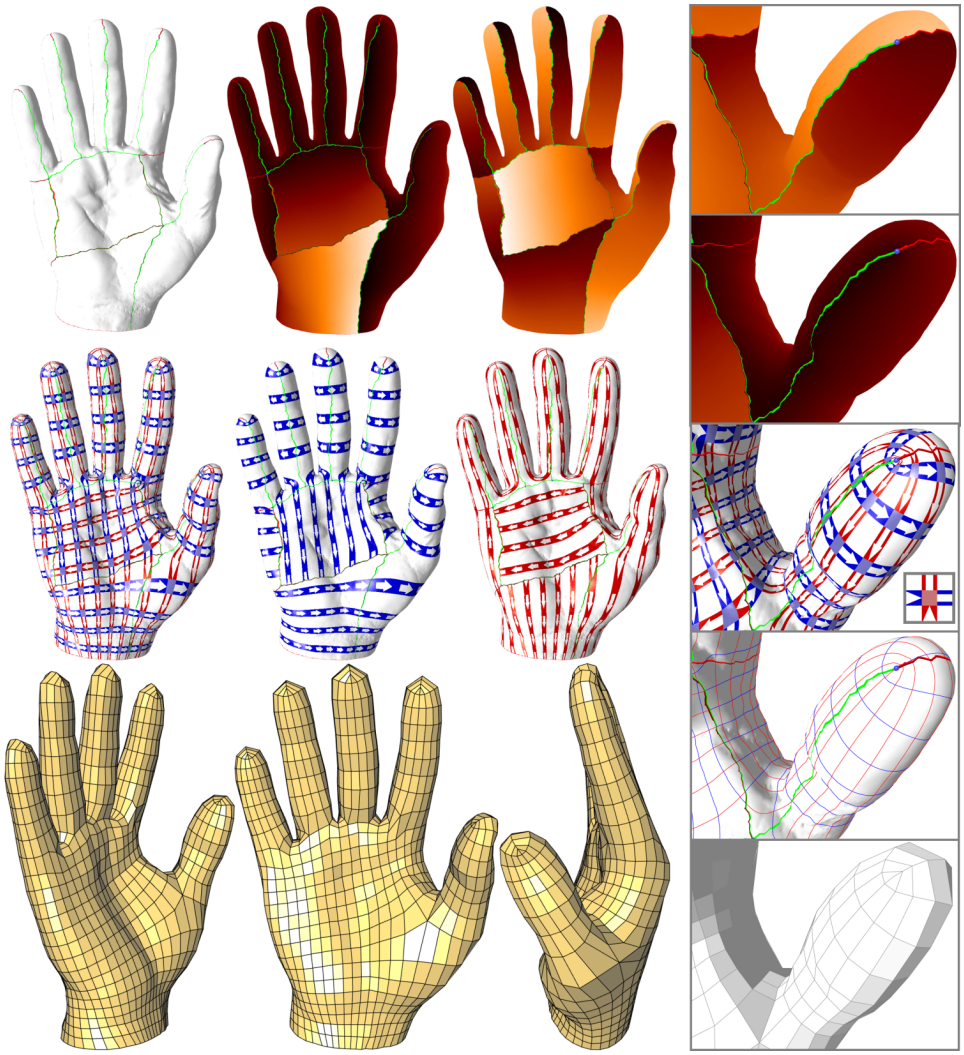


**Figure 2.39:** Scanned Hand. From a triangulated surface and a set of line singularities assembled into a singularity graph, our technique solves a linear, modified Laplace equation to get two potentials (top); The pair of 1-forms associated to the potential differentials is specified as either regular, reverse or switch across *singularity lines* (center). An isocontouring of these potentials results in a pure-quad mesh with non-integer index singularities capturing the geometry (see close-up, right), and no T-junction (bottom).

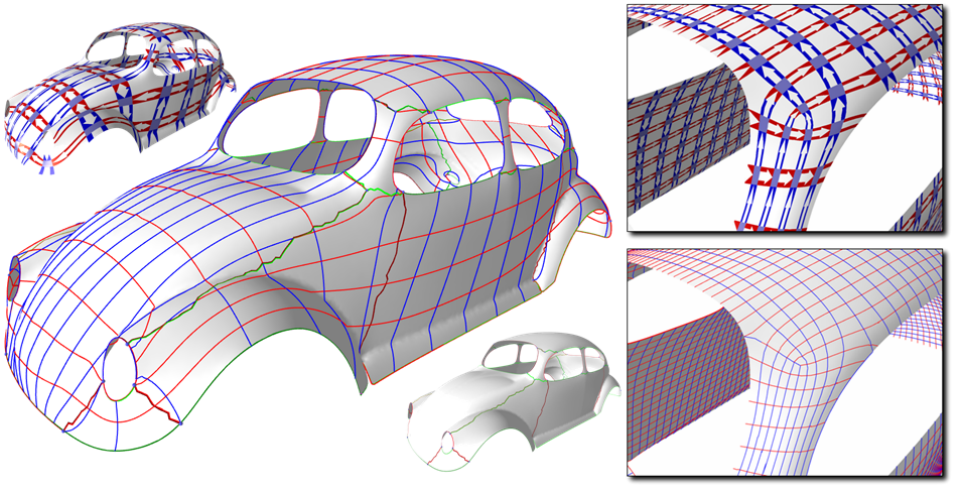


**Figure 2.40:** Beetle. A set of iso-contouring lines are constrained to be parallel to the boundary.

### 2.4.4 Summary

The approach presented above computes two piecewise smooth harmonic scalar functions, whose isolines tile the input surface into quadrangles, without any T-junctions. The main contribution is an extension of the discrete Laplace operator which encompasses several types of line singularities. The resulting two discrete differential 1-forms are either regular, opposite or switched along the singularity graph edges. We show that this modification guarantees the continuity of isolines across singularity lines, while the locations of the isolines themselves depend on the global solution to the modified Laplace equation over the whole surface. Design flexibility is provided through specification of the type of each line singularity of the graph, as well as the number of isolines along independent meta-edges to control quad sizes.

#### 2.4.4.1 Strengths

The main strengths of the method include a good control over orientation through line singularities, and a control over sizing through the number of isolines along independent meta-edges. In addition, the method is almost insensitive to the input tessellation due to its global, variational nature, as Figure 2.35 illustrates. Another appealing feature is the independence of iso-contouring lines to the geometry of singularity lines (meta-edges), as Figure 2.31 shows. This allows the user to focus only on the topology of the lines as well as on the placement of meta-vertices. Finally, the simplicity and scalability of computations (a linear solve followed by iso-contouring) is also a desirable feature. In our experiments we could solve up to a 2M triangle mesh on a 32bit computer with the Cholesky factorization from the TAUCS library.

#### 2.4.4.2 Weaknesses

The design part of the algorithm is interactive and not automatic. This is certainly the main weakness of this approach. Unlike previous work [GY03], we did not require our pair of 1-forms $du$ and $dv$ to be holomorphic (i.e., related through the Hodge star) to allow for more design flexibility. Therefore, an unreasonable choice of meta-parameters will result in regions where the quads are grossly non orthogonal. One way to palliate this issue is to find the meta-parameters so that the pair $(du, dv)$ minimizes (in the $\mathcal{L}^2$ sense) the 1-form $du - \star dv$ but we have not pushed this idea further. Also, the iso-contouring lines may not be smooth in the vicinity of a meta-vertex, as the values of the functions $u$ and $v$ are constrained there, contrary to along

a singularity line where only the direction and norm of $(du, dv)$ are constrained to match (see Figure 2.41). In some cases this can even translate into, e.g., a trisector (of index minus one half) splitting into a wedge (index one half) and saddle (index minus one). In our experiments relocating the meta-vertices often helps, but we have not devised a satisfactory automatic manner to perform such relocation. Finally, if the user decides to significantly override the values on meta-vertices automatically prescribed by our technique, the resulting meshes may have folds and significant stretch.

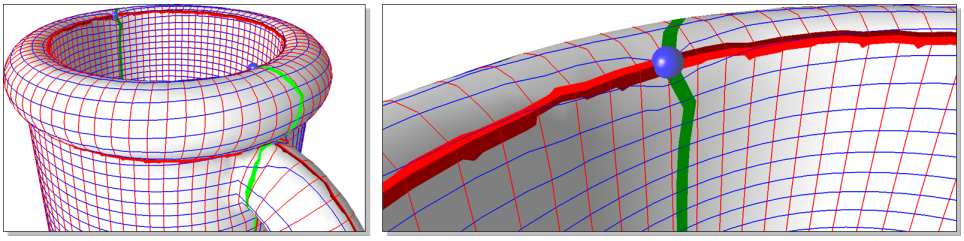


**Figure 2.41:** Cup. The iso-contouring lines may be not smooth in the vicinity of meta-vertices.

#### 2.4.4.3 Follow-ups

Recent related work include the QuadCover algorithm [KNP07] which computes frame fields based on branched covering spaces on a surface. In this approach branch points are singular points with fractional indices, and the equivalent of the singularity graph is a tree that goes through all singular points. The added value is the alignment of parameter lines with given vector fields.

#### 2.4.4.4 Future Work

As future work we intend to further investigate approaches which tile a surface through contouring scalar functions, be they harmonic or with other variational properties. The main reason for pursuing with such approach is the robustness of the contouring process, superior to most other greedy approaches in our belief.

One goal that we wish to pursue is to elaborate upon a variational formulation on top of our approach. The key idea is to consider the input to a variational formulation as a set of variables (both discrete and continuous), and to optimize these variables in order to optimize the real final objective. One example would be to consider as final objective an approximation error between the bilinear interpolation (or more involved NURBS) over the quadrangle tiling defined by a set of singularity lines and their associated

parameters. The optimization algorithm could refine the set of lines, relocate them, and optimized their associated parameters. We wish in essence to consider the singularity lines and parameters as generator of a unique quadrangle tiling, similar in spirit to a set of vertices and edges defining a unique 2D constrained Delaunay triangulation, except that the construction algorithm involves more numerical tools. Similar to what has been done for Delaunay refinement and mesh optimization, we wish to elaborate upon a set of local operators, driven by the end goal of the original problem. The ultimate surface tiling algorithm would simply take as input parameter an approximation tolerance and an interpolation scheme over the tiles (e.g., bilinear), and would minimize the number of tiles.

Another possible extension consists in changing the Hodge star $\star$. Altering the metric locally amounts to change the coefficient $w_{ij}$ used in the Laplacian operator. Theoretically speaking, the optimal quad shape for a best surface approximation derives from the curvature tensor in the asymptotic limit, but this asymptotic property may not be appropriate in the context of mesh coarsening and remeshing.

A recent non-linear approach [SSP08] provides strong theoretical guarantees on the resulting maps, based on a precise notion of discrete conformal equivalence for triangle meshes. As future work we wish to understand how these concepts can be used for surface tiling.

Finally, the last and most challenging tiling problem we intend to tackle is the automatic hexahedral tiling of 3D domains bounded by piecewise smooth surfaces. We intend to use again applied geometry tools (harmonic forms, discrete differential operators) onto an initial 3D triangulation of the input domain. The challenge is clearly not a simple extension of the presented approach with three harmonic functions.

## 2.5 Tetrahedron Mesh Generation

In this section we focus on isotropic tetrahedron mesh generation for 3D domains. Meshing a domain consists in defining a concise set of simple elements whose non-overlapping union best describes the domain and its boundaries, while satisfying a series of criteria on element shapes and sizes. Most ubiquitous in computer animation and computational sciences is the need for unstructured isotropic tetrahedral meshes: these versatile geometric representations are used in finite element and finite volume simulations of physical phenomena as varied as material deformation, heat transfer, and electromagnetic effects. As the accuracy and stability of such computational endeavors heavily depend on the shape of the worst element [She02a], mesh element quality is a priority when conceiving a mesh generation algorithm. In this paper we introduce a robust, hybrid meshing algorithm to generate high-quality, graded isotropic tetrahedron meshes. Delaunay refinements and variational optimizations are interleaved in order to produce a discretization of the domain that meets a series of desired geometric and topological criteria, while offering smooth gradation of the resulting well-shaped tetrahedra.

### 2.5.1 Related Work

Most previous work aimed at generating isotropic tetrahedral meshes were designed around one (or more) of four basic concepts: packing, regular lattices, refinement, and optimization. While packing methods (including advancing front approaches) were initially favored, their relatively high computational complexity and lack of theoretical guarantees have spawned the investigation of alternative methods. Regular lattices (be they red-green tessellations or octrees) have, recently, been at the core of some of the fastest meshing techniques, as they provide a blazingly fast approach to mesh most of the domain. While smooth surface boundaries can also be efficiently handled with guaranteed minimum dihedral angles [LS07], the regularity of the mesh resulting from these methods (*i.e.*, the presence of preferred edge directions) induces severe aliasing effects in simulation.

