Synergistic Geometry Processing: from Robust Geometric Modeling to Scalable Physical Simulations

by

Zhongshi Jiang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September, 2022

Professor Daniele Panozzo

Acknowledgements

I am priviledges to be able to finish the work, in the turbulent time. Working remotely from my home.

Something about COVID.

- nTopology Inc, Suraj, support.
- Adobe: Vova
- Mentor: Daniele, Denis,
- Collaborators: Scott, Teseo,
- Labmates: Francis, Francisca, Chase, Julian, Qingnan, Davi, Yixin, YunFan, Jiacheng, Jeremie
- Officemates: Junbo, Xiang Z, Xifeng G,
- Staff: Shenglong, Hong
- Friends and family:

Abstract

Contents

List of Figures

List of Tables

1 Introduction

Most of the objects that we invent and interact in the physical world, ranging from electronic devices to transportation vehicles, are manufactured through a digital aided production pipeline. In practice, a typical product begins its lifecycle from a digital specification of the shape and material. And we can take advantage of the numerical solution of partial differential equations (PDEs) to portray how a design would behave even before it is created. Computer simulation would then test the behavior to match the designer's intent: a car hood's deformation under pressure of collision or a steel pan's temperature distribution when heated on a stove. In this case, a *robust*, *efficient* and *accurate* design-simulation pipeline could lift many iterations of manufacturing to the digital platform. In addition, as it is safer to conduct more stress experiments in a virtual environment, a reliable simulation also promotes safety, especially in designing biomechanical equipments.

On the other hand, we also create digital assets from real objects through an inverse pipeline. Traditionally, it serves as a way to reverse engineer a part into a digital format, or storing ancient or otherwise valuable artifacts. And more recently, with the advance of 3D scanning and capture devices, people are increasingly interested in representing themselves and surroundings as digital avatars and virtual environment. Such techniques typically go from raw scanning depth images to obtain a digital 3D shape. The users and algorithms may then perform simplification, editing, fairing, modifications, or other modeling operations. The digital objects will then interact with each other in the virtual environment, preferably abiding essential physics. Such advances produces growing quantites of 3D model datasets with great versatility of organic shapes, as well as their defects, including holes, self-overlapping surfaces, and so on. Such defects requires expertise and manual labor to fix, throttle the effort for everyone to enjoy and create in the virtual world. To build an information rich virtual environment, it is also essential for us to extend and scale up the algorithmic capabilities to process and understand the shape, material, and physics of these shapes.

There are various approaches to specify a shape, depending on the different geometry characteristics. Shape modeling evolves beyond smooth and regular objects like sculpture or furniture. Algorithm deduced structures have gained popularity since they deliver the promise of maximal durability with limited material. Downstream, computer simulation requires a discrete tessellation of the object. A notable example is a mesh: it contains a list of vertices for the spatial locations and a list of simple polygons or polyhedra. However, it is rarely the case where one mesh fits all. More degrees of freedom give a more accurate description of the desired shape, but it inevitably means more computational resources are needed to handle the computation.

Researchers usually examine the 3D acquisition, computational design, deformable animation,

and physical simulation separately, leading to different computational tools for the discretized geometry. Different representations between design methods and various adaptations of simulations, even for the same shape, are not reliably correlated. In the general case, the material, attribute, or experiment setting cannot be robustly re-used on another representation without laborious manually fixing. Such a pipeline demands the engineer or designer to understand precisely the properties of the specific numerical methods of choice. It also prohibits an automatic and robust pipeline from optimizing the desired physical durability or permeability structure.

In contrary to the 2D representations of image, which commonly lives on grid data, there has been different candidates to represent and process information on a 3D shape have various representations. Ranging from multi-view depth images, point clouds, strucuted and unstructured meshes, and there are different trade-off assocaited, ranging from memory efficiency, and specialized algorithms. And across different stages of the computational pipeline regarding a 3D shape, different representations might be used, to adapt to specialized algorithms: for example, 3D acquisition devices produces multiview depth images; some design and modeling tools make use the topology-flexible implicit surfaces as the representation for lattice or porous structures; digital asset modeling toolchains and industry make use of surface triangle or quadralaterial meshes extensively; physical simulation requires a spatial partitioning, as commonly tetrahedral meshes to carry out the quantities inside a volume.