Techniques combining Delaunay triangulation and refinement have received special attention due to their versatility and theoretical foundations. They have been used initially in 2D [Che89], then in 3D for polyhedral domains [NCC02], for smooth surfaces [Che93], for 3D domains bounded by smooth surfaces [ORY05, BOG02] and for 3D domains bounded by piecewise smooth surfaces [RY07, CDL07, CDR07]. They proceed by refining and filtering a 3D triangulation until a set of user-specified criteria is satisfied. Refining is achieved through iterative insertion of Steiner points, either in-

side the domain or on the domain boundary, to meet the desired criteria. A filtering process is also iteratively applied to cull simplices so that the triangulation restricted to the input domain tessellates the domain and that the boundary of this restricted triangulation approximates the domain boundary. This procedure becomes more delicate for piecewise-smooth surface inputs (a more general class of shape where the boundary is a collection of smooth patches meeting at potentially sharp creases), as sharp creases require additional care. Non-smooth regions subtending small angles add yet another level of difficulty for Delaunay refinement. Refinement techniques are usually judged on the quality of the resulting mesh elements and the scarcity of Steiner point insertion.

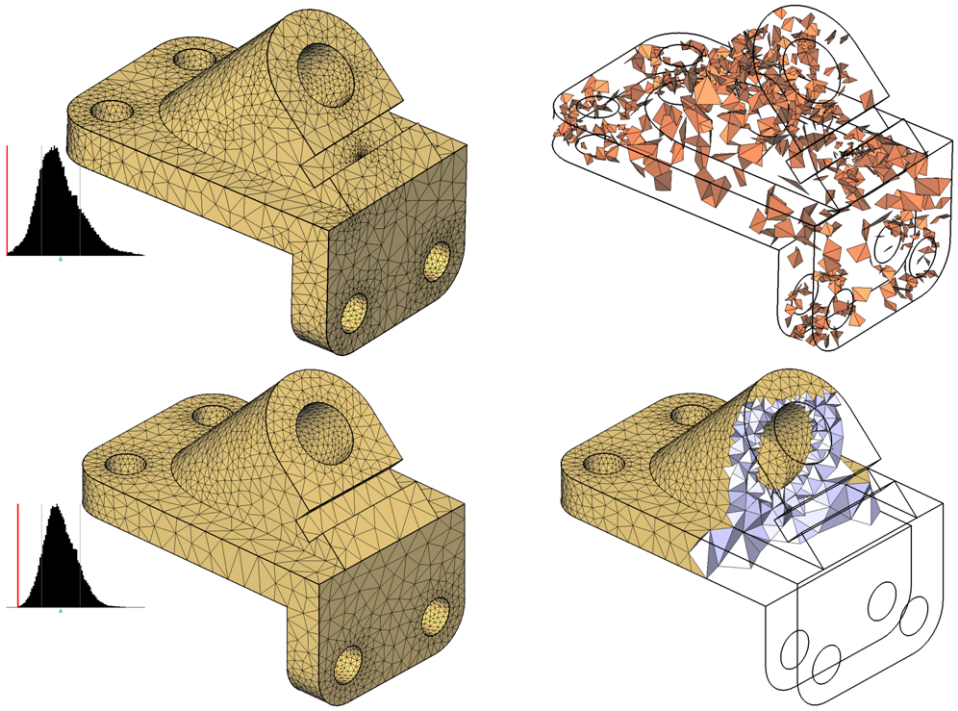


**Figure 2.42:** Top: Mesh generated by Delaunay refinement (shape and boundary approximation criteria activated). The mesh contains 5499 vertices, notice the cluster in the middle of the armhole. Right image depicts tetrahedra with dihedral angles lower than 15 degrees (all angles are in [0.45-179.1]). Bottom: Mesh generated by interleaving batches of Delaunay refinement and optimization so as to satisfy the same criteria. The mesh contains 3701 vertices and all dihedral angles are in [15.01-156.28]. Distribution of dihedral angles are shown on the left.

The quest for ever better quality meshes has also stimulated significant advancements in mesh optimization, through local vertex relocation to optimize a specific notion of mesh quality [ABE99], topological operations [CDE+00], or both [FOG97]. Further improvement of the mesh quality

can be achieved by inserting additional vertices, and/or incorporating a rollback mechanism to undo previous optimizations in order to guarantee a monotonous increase in mesh quality [KS07]. Among the large body of work in mesh optimization the *Optimal Delaunay Triangulation* approach (ODT for short) stands out, as it casts both geometric and topological mesh improvement as a single, unified functional optimization [CX04, Che04], that tries to minimize in $\mathbb{R}^4$ the volume between a paraboloid and the linear interpolation of the mesh vertices lifted onto the paraboloid. This approximation-theoretical method to obtain isotropic meshes was adapted for tetrahedral meshing of 3D domains [ACSYD05], mixed with a constrained Lloyd relaxation on the domain boundary. While this technique was shown to only produce nicely-shaped tetrahedra throughout the domain, slivers (i.e., nearly degenerate elements) could appear near the domain boundary, as the boundary vertices were guided by Lloyd relaxation and thus unaffected by the 3D optimization. Furthermore, this method lacks a number of useful features. First, the algorithm is not designed to satisfy the type of user-defined criteria commonly handled by Delaunay-based mesh generation techniques [RY07]. Also, an estimate of the boundary local feature size is required to derive a sizing function; such estimate turns out not to be always reliable depending on the tessellation of the input polyhedral domain boundary. Finally, this method cannot handle arbitrary boundary meshes as input, requiring a *restricted* Delaunay triangulation instead.

### 2.5.2 Approach

We combine the efficacy of Delaunay refinement methods with the isotropic quality induced by optimal Delaunay optimization techniques to provide a practical, high-quality meshing algorithm for domains bounded by piecewise smooth boundaries. This combination of techniques is motivated by the desire to maximize mesh quality while reducing mesh size. Delaunay refinement alone tends to generate overly complex meshes, with, e.g., spurious clusters of vertices due both to the greedy nature of the algorithm and to the encroachment mechanisms; interleaving parsimonious refinement and mesh optimization instead turns out both to reduce the number of Steiner points and to improve the overall mesh quality (see Figure 2.42).

Unlike previous mesh optimization methods which either consider the boundary fixed or use boundary conditions incompatible with global mesh improvement, we introduce a consistent, unifying variational treatment applied to both interior and boundary nodes, improving the overall quality of the mesh. To speed up Delaunay refinement and make it parsimonious, we pick subsets of isolated Steiner points using the probabilistic multiple choice approach [WK02, VLV+04] to reduce the occurrence of short-lived

primitives and provide independent refinements before each round of optimization. The practicality of our approach further stems from additional, distinctive features. First, we only rely on simple intersection tests to probe the domain boundary to make the approach as generic as possible with respect to the boundary surface representation. Second, we do not require a mesh sizing function as input and provide instead a dynamic sizing function which evolves throughout refinement until all user-specified criteria are satisfied. Finally, while we present a concrete implementation of our approach, the method is versatile enough to serve as a general framework for isotropic tetrahedron meshing, as each step involved in the process can be adapted to special requirements.

The algorithm we now detail interleaves refinement and optimization of an initial 3D Delaunay triangulation. Mesh simplices are gradually improved to meet user-defined criteria on boundary approximation and on the shape and size of elements through refinements, while passes of optimization further improve the shape of the elements. The high-level pseudo-code is as follows:

**Algorithm 1** Mesh generation at a glance

**Require:** *Domain* $\Omega \in \mathbb{R}^3$ *(Section 2.5.2.1) and a set* $\{k_1, k_2, ..., k_n\}$ *of user-defined criteria (Section 2.5.2.2).*
Initialize coarse mesh $\mathcal{M}$ (Section 2.5.2.3)
**while** Refine through sparse vertex insertions (Section 2.5.2.4) **do**
Optimize mesh (Section 2.5.2.8)
**end while**
Remove leftover slivers (Section 2.5.2.9)

The refinement procedure inserts Steiner points so as to satisfy the criteria $\{k_1, k_2, ..., k_n\}$. It returns true if at least one Steiner point has been inserted and false otherwise, so that the mesh is further optimized only when refined. This way, the algorithm benefits from the same guarantee of common Delaunay refinement algorithms.

#### 2.5.2.1 Input

The input is a 3D domain $\Omega$ whose boundary is defined as a piecewise smooth complex (PSC). More specifically, our current implementation takes as input a piecewise linear approximation of a PSC. The latter is provided as a triangle surface mesh, watertight, and forming a 2-manifold with no self-intersection. In addition, we assume that sharp edges as well as feature vertices of this mesh are tagged. Dart (resp., corner) vertices are easily deduced from tagged sharp edges as they are incident to one (resp., three or

more) sharp edges. Tip and cusp vertices, which are incident respectively to zero and two sharp edges, cannot be derived solely using the sharp edge tags and hence must be specified by the user. By chaining sharp edges together, we obtain a set of polylines that we will refer to as creases from now on. A crease may either connect two feature vertices, or form a cycle. All creases are enumerated, and each sharp edge of the input surface mesh is marked with the index of its associated crease. Finally, we identify and enumerate surface patches as connected components of the boundary, bounded (or not) by sharp creases. Each face of the input surface mesh is marked with the index of its associated patch as depicted in Figure 2.43.

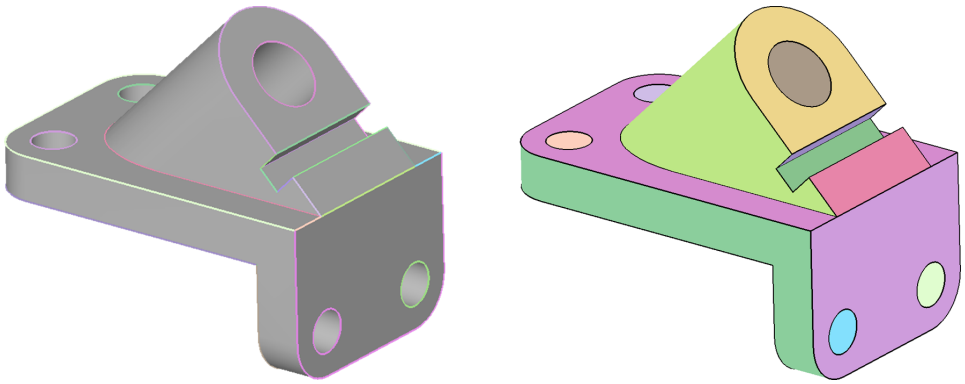


**Figure 2.43:** Input PSC with sharp creases and surface patches marked.

#### 2.5.2.2 Parameters

The user can also input a number of desired criteria that the final mesh must satisfy. These criteria will be used to guide the refinement process as explained in Section 2.5.2.4. Our meshing framework can handle five types of criteria:

- *Sizing:* a spatially-varying sizing function (or possibly a single value if constant) indicates the maximum mesh edge length desired within the domain.
- *Approximation:* an approximation control function defines a local upper bound for the surface or crease approximation error, $\epsilon_{\max}$. Similarly to the mesh sizing function, it can be defined as a single value if the function is constant over the boundary, or as a spatially-varying scalar function.
- *Shape:* two global element shape quality bounds are defined as the maximum circumradius to shortest edge ratio allowed in the final mesh.

We denote by ${\sigma^f}_{\max}$ and ${\sigma^t}_{\max}$ these bounds for facets and tetrahedra, respectively.

- *Topology:* a Boolean flag determines whether the topology of the input PSC should be preserved, *i.e.*, if the vertices of each restricted facet must belong to the same patch, and the vertices of each restricted edge must belong to the same crease.

- *Manifold:* a Boolean flag determines whether the final mesh boundary should be a two-manifold surface.

These criteria accommodate the typical user requirements for mesh generation. Note that they are all optional in our implementation, except for the *sizing* field.

#### 2.5.2.3 Initialization

A first mesh $\mathcal{M}$ of the domain is obtained as follows. We begin by inserting in $\mathcal{M}$ all *feature vertices* (corners and such) of the input surface mesh. These vertices remain untouched throughout the mesh generation procedure. We also add the eight corners of a large bounding box of the input domain, in order not to have to deal with infinite Voronoi cells in later stages. Finally, we ensure that each surface patch and each crease have received the minimal number of sample points to seed the refinement process by adding more vertices if necessary, as in [RY07]. The mesh $\mathcal{M}$ is defined to be the Delaunay mesh of all these vertices. Finally, we refine this initial mesh with respect to looser criteria than the ones defined by the user (typically, we halve the various input criteria parameters), using our refinement procedure that we detail next.

#### 2.5.2.4 Refinement

The refinement process is entirely driven by the user-defined criteria listed in Section 2.5.2.2. Each refinement step is designed to remove a set of *bad elements* (simplices that do not satisfy at least one of the given criteria) by inserting so-called *Steiner vertices* to $\mathcal{M}$. Unlike typical Delaunay refinement techniques which insert one Steiner point at a time, we proceed in batches of refinement, inserting a sparse subset of all the candidate Steiner points per batch.

#### 2.5.2.5 Bad Elements

The simplices considered for refinement are the so-called *restricted simplices*, that is, the ones considered as inside the domain $\Omega$ or on the domain boundary $\partial\Omega$—namely, edges whose dual Voronoi facet intersects an input crease, facets whose dual Voronoi edge intersects the domain boundary, and tetrahedra whose dual Voronoi vertex is located inside the domain. We consider one of these restricted elements bad if it violates one of the following criteria.

*Size.* A restricted edge is considered bad if it is longer than the sizing function evaluated at its midpoint. A restricted facet or tetrahedron is considered bad if at least one of its edges is badly sized.

*Approximation error.* A restricted edge $e$ is considered bad if the distance from its midpoint to the farthest intersection point between its dual Voronoi face and an input crease is larger than the local approximation bound. Similarly, a restricted facet $f$ is considered bad if the distance from $f$'s circumcenter to the farthest intersection point between its dual Voronoi edge and the domain boundary is larger than the approximation bound.

*Shape.* A restricted facet (resp., tetrahedron) is considered bad if the ratio of its circumradius to shortest edge is higher than the user-specified bound $\sigma^f_{\max}$ (resp., $\sigma^t_{\max}$).

*Topology.* A restricted edge (resp., facet) is considered as not capturing the proper topology if its two (resp., three) vertices do not belong to the same input crease (resp., surface patch). If the topology criterion is activated, we store for each vertex $v$ of the mesh its location with respect to the input PSC. That is, each vertex is tagged either as an *interior* vertex, a *feature* vertex, a *crease* vertex, or a *boundary* vertex. In the last two cases, the index of the feature (crease or surface patch) is stored too.

In addition to these types of bad elements, we add an extra one to enforce the *topological disk condition* [RY07] as it is an important indicator of topological conformity of the mesh to the input domain. For a vertex $v$ tagged as boundary (*i.e.*, on an input surface patch), the topological disk condition is satisfied iff the boundary facets incident to $v$ form a topological 2-disk. If $v$ belongs to an input crease, its incident restricted edges (edges whose dual Voronoi facet intersects input creases) have to form a topological 1-disk. We thus mark every boundary vertex of the mesh whose topological disk condition is not satisfied as bad as well.

#### 2.5.2.6 Steiner Vertices

For each bad simplex, we define its associate Steiner point location. The associated Steiner point to a restricted *edge* is the farthest intersection point between its dual Voronoi face and the input creases. The associated Steiner point to a restricted *facet* is the farthest intersection point between its dual Voronoi edge and the input domain boundary. The associated Steiner point to a restricted tetrahedron is its circumcenter. Finally, for each boundary vertex of the mesh whose topological disk condition is not satisfied, we define its associated Steiner point to be the Steiner point of the facet (resp., crease edge) incident to $v$ that realizes the largest approximation error: its insertion will help enforcing the topological disk condition.

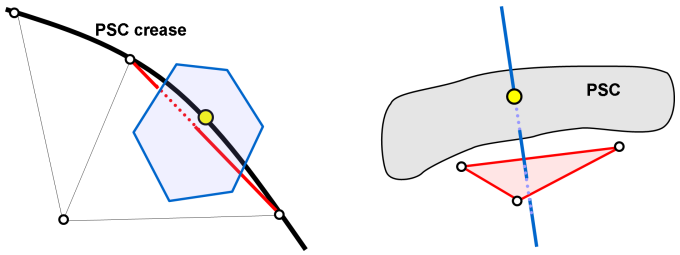



**Figure 2.44:** Left: Steiner point (yellow) of a restricted edge (red) computed as the furthest intersection point of its dual Voronoi facet (blue) with the input PSC creases. Right: Steiner point (yellow) of a restricted facet (red) computed as the furthest intersection point of its dual Voronoi edge (blue) with the input PSC surface.

To ensure termination of the refinement process, we further check for *encroachment* [Rup95, She98, She02b, CDL07, RY07]. The Steiner point $p$ of a tetrahedron, candidate for insertion, is said to *encroach* a boundary facet $f$ if it is inside its restricted Delaunay ball (centered at $f$'s Steiner point and passing through the vertices of $f$). Similarly, the Steiner point of a facet is said to *encroach* a crease edge if it is inside its restricted Delaunay ball (centered at its Steiner point and passing through its endpoints). In these two cases of encroachment, we alter the position of the associated Steiner point, replacing it by the Steiner point of the encroached primitive (and recursing the encroachment check).

#### 2.5.2.7 Independent Set Refinement

To help define a good subset of Steiner points to add in batch, we introduce the notion of conflict regions and independent set of conflict regions. For each Steiner point $p$, we call "conflict region" the tets that would be affected by its insertion as well as their adjacent tets, since these elements are likely

to be destroyed by the insertion of $p$. We call "an independent set" of conflict regions a set that does not contain overlapping conflict regions, so that none of the insertions of these selected Steiner points would influence each other. We construct such an independent set of conflict regions: we iteratively select Steiner points in order of increasing dimension of their associated simplices. That is, first crease edges are collected and sorted from worst to best. As many crease-edge Steiner points as possible are inserted into the set, along with their conflict region, while making sure there is no overlap of conflict regions. Second, we similarly treat boundary facets. Finally, tetrahedra are handled; however, as there can be a large amount of bad tets during the meshing process, the same process of sorting elements before choosing them would be too costly. We therefore process bad tetrahedra through a more efficient Multiple-Choice approach as explained next, and this is done iteratively until no Steiner point can be inserted to the independent set without overlapping the regions already inserted. Inset shows an independent set on the mesh of a cylinder for which only the approximation criterion is not yet satisfied.

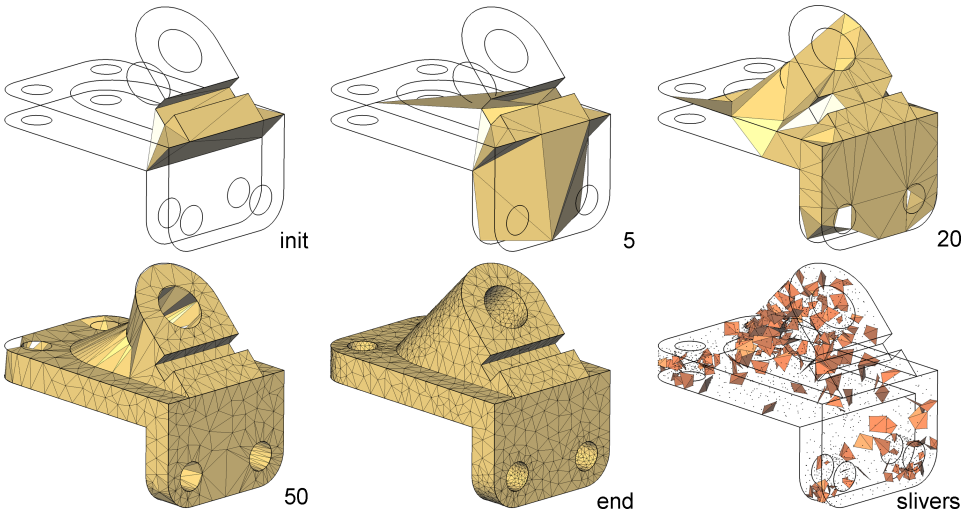



**Figure 2.45:** Refinement steps. From left to right and top to bottom: The mesh initialized with feature vertices; after a few batch refinement steps (from 5 to 50); the final refined mesh with shape and approximation criteria satisfied; and its 244 slivers (tetrahedra with dihedral angle smaller than 10 degrees).

**Multiple-Choice Selection of Tetrahedra** Although many Delaunay refinement algorithms use modifiable priority queues to store all bad simplices of the mesh $\mathcal{M}$, most queue elements are short-lived as each Steiner position insertion affects its surrounding. In fact, our experiments consistently showed that the computational burden spent maintaining the global priority queue of all bad simplices is overly high compared to the number of

primitives actually refined. We thus depart from the usual refinement strategy by using a multiple choice approach (proposed for mesh decimation in [WK02, VLV+04]) as follows. At each step, a small container of $N_{mc}$"bad" tetrahedra ($N_{mc}$= 20 in our implementation), randomly updated among the non-conflicted tetrahedra, is used to select the worse tetrahedron. As our goal is to only sparingly refine the mesh before further optimization, this multiple-choice approach significantly speeds up our refinement process.

#### 2.5.2.8 Optimization

Chen [Che04] defines an *Optimal Delaunay Triangulation (ODT)* as the minimizer of the energy

$$E_{\text{ODT}} = ||f_{\text{PWL}} - f||_{\mathcal{L}^1} = \sum_j \int_{T_j} |f_{\text{PWL}} - f|,$$

where $f(\mathbf{x}) = ||\mathbf{x}||^2$ and $f_{\text{PWL}}$ is the linear function interpolating the values of $f$ at the vertices of the tets $T_j$'s of the triangulation. This energy has a simple geometric interpretation: it is the volume between the 4D paraboloid (defined by $f$ and its inscribed piecewise linear approximation $f_{\text{PWL}}$ through lifting off the triangulation onto the paraboloid [BWY07]. Because of a result of function approximation theory stating that the best interpolating approximation of a function is achieved when the elements' size and orientation match the Hessian of the function, an ODT is thus isotropic (see Figure 2.46).

By integrating this function over each tet and summing all the contributions, one gets:

$$E_{\text{ODT}} = \frac{1}{4} \sum_{x_i \in T_j} {\mathbf{x}_i}^2 |\Omega_i| - \int_{\mathcal{M}} \mathbf{x}^2 d\mathbf{x}, \tag{2.4}$$

where $|\Omega_i|$ is the volume of the 1-ring neighborhood of vertex $\mathbf{x}_i$. Noting that the last term is constant given a fixed boundary $\partial\mathcal{M}$, a simple derivation of this quadratic energy in $\mathbf{x}_i$ leads to the following *optimal position* $\mathbf{x}_i^\star$ of the interior vertex $\mathbf{x}_i$ in its 1-ring [Che04]:

$$\mathbf{x}_i^\star = -\frac{1}{2\,|\Omega_i|} \sum_{T_j \in \Omega_i} \left( \nabla_{\mathbf{x}_i} |T_j| \left[ \sum_{\substack{\mathbf{x}_k \in T_j \\ \mathbf{x}_k \neq \mathbf{x}_i}} ||\mathbf{x}_k||^2 \right] \right). \tag{2.5}$$

The term $\nabla_{\mathbf{x}_i}|T_j|$ is the gradient of the volume of the tet $T_j$ with respect to $\mathbf{x}_i$. Replacing the paraboloid function $f(\mathbf{x}) = ||\mathbf{x}||^2$ by the translated

function $f(\mathbf{x}) = ||\mathbf{x}-\mathbf{x}_i||^2$, does not change the interpolation error, leading to the same optimal position. We thus get the following equivalent expression used to update a vertex position:

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2\,|\Omega_i|} \sum_{T_j \in \Omega_i} \left( \nabla_{\mathbf{x}_i} |T_j| \left[ \sum_{\mathbf{x}_k \in T_j} ||\mathbf{x}_i - \mathbf{x}_k||^2 \right] \right). \tag{2.6}$$

We also know that $\sum_{T_j \in \Omega_i} \nabla_{\mathbf{x}_i} |T_j| = 0$, it thus follows that when all $||\mathbf{x}_i - \mathbf{x}_k||^2$ are equal, $\mathbf{x}_i^\star = \mathbf{x}_i$. In other words, when the neighbors of $\mathbf{x}_i$ lie on a sphere with center $\mathbf{c}$, $\mathbf{x}_i^\star = \mathbf{c}$; we call this property the *ODT circumsphere property*.