The thesis aims to develop computational principles to reliably associate different representations of discrete geometry entities at diverse design and simulation stages. Instead of re-assigning properties at various phases of the design-simulation pipeline, or whenever the required budget is different, a good association between geometry at different stages gives more freedom to the design and development of automatic and adaptive solvers. The adaptation can be bi-directional: on the one hand, with coarse and concise tessellation, the algorithm utilizes the information associated with the original design to obtain more faithful simulations. On the other hand, in early-stage prototyping, and automatic data generation for deep learning, the principles I established allows for simplifying the shape and retaining the settings, providing a fast answer. In addition, I use the same principle to design the first large-scale and robust approach for curved mesh representation, which provides accurate representation as well as efficient physical simulations. In addition, by reliably connecting the data across different representations and at different stages, guarantees of correctness of properties can be safely uphold, which would reduce the chance of mistakes and iterations by manual labor. Furthermore, since different algorithms perform well on different representations, multiple algorithms can finally work together, and synergistically improve the performance of an overall pipeline. Such a contribution outreach to 3d model design and acquisition pipeline, as well as their physical behavior predication, and manufactured.

The thesis consists of several different, but highly related techniques, and the central technique is to leverage auxiliary geometric constructions to design novel constraints, usable in the framework of mathematical optimization. By integrating such geometric constraints into the surface parameterization, or mesh generation and optimization pipeline, I am able to arrive at provable guarantees. In the meantime, with the limited precision implementations, geometric algorithms are often troubled with numerical problems. Thus, I also perform extensive experiments to back up the claim and make the algorithms theory more easily accessible to industrial applications.

I showcased reliable, efficient, and robust simulations from an arbitrarily complex design setup using the principle. After establishing such connections, we are now ready to systematically evaluate and improve the computational bottlenecks on the entire engineering pipeline. The central idea in this dissertation is to develop geometric computing tools that reliably associate different representations of discrete geometry entities at diverse design and simulation stages. Instead of re-assigning properties at various phases of the design-simulation pipeline, or whenever the required budget is different, a good association between geometry at different stages gives more freedom to the design and development of automatic and adaptive solvers. The adaptation can be bi-directional: on the one hand, with coarse and concise tessellation, the algorithm utilizes the information associated with the original design to obtain more faithful simulations. On the other hand, in early-stage prototyping and automatic data generation for deep learning, the principles established to simplify the shape and retain the settings provide a fast answer for update.

For the remainder of this dissertation, chapter 2 reviews related works and our contribution in the broader technical context. Following, chapter 3 introduce the problem of surface parameterization, where an injective map, from the 3D surface to the 2D domain, assignes a planar u-v coordinate to each of the point on the surface. The algorithm guarantees the bijectivity between the surface and its codomain, which is essential to uniquely convert surface attached information into a planar representation, and take advantage of widely supported image based compression and processing techniques could for the color or material. The algorithm make use the technique of triangulation augmentation in the virtual expanded domain to fit the bijectivity condition into a mathematical optimization framework, which is able to scale up to large models with ease. To demonstrate the practical applicability, we use it to compute globally bijective single patch parametrizations, to pack multiple charts into a single UV domain, to remove self-intersections from existing models, and to deform 3D objects while preventing self-intersections.

Chapter 4 further investiages the problem of bijective attribute transfer between surfaces, where I formulate a novel approach to associate bijective multiple close-by surfaces. The algorithm converts a self-intersection free, orientable, and manifold triangle mesh into a *generalized prismatic shell* equipped with a bijective projection operator to map the mesh to a class of discrete surfaces contained within the shell whose normals satisfy a simple local condition. Properties can be robustly and efficiently transferred between these surfaces using the prismatic layer as a common parametrization domain. The combination of the prismatic shell construction and corresponding projection operator is a robust building block readily usable in many downstream applications, including the solution of PDEs, displacement maps synthesis, Boolean operations, tetrahedral meshing, geometric textures, and nested cages.

Chapter 5 incorporates the shell into two mesh generation tasks: a conforming tetrahedral meshes, and a near-surface high order shell meshes. Combined, consitutes a robust and automatic algorithm to convert dense piece-wise linear triangle meshes with feature annotated into coarse tetrahedral meshes with curved elements. The construction guarantees that the high-order meshes are free of element inversion or self-intersection. The user can specify a maximal geometrical error from the input mesh, which controls the density of the curved approximation. The boundary of the output mesh is in bijective correspondence to the input, enabling to transfer attributes between them, such as boundary conditions for simulations, making it an ideal replacement or

complement for the original input geometry. We demonstrate the robustness and effectiveness of our algorithm generating high-order meshes for a large collection of complex 3D models.

While being useful in our various applications that I design and investiagte, I believe that our geometry constructions can have greater benefit, if incorporated into many existing implementations. While straightforward in theory, such integration often involves error prone software engineering efforts, especially when our goal is to arrive at robust algorithms. Therefore, an abstraction and novel mesh editing specification pardigm is introduced in Chapter 6. Such an algorithm developement toolkit aims to make our contribution accesible for more researchers in the field between robust geometry modeling, and scalable physical simulation. We introduce a novel approach to describe mesh generation, mesh adaptation, and geometric modeling algorithms relying on changing mesh connectivity using a high-level abstraction. The main motivation is to enable easy customization and development of these algorithms via a declarative specification consisting of a set of per-element invariants, operation scheduling, and attribute transfer for each editing operation. We demonstrate that widely used algorithms editing surfaces and volumes can be compactly expressed with our abstraction, and their implementation within our framework is simple, automatically parallelizable on shared-memory architectures, and with guaranteed satisfaction of the prescribed invariants. These algorithms are readable and easy to customize for specific use cases. We introduce a software library implementing this abstraction and providing automatic shared memory parallelization.