As a special case of this property, the optimal position of a vertex that has only four neighbors is exactly *at* $\mathbf{c}_T$. Using Equation (2.6) in this special case of a 1-ring in the shape of a tet $T = (\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r, \mathbf{x}_s)$, and taking the point $\mathbf{x}_i$ to be located at $\mathbf{x}_p$, we get:

$$\begin{aligned} \mathbf{c}_T = \mathbf{x}_p - \frac{1}{2\,|T|} \Bigg[ & \nabla_{\mathbf{x}_p} |T| \left[ \sum_{\mathbf{x}_k \in T} ||\mathbf{x}_p - \mathbf{x}_k||^2 \right] \\ & + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r) + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_s) + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_r, \mathbf{x}_s) \Bigg] \end{aligned} \tag{2.7}$$

where the extra terms on the rhs only depend on each face of the tet (because, as we took $\mathbf{x}_i$ to be at $\mathbf{x}_p$, all but one of the tets inside $T$ are degenerate and become *faces* of $T$). More precisely, these terms are explicitly given as:

$$\mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r) = +\frac{1}{3} \Big[ ||\mathbf{x}_p - \mathbf{x}_q||^2 + ||\mathbf{x}_p - \mathbf{x}_r||^2 \Big] \mathbf{N}_{p,q,r}$$

where $\mathbf{N}_{p,q,r}$ is the area-weighted normal of the face $(p, q, r)$ pointing towards the inside of the tet, *i.e.*, $\mathbf{N}_{p,q,r} = |(p, q, r)|\mathbf{n}_{p,q,r}$.. Now, go back to Equation 2.6 for an arbitrary 1-ring centered on $\mathbf{x}_p$, and note that the term in parenthesis appears as is (for $p \equiv i$) in Equation 2.7. Substitute this term by the circumcenter and all the other terms that Equation 2.7 contains. All the face terms $F$ cancel each other out, thus simplifying the expression to:

$$\mathbf{x}_i^\star = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| \mathbf{c}_j. \tag{2.8}$$

**Natural ODT for Boundary Vertices** While [Che04, ACSYD05] do not involve the boundary vertices in the minimization of the ODT energy, we propose an extension that changes the update of boundary vertices during optimization so as to further reduce the total energy, thus providing a

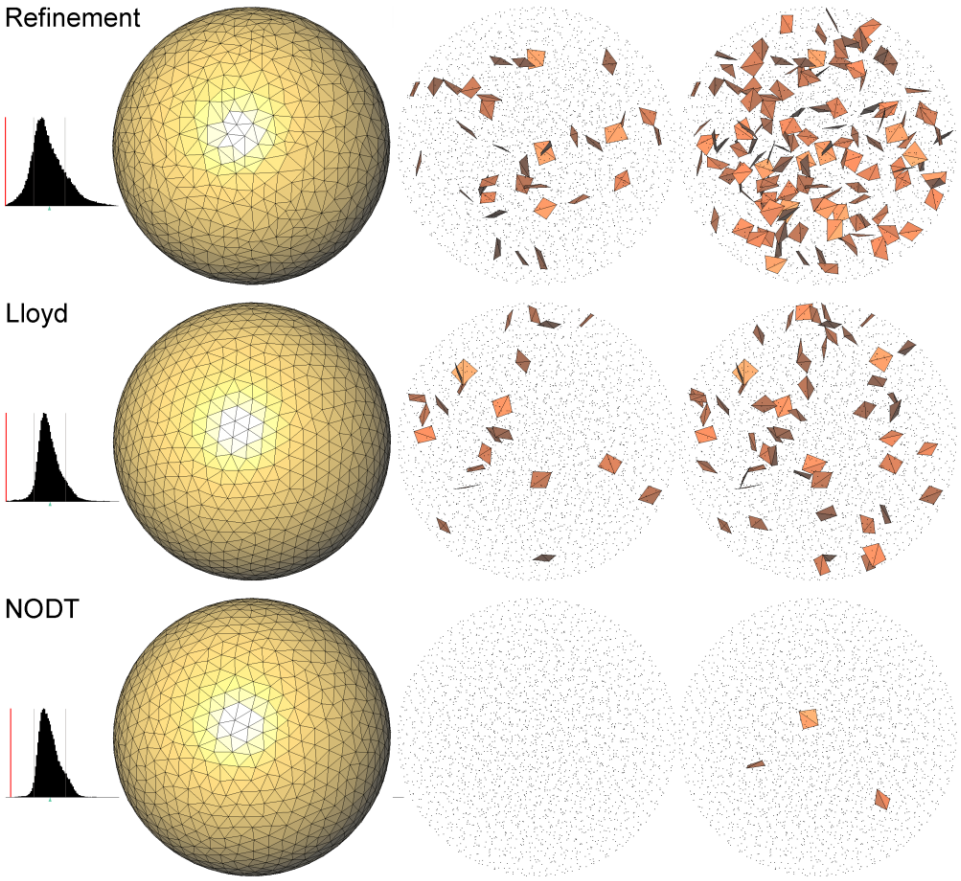



**Figure 2.46:** Comparing Delaunay refinement and mesh optimization. Distributions of dihedral angles are shown to the left. Slivers are shown for a dihedral angle bound of respectively 5 (middle) and 10 degrees (right). Top: Delaunay Refinement alone (resp. 35 and 136 slivers). Middle: Optimized mesh with 100 Lloyd iterations (resp. 23 and 55 slivers). Bottom: Optimized mesh with 100 NODT iterations (resp. 0 and 3 slivers).

boundary extension to the original ODT mesh smoothing procedure. Denote by $\mathbf{x}_p$ a vertex on the *boundary* of a 3D mesh (i.e., it does not have a full 1-ring $\mathcal{N}(\mathbf{x}_p)$ of restricted (inside) tetrahedra. For a given connectivity, the new position $\mathbf{x}_p^\star$ of $\mathbf{x}_p$ that extremizes the ODT energy is a bit more complicated, as some of the face terms $F$ do *not* disappear:

$$\mathbf{x}_p^\star = \left[ \left( \sum_{T \in \mathcal{N}(\mathbf{x}_p)} |T| \mathbf{c}_T \right) + B \right] / \sum_{T \in \mathcal{N}(\mathbf{x}_p)} |T|,$$

where the boundary terms $B$ are

$$B = \frac{1}{6}\left(\sum_{(p,q,r)\in\partial\mathcal{M}} \mathbf{N}_{p,q,r}\left[||\mathbf{x}_p - \mathbf{x}_q||^2 + ||\mathbf{x}_p - \mathbf{x}_r||^2\right]\right).$$

(Again, by $\mathbf{N}_{p,q,r}$, we mean the vector that is along the normal of the face (pointing toward the inside of the domain), with a magnitude equal to the area of the triangle.) The first part ($|T|\mathbf{c}_T$) is the weighted barycenter of the circumcenters and is divided by the total 1-ring volume just as before. Note however that there is an extra term: a sum on triangles $(p, q, r)$ that are on the boundary of the domain, involving the squared length of the "spokes" of the triangle 1-ring.

This formula, applied as is, shrinks the domain as it obviously decreases the total energy. However, as seen previously, we can add a multiplicative weight to the supplementary terms $B$ without changing the update rule in the case of a full 1-ring, because they cancel each other out anyway:

$$\mathbf{x}_p^\star = \left[\left(\sum_{T\in\mathcal{N}(\mathbf{x}_p)} |T|\mathbf{c}_T\right) + \lambda B\right] / \sum_{T\in\mathcal{N}(\mathbf{x}_p)} |T|.$$

We now use this flexibility to choose $\lambda$ so that we retain the *ODT circumsphere property* mentioned earlier, but now in the case of an incomplete 1-ring: if all neighbors of $\mathbf{x}_p$ are at the same distance from $\mathbf{x}_p$, we want $\mathbf{x}_p^\star = \mathbf{x}_p$: we will thus obtain a formula valid for both the complete 1-ring and incomplete 1-ring cases, while preserving the ODT circumsphere property. We have

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2|\Omega_i|}\sum_{T_j\in\Omega_i}\left(\nabla_{\mathbf{x}_i}|T_j|\Big[\sum_{\mathbf{x}_k\in T_j}||\mathbf{x}_i - \mathbf{x}_k||^2\Big] + (1-\lambda)\sum_{r,s}\mathrm{F}(\mathbf{x}_i, \mathbf{x}_r, \mathbf{x}_s)\right). \tag{2.9}$$

Consider the case where all $||\mathbf{x}_i - \mathbf{x}_k||^2$ are equal to some constant $R$. We want $\mathbf{x}_i^\star = \mathbf{x}_i$ and we know, from the divergence theorem applied on the 1-ring of the boundary vertex, that

$$\nabla_{\mathbf{x}_i}|T_j| + \sum_{r,s}\frac{1}{3}\mathbf{N}_{p,q,r} = 0.$$

On the one hand, $\nabla_{\mathbf{x}_i}|T_j|$ is weighted by $3R$ in (2.9). On the other hand, each $\frac{1}{3}\mathbf{N}_{p,q,r}$ is weighted by $2R$ in (2.9). It follows that we need $(1-\lambda) = 3/2$ for each term

$$\left(\nabla_{\mathbf{x}_i}|T_j| \left[\sum_{\mathbf{x}_k \in T_j} ||\mathbf{x}_i - \mathbf{x}_k||^2\right] + (1-\lambda)\sum_{r,s} \mathrm{F}(\mathbf{x}_i, \mathbf{x}_r, \mathbf{x}_s)\right)$$

to vanish.

Hence, for $\lambda = -1/2$, the ODT circumcenter property is still enforced on the boundary. In particular, the optimal position for this method (denoted NODT for *Natural ODT*) is computed as

$$\mathbf{x}_p^\star = \left[\left(\sum_{T \in \mathcal{N}(\mathbf{x}_p)} |T|\mathbf{c}_T\right) - \frac{1}{2}B\right] / \sum_{T \in \mathcal{N}(\mathbf{x}_p)} |T|,$$

where the boundary terms $B$ (if any) are

$$B = \frac{1}{6}\left(\sum_{(p,q,r)\in\partial\mathcal{M}} \mathbf{N}_{p,q,r}\left[||\mathbf{x}_p - \mathbf{x}_q||^2 + ||\mathbf{x}_p - \mathbf{x}_r||^2\right]\right).$$

**Variable Sizing** The optimization formula above is only valid to generate uniform isotropic meshes. To account for a variable mesh sizing, we update a dynamic mesh sizing function after each batch of refinement, and replace all measures in above formulas (lengths, areas, volumes) by measures in the metric of this sizing function. Such measures are obtained by quadratures over the mesh elements. The dynamic sizing function [ADA07] is guaranteed to be $K$-Lipschitz and is obtained by averaging the lengths of the mesh edges incident to all mesh vertices. Intuitively, the refinement is in charge of discovering the local feature size of the domain boundary. One of the user-defined criteria triggers a local refinement of the mesh, which induces the updating of the sizing function which, in turn, imposes further refinements to maintain the $K$-grading of the mesh. The optimization part of the algorithm then takes the current sizing function as input, so that it does not smooth out the increased density brought by the refinement steps to satisfy the user-defined criteria.

**Restriction and Projection** In practice, as we want the mesh to interpolate the domain, each boundary vertex of the mesh should be on the boundary $\partial\Omega$ of the PSC domain. To enforce this property, the new location $\mathbf{x}_p^\star$ of $\mathbf{x}_p$ is projected onto $\partial\Omega$. Two cases are distinguished: $\mathbf{x}_p^\star$ can belong to a surface patch, or to a sharp feature of the mesh. If at least one of the

incident edges to $\mathbf{x}_p$ is a *crease edge* (i.e., its dual Voronoi facet intersects a PSC crease), then we project $\mathbf{x}_p^\star$ onto the closest crease. Similarly, if at least one of the incident facets to $\mathbf{x}_p$ is a boundary facet (i.e., its dual Voronoi edge intersects the PSC), we project $\mathbf{x}_p^\star$ onto the closest facet of the input PSC.

#### 2.5.2.9 Sliver Removal

While our NODT boundary treatment significantly reduces the number of slivers compared to the results reported in [ACSYD05], we cannot guarantee a total absence of slivers (see Figure 2.47). We thus perform a final phase of sliver removal. We implemented an explicit perturbation inspired by [Li00], which performed better and faster than sliver exudation [CDE+00] in our experiments. This method applies repeatedly a small perturbation of each vertex incident to slivers. Vertices located on sharp creases or boundary are then reprojected onto their respective crease or boundary. This relocation is validated if it both reduces its number of incident slivers and preserves the restricted triangulation locally. This process is iterated a number of times (max. 100 in our implementation) for each vertex incident to one or more slivers, and interrupted earlier if all incident slivers are removed. Note that for completeness we have also applied such perturbation after Delaunay refinement and Lloyd-based optimization. In these meshes many slivers remain due to the presence of chains of slivers, i.e., several slivers incident to each other.

#### 2.5.2.10 Implementation

The algorithm is implemented using the Computational Geometry Algorithms Library CGAL [CGA09]. We use its 3D regular (weighted Delaunay) triangulation as our core data structure. The input PSC is represented as a surface triangle mesh with attributes. One crucial component for reaching good timings is the efficient update of the restricted triangulation and Steiner points: this requires many intersection tests between rays and Voronoi edges and the input domain boundary, as well as intersections between Voronoi faces and the input sharp creases. We have implemented a collision detection library based on the principles used in OPCODE [Ter05]. Two hierarchies of axis-aligned-bounding-boxes (AABBs) are created right after loading the input PSC: one for the PSC triangle facets and one for the PSC segment sharp creases. Each intersection query (be it a test or an exhaustive enumeration) then calls intersection with AABBs during traversal, and intersection with PSC primitives (triangle or crease segments) at the leaves of the tree (see [dB05]). In addition, the same AABB trees are used

for projecting the optimized boundary vertices onto the domain boundary or creases. The trees are this time queried with 3D balls whose radius decreases during the tree traversal. In our tests, these hierarchies of AABBs result in the fastest intersection and projection routines on average, compared to object-aligned bounding boxes and KD-trees. Finally, we also speed up the NODT procedure through a locking process. We lock up (i.e., deactivate the optimization of) all mesh vertices which are incident to only excellent restricted tetrahedra. A tetrahedron is defined as excellent when all its dihedral angles are within a user-specified interval (typically [45-95]). Only the vertices newly inserted during refinement or relocated during optimization are allowed to unlock their incident vertices. Consequently, entire parts of the mesh which do not need to be improved either by refinement or by optimization are skipped throughout the refinement/optimization alternation. Tuning the interval bounds which qualify excellent tetrahedra is our mean to trade efficiency for the final mesh quality.

### 2.5.3 Results

To evaluate our approach, we tested the various steps of our algorithm separately, then together. Figure 2.45 shows our refinement routine when no optimization step is performed. Notice that the resulting mesh lacks gradation, as typical for Delaunay refinement methods. We compare results of Delaunay refinement, Lloyd relaxation [DFG99], and our NODT in terms of number of slivers (*before* sliver removal for fairness) in Figure 2.46. Figure 2.47 shows the mesh of an elephant model obtained by Delaunay refinement (top) as it gets optimized by our NODT routine (middle), then after sliver removal (bottom).

Figure 2.48 shows the mesh of the bimba model obtained by interleaved refinement and optimization with approximation and element quality criteria activated (the sizing criterion $l_{max} = 0.1$ is not significant as the input PSC fits into a unit bounding box). The mesh contains 43K vertices and all dihedral angles are above 15 degrees. The input PSC has 400K vertices. We also tested our method on mechanical parts. Figure 2.50 shows the mesh of a turbine generated by our interleaved algorithm. The mesh contains significantly fewer vertices (13%) than Delaunay refinement alone.

We also compared our technique to DelPSC [CDL07] and TetGen [Si] in Figure 2.49. Our interleaved technique improves both over the mesh quality and complexity. For TetGen we provided as input both the input PSC (which is then refined) and the boundary of our optimized mesh for fair comparison. Figure 2.52 shows the mesh of the buddha model obtained by interleaved refinement and optimization. This example illustrates the mesh of a domain boundary with larger range of feature size.

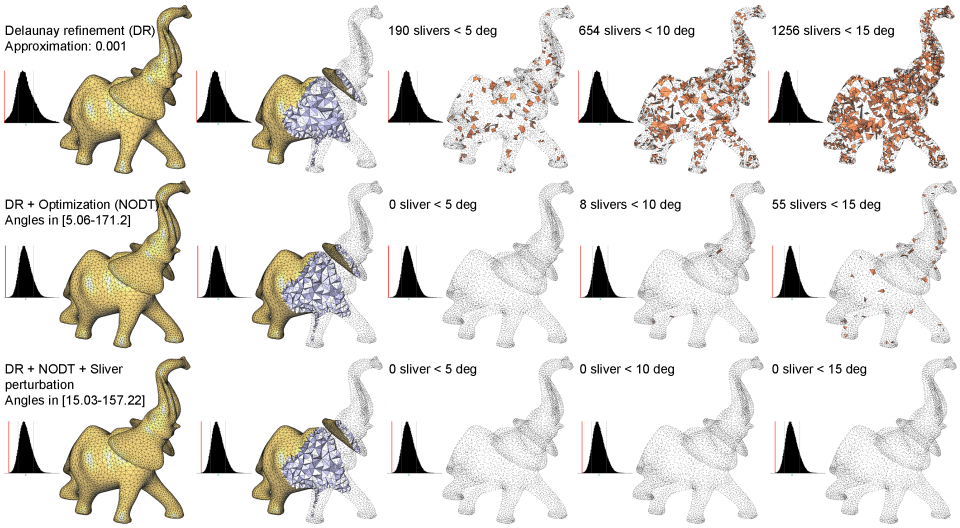



**Figure 2.47:** Elephant. Top: mesh generated by Delaunay refinement with $l_{max} = 0.1$, $\epsilon_{max} = 0.001$. Distribution of dihedral angles, and sliver tetrahedra with dihedral angles lower than 5, 10 and 15 degrees are shown. Middle: output of Delaunay refinement (i.e., top row) optimized with our technique. Bottom: optimized mesh (middle row) after sliver perturbation.

Activating the topology criterion enforces that each restricted facet has its three vertices on the same PSC patch, and that each restricted edge has its two vertices on the same PSC crease. The mesh can thus be refined beyond the specified approximation criterion until all surface sheets are separated, as illustrated by Figure 2.51. In our experience, computational times to obtain a mesh range from seconds for the sphere and nested spheres models to 3 hours for the Michelangelo David models through minutes for the anchor, turbine and bimba models (resp. 10, 15 and 23).

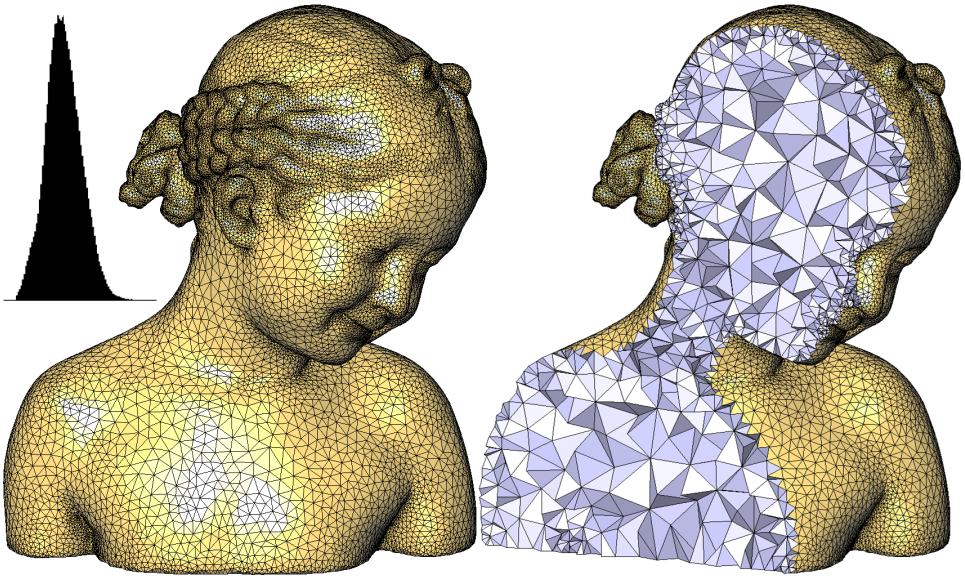


**Figure 2.48:** Bimba. Mesh generated by interleaved refinement and optimization with $l_{max} = 0.1$, $\epsilon_{max} = 0.0005$.

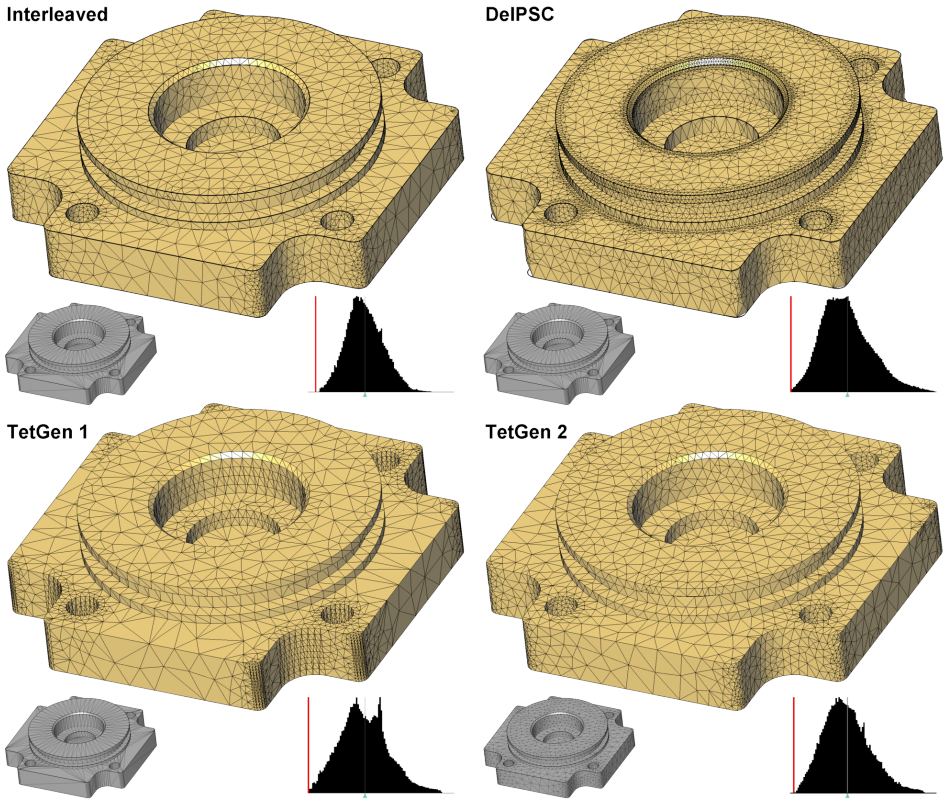



**Figure 2.49:** Cover rear. Top left: mesh obtained by interleaved refinement and optimization with $l_max = 0.1$, $\epsilon_{max} = 0.001$, $\sigma^f{}_{\max} = 1.5$, $\sigma^t{}_{\max} = 1.5$ and the topology criterion activated. It contains 4,050 vertices and all dihedral angles are above 12.0 degrees. Top right: mesh generated by DelPSC, with the same parameters. It contains 15,157 vertices and all dihedral angles are above 0.1 degree. Bottom left: mesh generated by TetGen, with the same parameters and input. It contains 6,966 vertices and all dihedral angles are above 0.2 degree. Bottom right: mesh generated by TetGen, with the same parameters and the boundary of our optimized mesh taken as input. It contains 4,189 vertices and all dihedral angles are above 3.0 degrees.

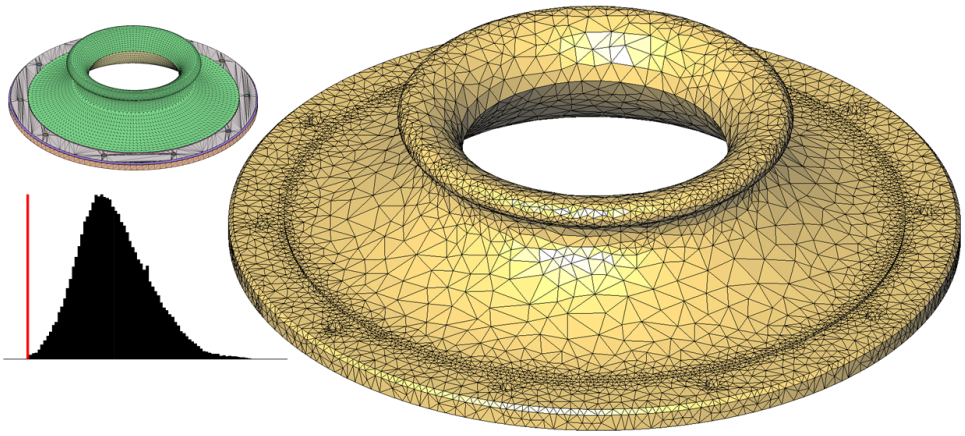


**Figure 2.50:** Turbine. Mesh generated by interleaved refinement and optimization with $l_{max} = 0.1$, $\epsilon_{max} = 0.001$. The inset shows the input PSC with all patches segmented. The mesh has 14K vertices and 51K tetrahedra, with all dihedral angles greater than 15 degrees.

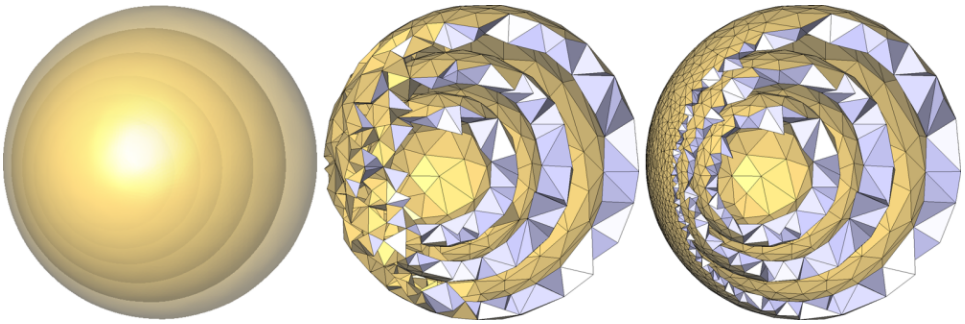


**Figure 2.51:** Nested spheres. Left: input PSC. Middle: mesh generated by refinement with $l_{max} = 1$, $\epsilon_{max} = 0.03$ and topology criterion not activated. Right: mesh further refined with same criteria but with topology activated.