In addition to the mathematical formulation and algorithmic design, we also provide open-source implementations for the algorithms in the thesis. Additionally, we produce source code [Jiang 2017, 2021b,a] for each of the algorithm implementations, along with reproducible instructions, to foster further research in this direction.

1.1 Preliminaries

1.1.1 Finite Element Method

Partial Diferential equations describe etc. Finite element method is a widely used methods for engineering analysis and design. And high order

1.1.2 Unstructured Meshes

And their edit, optimization, simplification.

1.1.3 Computational Geometry

1.1.4 Numerics and predicates

1.1.5 maps

1.1.6 mathematical optimization

2 Related Works

2.1 Meshes in Computer Graphics

2.1.1 Mesh Data Structures

Efficient data structures for representing solid geometry have been an intriguing research topic since the early days of computer graphics [Requicha 1980]. As a result, there is a large variety of mesh data structure designs, where they are each optimized for different usage scenarios. Index-array-based mesh data structure encodes each element as a list of vertex indices on its boundary. It is simple and memory efficient, but neighborhood query and local operations are not directly supported. Graph-based mesh data structures, including half-edge [Mäntylä 1987], winged-edge [Baumgart 1972], quad-edge [Guibas and Stolfi 1985], cell-tuple [Brisson 1989], etc., view meshes as graphs, where each element contains links to its adjacent elements. This design allows for efficient local query and update, making it ideal for algorithms like mesh simplification [Garland and Heckbert 1997]. Linear-algebra-based mesh data structures, such as [DiCarlo et al. 2014; Zayer et al. 2017; Mahmoud et al. 2021], encode adjacency information as sparse matrices. This design elegantly reduces neighborhood query and local operations to sparse matrix computations, which are highly optimized for modern parallel computing architecture. Closely related, is the concept of generalized combinatorial maps [Lienhardt 1994; Dufourd 1991], and the CGoGN library [Kraemer et al. 2014] provide an efficient implementation which provide parallel traversal of the mesh. By design, mesh data structures provide a low-level interface to manipulate vertices, edges, faces, and tetrahedra. Different designs differ vastly in API and implementation details, making it hard to port algorithm from one data structure to another. Our contribution mostly focus on algorithm specification, and is benefitial regardless of the choice of underlying mesh data structure.

2.1.2 Domain specific languages in graphics

In our final chapter 6, the abstraction model of mesh processing algorithms draw inspiration from domain specific languages (DSL) in graphics. For dense regularly structured data such as images, Halide [Ragan-Kelley et al. 2013] popularized the idea of decoupling image processing operations from low level scheduling tasks. Similar abstraction that separates algorithm description from low level data structure and/or parallel architecture can also be found in other DSLs such as Simit [Kjolstad et al. 2016] for simulation over triangle meshes, Taco [Kjolstad et al. 2017] for dense and sparse tensor algebra, Taichi [Hu et al. 2019a] for simulation over sparse volumetric data, and

Penrose [Ye et al. 2020] for generating diagrams from math notation.

2.2 Maps and Parameterizations

Injective Surface Parameterizatons

Bijective maps find a host of applications in a variety of fields including physical simulation, surface deformation, and parametrization. We review only the most relevant prior works here and refer to the following surveys for more details [Floater and Hormann 2005; Sheffer et al. 2006; Hormann et al. 2007]. There are many methods that focus on creating locally injective maps, which amounts to requiring that triangles maintain their orientation (i.e. they do not flip). In mesh parameterization, many flip-preventing metrics have been developed: the idea is to force the metric to diverge to infinity as triangles become degenerate, inhibiting flips. These metrics optimize various geometric properties such as angle [Hormann and Greiner 2000; Degener et al. 2003] or length [Sander et al. 2001; Sorkine et al. 2002; Aigerman et al. 2014; Poranne and Lipman 2014; Smith and Schaefer 2015] preservation. Similar techniques in the context of deformation have been used to add barrier functions to enforce local injectivity in deformations [Schüller et al. 2013]. Our method uses these techniques to prevent flips in the scaffold.

Many methods have also been developed to optimize these distortion energies including moving one vertex at a time [Hormann and Greiner 2000], parallel gradient descent [Fu et al. 2015], as well as other quasi-newton approaches [Smith and Schaefer 2015; Kovalsky et al. 2016; Rabinovich et al. 2017]. Other approaches construct such maps by performing a change of basis, projecting to an inversion free space, and then constructing a parametrization from the result [Fu and Liu 2016]. While our method could potentially use any of these optimization methods, we use [Rabinovich et al. 2017] for its large step sizes. We elaborate on this choice in Section **??**.