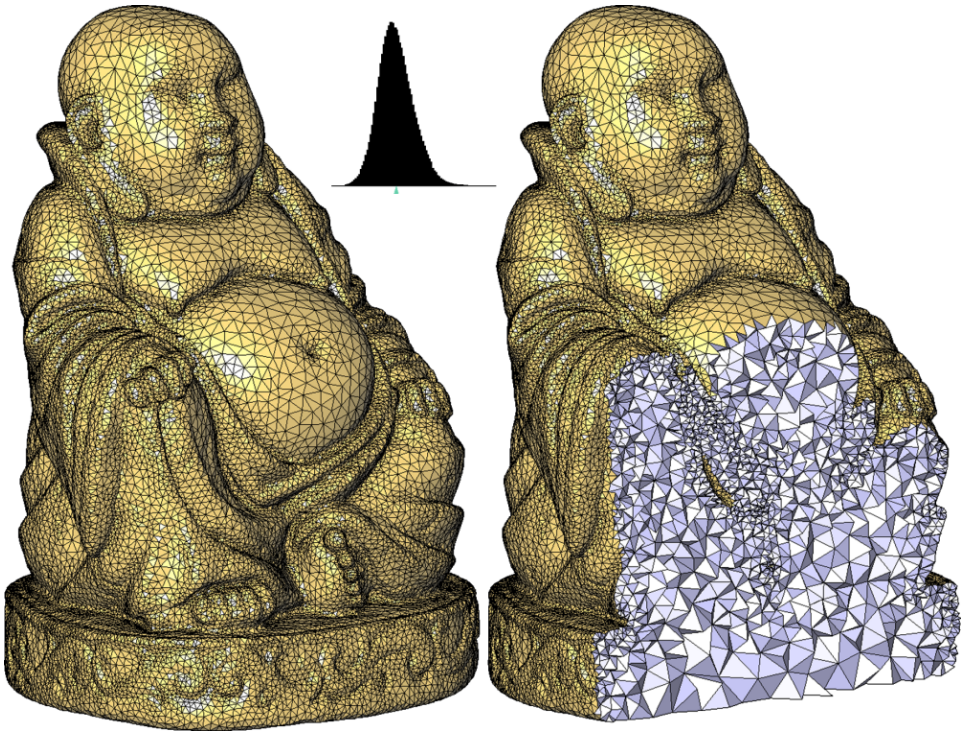


**Figure 2.52:** Buddha. Mesh obtained by interleaved refinement and optimization. All dihedral angles are above 15 degrees.

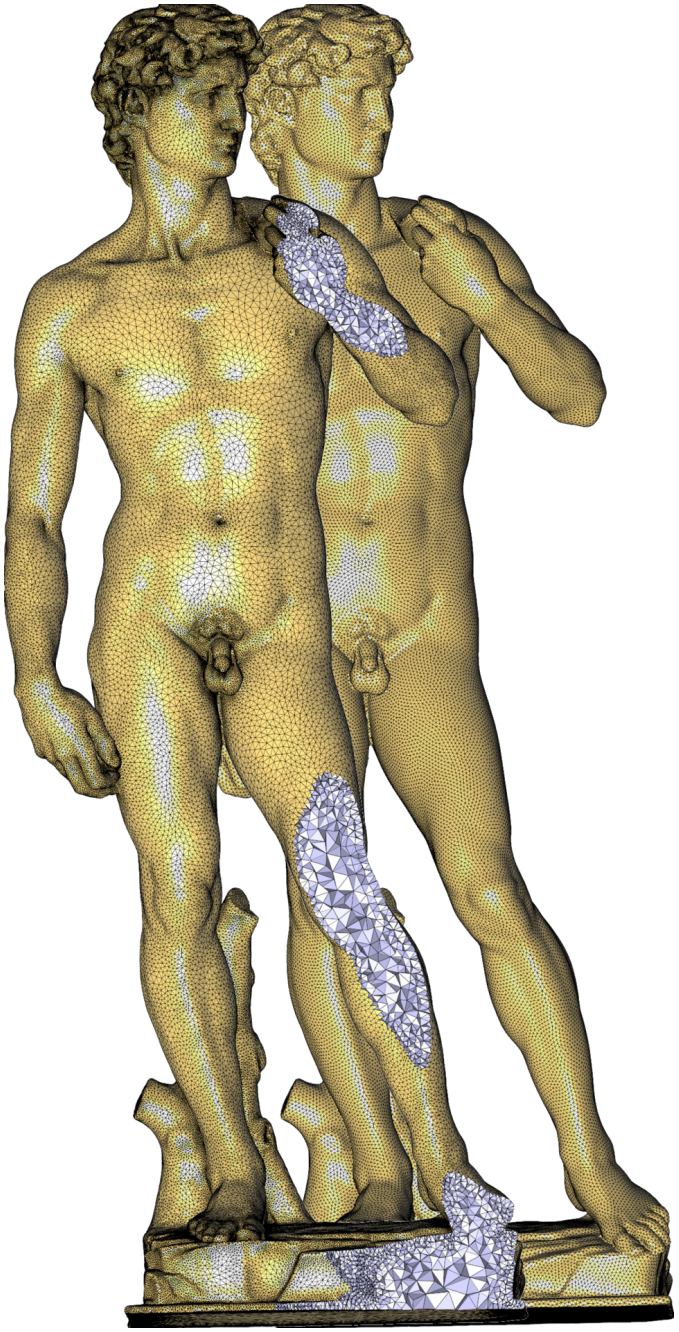


**Figure 2.53:** Michelangelo David. Michelangelo David. In this example, the input PSC has 800K triangles; on the right, illustrating scalability, a uniform sizing criterion generates a 1M vertices mesh; on the left, approximation error and shape criteria alone generate a smaller graded mesh (250K vertices), while guaranteeing the same quality of tetrahedra, and a better approximation error.

### 2.5.4 Summary

We introduced a mesh generation framework based on the idea of interleaving refinement and optimization. Guided by user-defined criteria such as size, shape, and approximation error of mesh elements, refinement steps are parsimoniously applied batch-wise through the insertion of independent sets. Optimization steps are performed through a variant of Chen's ODT that handles boundary as well as spatially-varying mesh sizing.

#### 2.5.4.1 Strengths

As the general framework of this algorithm is generic enough, we can accommodate other representations for the input, such as implicit surfaces or piecewise smooth parametric surfaces represented as NURBS patches. However, the latter would require efficient intersection computations using, e.g., the SINTEF Spline Library.

#### 2.5.4.2 Weaknesses

One important limitation of the current algorithm is that it does not handle sharp input creases subtending small angles (the theoretical bound on input angles is 90°, see [RY07]). Another limitation is that our implementation is slow (up to 1h for 1M vertices). Activating the interleaving later during refinement helps reducing the timings, but the mesh complexity can suffer from this simplification.

#### 2.5.4.3 Future Work

We believe that Delaunay-based mesh generation is a relevant methodological framework due to its versatility. Nevertheless, we have identified several problems which are not yet fully resolved: 3D domains bounded by sharp creases subtending small angles, slivers and anisotropic meshes. These problems correspond to the proposed directions of research.

We plan on dealing with small angles through the use of weights of a regular triangulation [CDL07]. Delaunay triangulation alone is not sufficient to mesh 3D domains bounded by piecewise smooth surfaces when some sharp creases subtend small angles. Cheng et al. [CDR07] have shown that switching to a regular triangulation provides a solution to these inputs. However this solution is not fully satisfactory, as the final mesh is overly dense along sharp creases. We intend to investigate a Delaunay refinement technique which would be more parsimonious while providing the same guarantees.

We intend to improve this work by switching to a function approximation problem instead of interpolation, where the vertex weights of a weighted Delaunay triangulation are optimized together with the vertex locations. At the intuitive level, we wish to embed a sliver exudation process [CDE+00] as part of the optimization in order to further reduce the number of slivers in the final mesh. In addition, we wish to investigate sliver perturbation further [Li00]. Our plan is to characterize a sliver with various criteria (radius of circumsphere, volume, length of Voronoi edges, intersection with incident circumspheres) and devise the minimal vertex relocation procedure which would make the sliver disappear. Although our initial experiments show that, e.g., simply relocating a sliver vertex along the gradient of its circumsphere radius is efficient enough to get rid of most isolated slivers by violating the empty circumsphere Delaunay property, it remains to understand how to get rid of structured chains of slivers. In particular, it may happen that one vertex incident to two or more slivers exhibit incompatible relocation vectors, or that the said vertex is constrained to lie on a sharp crease or corner of the input domain boundary.

At the theoretical level, we wish to certify bounds on the shape of the elements after optimization. One way to obtain such guarantees is to derive a hill-climbing version of the optimization step (a local optimization is validated only when the mesh quality is improved), as as to keep the same guarantees provided by the refinement step. Two reasons prevent us from deepening this way: the current bounds on dihedral angles of Delaunay refinement are not satisfactory and our current experiments show that the optimization step goes through intermediate states where the quality is decreased before being increased.

Finally, we wish to investigate the optimization of anisotropic tetrahedron meshes. For many simulations, a mesh must be anisotropic, with long, skinny tetrahedra with orientations and aspect ratios dictated by the application. Functions with strongly anisotropic Hessians are best interpolated with anisotropic meshes and partial differential equations that are inherently anisotropic are numerically best conditioned if anisotropic elements are used. In a recent work [BWY08] it has been shown how anisotropic meshes can be generated by refining a triangulation which is everywhere the image of a Delaunay triangulation under a stretching transformation. Refinement is carried on until all stars of the vertices tile the input domain. The main added value of this approach is a a guaranteed termination and a straightforward implementation as the idea involves simple predicates. It remains to understand how these ideas can be combined with mesh optimization.

# Chapter 3

# General Summary

This habilitation thesis has presented a selection of four contributions in the field of digital geometry processing achieved between 2004 to 2008: surface reconstruction from unoriented point sets, surface approximation, quadrangle surface tiling and generation of isotropic tetrahedron meshes. The narrative is mostly cumulative of the corresponding papers [ACSTD07, CSAD04, TACSD06, ACSYD05] (published at ACM SIGGRAPH and EUROGRAPHICS Symposium on Geometry Processing), and includes added figures and explanations, as well as detailed summaries with follow-ups and perspective for future work. These contributions have been chosen for two reasons: they represent important topics in the field, while following the variational methodology favored by the author.

The variational methodology, which casts each problem as an optimization, requires solving two parts: i) defining the most appropriate energy, and ii) elaborating upon a tractable way to solve for this energy. In general the second part of the problem (solving) is at least as difficult as defining the energy, and hoping to reach a global optimum is often wishful thinking. Nevertheless resorting to global numerical solvers such as eigenvalue solvers for surface reconstruction provides us not only with a high level of sparse sampling resiliency, but also with a way to reduce the constraints over the inputs (in this case, no need for oriented normals).

Optimization comes often at the cost of additional computations compared to greedy algorithms, this being particularly true for the optimization of tetrahedron meshes. This added cost explains our will to combine optimization either interleaved with or applied as a post-process after greedy mesh refinement. Ideally, we would like to guarantee that the mesh generation algorithm best trades mesh quality for computational time. Similar in spirit to [KS07], we would even like to guarantee that each additional cpu cycle spent is spent improving the mesh.

## 3.1 Perspectives

An increasing trend in applications is to deal with measurement data from the physical world. This calls for an increased robustness of the algorithm to be resilient not only to, e.g., noise and sparse sampling in point sets, but also to outliers. None of the algorithms presented in the narrative are resilient to such data. In addition, measurement data covers a large variety of input, which calls for algorithms which are as independent as possible to the representation of the input data. The oracle-based approach adopted for tetrahedron mesh generation is one step in this direction.

All optimization algorithms presented in this document involve both continuous variables (values of implicit or harmonic functions, parameters of planar proxies, vertex coordinates of tetrahedron meshes) and discrete data structures (connectivity of surface and 3D meshes) and variables (number of isolines for quadrangle surface tiling). In essence this reflects the dual nature of geometric data structures, well understood in computational geometry. Although both can be related using, e.g., finite element formulations or discrete exterior calculus as done in the discretization of the Laplacian operator, the discrete variables are in general either not solved with the continuous ones but solved in sequence or simply provided by the user (e.g., integer number of isolines of a tiling). Automating the algorithms calls for simultaneous or at least coupled solving of both discrete and continuous variables. This will become in particular crucial when tackling the problem of hexahedral domain tiling.

These challenges, as well as the many other remaining ones, herald an exciting future research program, requiring both new concepts as well as generic and robust algorithms to ultimately result in a “standard applied geometry toolbox” useful for engineering, medicine and multimedia applications. Applied, here, means that our primary concern is the computability on real-world computers, whereas toolbox refers to a coherent set of concepts and software tools which would become indispensable to the engineers and practitioners in a increasingly digital world.

# Chapter 4

# Bibliography


[AB99] Nina Amenta and Marshall W Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22, 1999.

[ABE99] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999.

[ACSD+03] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic Polygonal Remeshing. *ACM Trans. on Graphics*, 22(3):485–493, 2003.

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 39–48, 2007.

[ACSYD05] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. In *Proc. of ACM SIGGRAPH 2005*, volume 24(3), pages 617–625, 2005.

[ADA07] L. Antani, C. Delage, and P. Alliez. Mesh sizing with additively weighted voronoi diagrams. In *Proc. of the 16th Int. Meshing Roundtable*, pages 335–346, 2007.

[AG05] Pierre Alliez and Craig Gotsman. *Recent Advances in Compression of 3D Meshes (in Advances in Multiresolution for Geometric Modelling)*. Springer-Verlag, 2005.

[AMCO08] Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust surface registration. In *ACM SIGGRAPH*, 2008.

[AS98] P. Agarwal and S. Suri. Surface Approximation and Geometric Partitions. *SIAM J. Comput.*, 19:1016–1035, 1998.

[ASCE02] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. MESH: Measuring Errors between Surfaces using the Hausdorff Distance. In *IEEE Multimedia*, pages 705–708, 2002.

[AUGA07] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. Recent advances in remeshing of surfaces. In Leila de Floriani and Michela Spagnuolo, editors, *Shape Analysis and Structuring*. Springer-Verlag, November 2007.

[AV07] D. Arthur and S. Vassilvitskii. k-means++ : The advantage of careful seeding. In *SODA*, pages 1027–1035, 2007.

[BBK05] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for mesh processing. In *IMA Conf. on Math. of Surfaces*, pages 62–83, 2005.

[BBVK04] Mario Botsch, David Bommes, Christoph Vogel, and Leif Kobbelt. Gpu-based tolerance volumes for mesh processing. In *Pacific Graphics*, 2004.

[BC01] J-D. Boissonnat and F. Cazals. Coarse-to-fine surface simplification with geometric guarantees. *Computer Graphics Forum (EUROGRAPHICS)*, 2001.

[BC02] Boissonnat and Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *CGTA: Computational Geometry: Theory and Applications*, 22, 2002.

[BF05] Houman Borouchaki and Pascal Frey. Simplification of surface mesh using hausdorff envelope. *Comput. Methods Appl. Mech. Engrg*, 2005.

[BH96] Frank Bossen and Paul Heckbert. A Pliant Method for Anisotropic Mesh Generation. In *5th Intl. Meshing Roundtable*, pages 63–76, oct 1996.

[BKBH07] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. Multilevel streaming for out-of-core surface reconstruction. In *Symposium on Geometry Processing*, pages 69–78, 2007.

[BMR+99] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cludio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5:349–359, 1999.

[BMRJ04] Ioana Boier-Martin, Holly Rushmeier, and Jingyi Jin. Parameterization of triangle meshes over quadrilateral domains. In *Symp. on Geometry processing*, pages 193–203, 2004.

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

[BOG02] C. Boivin and C. Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *Int. J. Numer. Methods Eng*, 55:1185–1213, 2002.

[Boi84] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. on Graphics*, 3(4):266–286, 1984.

[Bot05] Mario Botsch. *High Quality Surface Generation and Efficient Multiresolution Editing Based on Triangle Meshes*. PhD thesis, RWTH Aachen, 2005. Ph.D. Thesis.

[BPCZ07] Mario Botsch, Renato Pajarola, Baoquan Chen, and Matthias Zwicker. *Proceedings of Eurographics Symposium on Point-Based Graphics*. Eurographics Association, 2007.

[BPK05] Stephan Bischoff, Darko Pavic, and Leif Kobbelt. Automatic restoration of polygon models. *ACM Transactions on Graphics (TOG)*, 24(4):1332–1352, 2005.

[BPK+07] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Rössl. Geometric modeling based on polygonal meshes. In *Geometric Modeling*. ACM SIGGRAPH Course Notes, 2007.

[BR04] Benedict Brown and Szymon Rusinkiewicz. Non-rigid range-scan alignment using thin-plate splines. In *Symposium on 3D Data Processing, Visualization, and Transmission*, 2004.

[BSPG06] Mario Botsch, Robert Sumner, Mark Pauly, and Markus Gross. Deformation transfer for detail-preserving surface editing. In *Vision, Modeling and Visualization*, 2006.

[BVL02] Laurent Balmelli, Martin Vetterli, and Thomas M. Liebling. Mesh Optimization Using Global Error with Application to Geometry Simplification. *Graphical Models*, 64(3-4):230–257, May 2002.

[BWY07] J.D. Boissonnat, C. Wormser, and M. Yvinec. Curved voronoi diagrams. In *Effective Comp. Geometry for Curves and Surfaces*, pages 67–116. Springer, 2007.

[BWY08] J.-D. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proceedings of the Symposium on Computational Geometry*, 2008.

[Can06] Emmanuel Candès. Compressive sampling. In *Int. Congress of Mathematics*, pages 1433–1452, 2006.

[CBC+01] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH*, pages 67–76, 2001.

[CDDD01] Albert Cohen, Wolfgang Dahmen, Ingrid Daubechies, and Ronald DeVore. Tree Approximation and Optimal Encoding. *Appl. Comput. Harmon. Anal.*, 11(2):192–226, 2001.

[CDE+00] S.W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, and S.H. Teng. Sliver exudation. *Journal of the ACM*, 47(5):883–904, 2000.

[CDL07] S.W. Cheng, T.K. Dey, and J. Levine. A practical delaunay meshing algorithm for a large class of domains. In *Proc. of the 16th Int. Meshing Roundtable*, pages 477–494, 2007.

[CDR07] S.W. Cheng, T.K. Dey, and E.A. Ramos. Delaunay refinement for piecewise smooth complexes. In *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–1105, 2007.

[CDSM04] F. Cayre, O. Devillers, F. Schmitt, and H. Maitre. Watermarking 3d triangle meshes for authentication and integrity. Technical Report 5223, INRIA, 2004.

[CG06] Frédéric Cazals and Joachim Giesen. Delaunay triangulation based surface reconstruction. In J.D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 231–276. Springer-Verlag, Math. and Visualization, 2006.

[CGA09] Computational Geometry Algorithms Library CGAL-3.4. http://www.cgal.org/, 2009.

[Che89] L.P. Chew. Guaranteed-quality triangular meshes. Technical Report 89-983, Department of Computer Science, Cornell University, 1989.

[Che93] L.P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *SCG '93: Proceedings of the ninth annual symposium*

*on Computational geometry*, pages 274–280. ACM New York, NY, USA, 1993.

[Che04] L. Chen. Mesh smoothing schemes based on optimal delaunay triangulations. In *13th International Meshing Roundtable*, pages 109–120, 2004.

[CP03] Frederic Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Symposium on Geometry Processing*, pages 177–187, 2003.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, pages 905–914, 2004.

[CSM03] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of the Symp. on Computational Geometry*, pages 312–321, 2003.

[CX04] L. Chen and J. Xu. Optimal delaunay triangulations. *Journal of Computational Mathematics*, 22(2):299–308, 2004.

[D'A00] E. F. D'Azevedo. Are Bilinear Quadrilaterals Better Than Linear Triangles? *SIAM Journal on Scientific Computing*, 22(1):198–217, 2000.

[dB05] G. Van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Graphics Tools: The Jgt Editors' Choice*, 2005.

[DBG+06] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. to appear at ACM SIGGRAPH '06, July 2006.

[Dey06] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics, 2006.

[DFG99] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, 1999.

[DGGZ03] Tamal K. Dey, Joachim Giesen, Samrat Goswami, and Wulue Zhao. Shape dimension and approximation from samples. *Discrete & Computational Geometry*, 29(3), 2003.

[DHOS07] Joel II Daniels, Linh K. Ha, Tilo Ochotta, and Claudio T. Silva. Robust smooth feature extraction from point clouds. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 123–136, 2007.

[DKG05] Shen Dong, S. Kircher, and Michael Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Design (Special Issue on Geometry Processing)*, 22(4):392–423, 2005.

[DLS05] Tamal K. Dey, Gang Li, and Jian Sun. Normal estimation for point clouds: A comparison study for a Voronoi based method. In *Symposium on Point-Based Graphics*, pages 39–46, 2005.

[DLSCS08] T. Dey, K. Li, J. Sun, and D. Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3d models. In *ACM SIGGRAPH*, 2008.

[DS91] Ed D'Azevedo and Bruce Simpson. On Optimal Triangular Meshes for Minimizing the Gradient Error. *Numer. Math.*, 59:321–348, 1991.

[DS05] Tamal K. Dey and Jian Sun. Normal and Feature Estimations from Noisy Point Clouds. Technical Report OSU-CISRC-7/50-TR50, Ohio State University, 2005.

[Ede00] H. Edelsbrunner. Mathematical problems in the reconstruction of shapes, 2000. Talk at MSRI's Workshop on Computational Algebraic Analysis (`http://msri.mathnet.or.kr/`).

[FCODS08] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer. Upright orientation of man-made objects. *ACM Transaction on Graphics (Proc. SIGGRAPH)*, 27(3), 2008.

[FDCO03] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graph.*, 22(3):950–953, 2003.

[FKMS05] Thomas Funkhouser, Michael Kazhdan, Patrick Min, and Philip Shilane. Shape-based retrieval and analysis of 3d models. *Communications of the ACM*, 48(6):58–64, 2005.

[FOG97] L.A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *In. Jour. for Num. Methods in Eng.*, 40(21):3979–4002, 1997.

[Fu93] Joseph H. G. Fu. Convergence of curvatures in secant approximations. *Journal of Differential Geometry*, 37:177–190, 1993.

[GAP08] Ankit Gupta, Pierre Alliez, and Sylvain Pion. Principal Component Analysis in CGAL. Technical Report 6642, INRIA, 2008.

[GBK03] S. Gumhold, Pavel Borodin, and Reinhard Klein. Intersection free simplification. *International Journal of Shape Modeling*, 9(2):155–176, 2003.

[GH98] Michael Garland and Paul Heckbert. Simplifying Surfaces with Color and Texture using Quadric Error Metrics. In *IEEE Visualization Proceedings*, pages 263–269, 1998.

[Gra98] Alfred Gray, editor. *Modern Differential Geometry of Curves and Surfaces*. Second edition. CRC Press, 1998.

[GS] Francisco M. Gomes and Danny C. Sorensen. ARPACK++: A C++ implementation of ARPACK eigenvalue package.

[Gus02] Igor Guskov. An anisotropic parameterization scheme. In *Proc. of Int. Meshing Roundtable*, pages 325–332, 2002.

[GWH01] Michael Garland, Andrew Willmott, and Paul Heckbert. Hierarchical Face Clustering on Polygonal Surfaces. In *ACM Symp. on Interactive 3D Graphics*, pages 49–58, 2001.

[GY03] Xianfeng Gu and Shing-Tung Yau. Global conformal parameterization. In *Symposium on Geometry Processing*, pages 127–137, 2003.

[Hau01] Alejo Hausner. Simulating Decorative Mosaics. In *Proceedings of ACM SIGGRAPH*, pages 573–578, August 2001.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proc. of ACM SIGGRAPH*, pages 71–78, 1992.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh Optimization. In *ACM SIGGRAPH Proceedings*, pages 19–26, 1993.

[HG99] Paul Heckbert and Michael Garland. Optimal Triangulation and Quadric-Based Surface Simplification. *Journal of Computational Geometry: Theory and Applications*, 14(1-3):49–65, nov 1999.

[HK06a] A. Hornung and L. Kobbelt. Hierarchical volumetric multi-view reconstruction of manifold surfaces based on dual graph embedding. In *IEEE Conference on Computer and Pattern Recognition*, volume 1, pages 503–510, 2006.

[HK06b] Alexander Hornung and Leif Kobbelt. Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In *Symposium on Geometry Processing*, pages 41–50, 2006.

[Hop96] Hugues Hoppe. Progressive Meshes. In *ACM SIGGRAPH Proceedings*, pages 99–108, 1996.

[HSH09] L. Hu, P. Sander, and H. Hoppe. Gpu-based tolerance volumes for mesh processing. In *ACM Symposium on Interactive 3D Graphics and Games*, 2009.

[HUHJ01] B. Heckel, A. E. Uva, B. Hamann, and K. I. Joy. Surface Reconstruction Using Adaptive Clustering Methods. In *Geometric Modelling: Dagstuhl 1999*, volume 14, pages 199–218, 2001.

[HXMP05] Guofei Hu, Jie Xu, Lanfang Miao, and Qunsheng Peng. Bilateral estimation of vertex normal for point-sampled models. In *Int. Conf. on Comp. Science and Appl.*, volume 3480, pages 758–768, 2005.

[HZM+08] Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt, and Hujun Bao. Spectral quadrangulation with orientation and alignment control. In *Proceedings of SIGGRAPH Asia*, 2008.

[IIY+99] K. Inoue, T. Itoh, A. Yamada, T. Furuhata, and K. Shimada. Clustering Large Number Of Faces For 2-Dimensional Mesh Generation. In *8th Int. Meshing Roundtable*, pages 281–292, 1999.

[JKS05] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum (Proc. Eurographics)*, 24(3):581–590, 2005.

[JWYG04] Miao Jin, Yalin Wang, Shing-Tung Yau, and Xianfeng Gu. Optimal global conformal surface parameterization. In *IEEE Visualization*, pages 267–274, 2004.

[KBH06] Michael Kazhdan, M. Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Symposium on Geometry Processing*, pages 61–70, 2006.

[KLS96] Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh Reduction with Error Control. In *IEEE Visualization Proceedings*, pages 311–318, 1996.

[KMN+02] Tapas Kanungo, David M. Mount, Nathan Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm. *IEEE Trans. on PAMI*, 24(4):881–892, July 2002.

[KNP07] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384, 2007.

[KS07] B. Klingner and J.R. Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, 2007.

[KSCOS08] V. Kraevoy, A. Sheffer, D. Cohen-Or, and A. Shamir. Non-homogeneous resizing of complex models. In *ACM Transactions on Graphics (SIGGRAPH ASIA Conference Proceedings)*, volume 27(5), 2008.

[KSO04] Ravikrishna Kolluri, Jonathan R. Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing*, pages 11–21, 2004.

[KSS00] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive Geometry Compression. In *Proc. of ACM SIGGRAPH*, pages 271–278, July 2000.

[KSS06] Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. *ACM Trans. on Graphics*, 25(2), 2006.

[KT03] Sagi Katz and Ayellet Tal. Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts. *ACM Trans. on Graphics*, 22(3):954–961, 2003.

[KVLpS99] Leif Kobbelt, Jens Vorsatz, Ulf Labsik, and Hans peter Seidel. A shrink wrapping approach to remeshing polygonal surfaces. In *EUROGRAPHICS 99*, 1999.

[Li00] Xiangyang Li. *Sliver-free Three Dimensional Delaunay Mesh Generation*. PhD thesis, Computer Science, University of Illinois at Urbana-Champaign, 2000.

[Lie03] P. Liepa. Filling holes in meshes. In *Symposium on Geometry Processing*, pages 200–205, 2003.

[Llo82] S. Lloyd. Least square quantization in PCM. *IEEE Trans. Inform. Theory*, 28:129–137, 1982.

[LP05] Carsten Lange and Konrad Polthier. Anisotropic smoothing of point sets. *Computer Aided Geometric Design*, 22(7):680–692, 2005.

[LPK09] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data, 2009. submitted to the Computer Graphics Forum.

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least Squares Conformal Maps for Automatic Texture Atlas Generation. In *ACM SIGGRAPH Proceedings*, pages 362–371, 2002.

[LRC+02] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Elsevier, 2002.

[LS07] F. Labelle and J.R. Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 2007.

[LSP08] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (SGP)*, 2008.

[LT98] P. Lindstrom and G. Turk. Fast and Memory Efficient Polygonal Simplification. In *IEEE Visualization Proceedings*, pages 279–286, 1998.

[LT00] Peter Lindstrom and Greg Turk. Image-driven simplification. *ACM Transactions on Graphics*, 19(3):204–241, July 2000.

[ME08] Mark Pauly Michael Eigensatz, Robert W. Sumner. Curvature-domain shape processing. *Computer Graphics Forum - Eurographics*, 2008.

[MGP06] Niloy Mitra, Leonidas Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM SIGGRAPH*, 2006.

[MGP07] N. J. Mitra, L. Guibas, and M. Pauly. Symmetrization. In *ACM Transactions on Graphics*, volume 26(3), pages #63, 1–8, 2007.

[MK04] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proceedings of the Pacific Graphics*, pages 207–216, 2004.

[MKB+08] Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. Polyhedral finite elements using harmonic basis functions. In *Computer Graphics Forum 27(5), Proc. Geometry Processing 2008*, pages 1521–1529, 2008.

[MNG04a] N. J. Mitra, A. Nguyen, and L. Guibas. Estimating surface normals in noisy point cloud data. In *Int. J. of Comp. Geometry and Applications*, volume 14(4–5), pages 261–276, 2004.

[MNG04b] Niloy Mitra, An Nguyen, and Leonidas Guibas. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications*, 2004.

[MOG09] Quentin Merigot, Maks Ovjsanikov, and Leonidas Guibas. Voronoi-based robust estimation of feature and principal curvature directions. In *25th European Workshop on Computational Geometry*, 2009.

[MPS+04] Michela Mortara, Giuseppe Patane, Michela Spagnuolo, Bianca Falcidieno, and Jarek Rossignac. Blowing Bubbles for the Multiscale Analysis and Decomposition of Triangle-Meshes. *Algorithmica*, 38(1), January 2004.

[MYV93] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive Texture Mapping. In *ACM SIGGRAPH Proceedings*, pages 27–34, 1993.

[Nad86] Edmond Nadler. Piecewise-Linear Best $\mathcal{L}^2$ Approximation On Triangulations. In C. K. Chui, L. L. Schumaker, and J. D. Ward, editors, *Approximation Theory V*, pages 499–502. Academic Press, 1986.

[NCC02] D. Nave, N. Chrisochoides, and L.P. Chew. Guaranteed quality parallel delaunay refinement for restricted polyhedral domains. *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 135–144, 2002.

[OBA+03a] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *Proc. of ACM SIGGRAPH*, volume 22(3), pages 463–470, 2003.

[OBA+03b] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level Partition of Unity Implicits. *ACM Trans. on Graphics*, 22(3):463–470, July 2003.

[OBP03] Yutaka Ohtake, Alexander Belyaev, and Alexander Pasko. Dynamic Mesh Optimization for Polygonized Implicit Surfaces with Sharp Features. *The Visual Computer*, 19(2):115–126, 2003.

[OF05] Daoshan OuYang and Hsi-Yung Feng. On the normal vector estimation for point cloud data from smooth surfaces. *Computer-Aided Geometric Design*, 37(10):1071–1079, 2005.

[ORY05] S. Oudot, L. Rineau, and M. Yvinec. Meshing volumes bounded by smooth surfaces. In *Proc. of the 14th Int. Meshing Roundtable*, pages 203–219, 2005.

[Pau03] Mark Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry.* PhD thesis, ETH Zurich, 2003. Ph.D. Thesis.

[Peb02] Philippe Pebay. Planar Quadrangle Quality Measures: Is There Really A Choice? In *11th Intl. Meshing Roundtable*, pages 53–62, sep 2002.

[Pet07] Sylvain Petitjean. Contributions au calcul géométrique effectif avec des objets courbes de faible degré, 2007. Habilitation thesis (Institut National Polytechnique de Lorraine).

[PGK02] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient Simplification of Point-Sampled Surfaces. In *Proc. of IEEE Visualization*, pages 163–170, 2002.

[PGR07] Joshua Podolak, Aleksey Golovinskiy, and Szymon Rusinkiewicz. Symmetry-enhanced remeshing of surfaces. In *Symposium on Geometry Processing*, 2007.

[PK08] Darko Pavic and Leif Kobbelt. High-resolution volumetric computation of offset surfaces with feature preservation. *Computer Graphics Forum (EUROGRAPHICS)*, 2008.

[PKKG03] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. In *(SIGGRAPH)*, volume 22(3) of *ACM Trans. on Graphics*, pages 641–650, 2003.

[PKL05] Sung-Bum Park, Chang-Su Kim, and Sang-Uk Lee. Error resilient 3-D mesh compression. *IEEE transactions on multimedia*, 8(55):885–895, 2005.

[PMG+05] Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus Gross, and Leonidas Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, 2005.

[PMW+08] Mark Pauly, Niloy Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas Guibas. Discovering structural regularity in 3d geometry. In *ACM SIGGRAPH*, 2008.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[PPB+05] O. Polonsky, G. Patane, S. Biasotti, C. Gotsman, and M. Spagnuolo. What's in an image: Towards the computation of the "best" view of an object. *The Visual Computer (Proc. Pacific Graphics)*, 21(8-10):840–847, 2005.

[PSQ06] Sylvain Paris, Fran cois X. Sillion, and Long Quan. A surface reconstruction method using global graph cut optimization. *Int. J. Comput. Vision*, 66(2):141–161, 2006.

[PWY+06] Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai, and Shi-Min Hu. Principal curvatures from the integral invariant viewpoint. *Comput. Aided Geom. Design*, 2006.

[RL01] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3D Digital Imaging and Modeling*, 2001.

[RLL+06] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *Transaction on Graphics*, 25(4):1460 – 1485, 2006.

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

[RVAL09] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Lévy. Geometry aware direction field processing. *ACM Transactions on Graphics*, 2009. To appear.

[RY06] Laurent Rineau and Mariette Yvinec. A generic software design for Delaunay refinement meshing. Technical Report 5983, INRIA, 2006.

[RY07] L. Rineau and M. Yvinec. Meshing 3d domains bounded by piecewise smooth surfaces. In *Proc. of the 16th Int. Meshing Roundtable*, pages 443–460, 2007.

[SACO04] Andrei Sharf, Marc Alexa, and Daniel Cohen-Or. Context-based surface completion. In *SIGGRAPH Proceedings*, pages 878–887, 2004.

[SAG03] Vitaly Surazhsky, Pierre Alliez, and Craig Gotsman. Isotropic Remeshing of Surfaces: a Local Parameterization Approach. In *Proceedings of 12th International Meshing Roundtable*, pages 215–224, 2003.

[SBS05] Olivier Schall, Alexander Belyaev, and Hans-Peter Seidel. Robust filtering of noisy scattered point data. In *Point-Based Graphics*, pages 71–144, 2005.

[SBS07] Oliver Schall, Alexander Belyaev, and Hans-Peter Seidel. Feature-preserving non-local denoising of static and time-varying range data. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 217–222, 2007.

[SCOT03] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass Quantization for Mesh Encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 42–51, 2003.

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[She98] J.R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proc. of the Fourteenth Annual Symp. on Comp. Geometry*, pages 86–95, 1998.

[She01] Alla Sheffer. Model Simplification for Meshing Using Face Clustering. *Computer Aided Design*, 33:925–934, 2001.

[She02a] Jonathan R. Shewchuk. What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures, 2002. Preprint.

[She02b] J.R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[Si] Hang Si. TetGen, a quality tetrahedral mesh generator and three-dimensional delaunay triangulator. http://tetgen.berlios.de/.

[Sim94] R. Bruce Simpson. Anisotropic Mesh Transformations and Optimal Error Control. *Appl. Num. Math.*, 14(1-3):183–198, 1994.

[SK04] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *Proc. of Symp. on 3D Data Processing, Visualization and Transmission (3DPVT) '04*, pages 68–75, 2004.

[SKS01] Robert Schneider, Leif Kobbelt, and Hans-Peter Seidel. Improved bi-laplacian mesh fairing. *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 445–454, 2001.

[SLS+07] Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. Interactive topology–aware surface reconstruction. In *SIGGRAPH*, 2007.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Transactions on Pattern Analysis and Machine Intelligence*, pages 888–905, 2000.

[SNCH08] P. V. Sander, D. Nehab, E. Chlamtac, and H. Hoppe. Efficient traversal of mesh edges using adjacency primitives. In *SIGGRAPH Asia*, 2008.

[SP04] Robert Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *SIGGRAPH Proceedings*, 2004.

[SSGH01] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture Mapping Progressive Meshes. In *ACM SIGGRAPH Proceedings*, pages 409–416, 2001.

[SSP08] Boris Sprinborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. In *Proceedings of ACM SIGGRAPH*, 2008.

[SW90] W. H. F. Smith and P. Wessel. Gridding with continuous curvature splines in tension. *Geophysics*, 55(3):293–305, 1990.

[SWG+03] Pedro Sander, Zoë J. Wood, Steven Gortler, John Snyder, and Hugues H. Hoppe. Multi-chart Geometry Images. In *Proceedings of the ACM/EG Symposium on Geometry Processing*, pages 146–155, June 2003.

[TACSD06] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In Alla Sheffer and Konrad Polthier, editors, *Eurographics Symposium on Geometry Processing*, pages 201–210, 2006.

[TCR05] Sivan Toledo, D. Chen, and V. Rotkin. TAUCS. Available at `http://www.tau.ac.il/~stoledo/taucs`, 2005.

[Ter05] Pierre Terdiman. OPCODE 3D collision detection library, 2005. http://www.codercorner.com/Opcode.htm.

[VLV+04] A.W. Vieira, T. Lewiner, L. Velho, H. Lopes, and G. Tavares. Stellar mesh simplification using probabilistic optimization. *Computer Graphics Forum*, 23(4):825–838, 2004.

[WCS05] Christian Walder, Olivier Chapelle, and Bernhard Schölkopf. Implicit surface modelling as an eigenvalue problem. In *Machine Learning ICML 2005*, pages 936–939, 2005.

[WHDS03] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing Excess Topology in Isosurfaces, 2003. Preprint.

[WK02] J. Wu and L. Kobbelt. Fast mesh decimation by multiple-choice techniques. *Vision, Modeling, and Visualization 2002*, pages 241–249, 2002.

[WK05] J. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum (Eurographics proceedings)*, 24(3):277–284, 2005.

[WZS+06] Rui Wang, Kun Zhou, John Snyder, Xinguo Liu, Hujun Bao, Qunsheng Peng, and Baining Guo. Variational sphere set approximation for solid objects. *The Visual Computer*, 22(9):612–621, 2006.

[YLPM05] Sung-Eui Yoon, Peter Lindstrom, Valerio Pascucci, and Dinesh Manocha. Cache-oblivious mesh layouts. In *SIGGRAPH Proceedings*, 2005.

[YLW06] Dong-Ming Yan, Yang Liu, and Wenping Wang. Quadric surface extraction by variational shape approximation. In *Proceedings of Geometric Modeling and Processing 2006*, pages 73–86, 2006.

[YZ04] Lexing Ying and Denis Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Trans. on Graphics*, 23(3):271–275, 2004.

[ZRS05] Rhaleb Zayer, Christian R ossl, and Hans-Peter Seidel. Setting the boundary free: A composite approach to surface parameterization. In *Symposium on Geometry Processing*, pages 91–100, 2005.