In addition to (locally) injective constraints, bijective maps have the additional requirement that the boundary does not intersect. One simple method for creating a bijective map in 2D involves constraining the boundary to a convex shape such as a circle [Tutte 1963; Floater 1997]. Strictly speaking, such bijective is actually injective. Such parametrizations guarantee a bijective map in 2D but create significant distortion. Even so, these methods are commonly used to create a valid starting point for further optimization [Schüller et al. 2013; Smith and Schaefer 2015; Rabinovich et al. 2017]. While methods that produce bijective maps with fixed boundaries exist [Weber and Zorin 2014; Campen et al. 2016], we aim to produce maps where the boundary is free to move to reduce the distortion of the map.

[Gotsman and Surazhsky 2001; Surazhsky and Gotsman 2001] introduced the concept of scaffolding where the free space is triangulated for the purpose of morphing without self-intersection. In [Zhang et al. 2005], the scaffold triangles are given a step function for their error: zero if not flipped, otherwise infinity. Hence, the bijective condition becomes local in that the shape can evolve until a scaffold triangle flips, in which case the free space is retriangulated and the optimization continues. The main limitation of this work is the lack of an evolving triangulation during the line search and the absence of a rotationally invariant metric for the scaffold triangles, which lead to very small steps and an inefficient optimization.

The Deformable Simplicial Complex (DSC) method [Misztal and Bærentzen 2012] utilize a triangulation of both the free space and the interior of an object to track the interface between the two volumes. Similar to [Zhang et al. 2005], the DSC retriangulates at degeneracies but also performs operations to improve the shape of the triangles. This method changes the triangulation of the interface that it tracks, which works well for simulation, but it is not allowed in many other applications such as UV mapping.

Air meshes [Müller et al. 2015] extends the technique of Zhang et al. [Zhang et al. 2005] to add the concept of triangle flipping based on a quality measure during the optimization instead of simply retriangulating at the first sign of a degeneracy. However, this method does not maintain bijective maps as boundaries are allowed to inter-penetrate during optimization: the scaffold is only used to efficiently detect problematic regions, and the local injectivity requirement is a soft constraint in the optimization. The problem tackled in this chapter is much harder, because we do not allow any overlap during any stage of the optimization to guarantee that the resulting maps will be bijective.

[Smith and Schaefer 2015] take a different approach: instead of using a scaffold triangulation, the authors introduce a locally supported barrier function for the boundary to prevent intersection and explicitly limit the line search by computing the singularities of both the distortion energy and the boundary barrier function. Such an approach is inspired by traditional collision detection and response methods that are discussed below. Given a bijective starting point, this approach never leaves the space of bijective maps during optimization. Its main limitation is that it is computationally expensive, especially for large models. Our method is two order of magnitude faster (Figure **??**).

Global Parametrization Conformal mesh parametrization algorithms adapt the mesh during optimization, as a fixed triangulation restricts the space of metrics realizable [Luo 2004; Campen and Zorin 2017; Campen et al. 2019; Gu et al. 2018b,a; Springborn 2020; Sun et al. 2015]. Two very recent works [Gillespie et al. 2021; Campen et al. 2021] introduce robust algorithms based on Ptolemy flips to compute conformal maps satisfying a prescribed metric.

All these methods require changing the mesh connectivity of a triangle mesh, and could thus benefit from our framework in Chapter 6 to simplify their implementation and parallelize the mesh editing operations.

Texture Seam Creation.

In the context of parametrization, some approaches optimize the connectivity of the charts of the surface during parametrization to obtain a bijective map. [Lévy et al. 2002; Zhou et al. 2004] parameterize the surface and then split charts based on whether they intersect [Lévy et al. 2002] or based on a level of distortion [Zhou et al. 2004]. Sorkine et al. [Sorkine et al. 2002] employ a bottom-up approach and add triangles to a parametrization chart until bijectivity would be violated. The problem we are solving is more general (seams are only useful for texture mapping applications) and constrained (we preserve the prescribed seams). Our algorithm could be used by these algorithms to parametrize single charts, which could reduce the number of additional seams.

2.2.1 Isotopy between surfaces

[Chazal and Cohen-Steiner 2005; Chazal et al. 2010] presents conditions for two sufficiently smooth surfaces to be isotopic. Specifically, the projection operator is a homeomorphism. [Mandad et al. 2015] extends this idea to make an approximation mesh that is isotopic to a region. However, they did not realize a map suitable for transferring attributes.

Shell Maps 2.5D geometric textures, defined around a surface, are commonly used in rendering applications [Wang et al. 2003, 2004; Porumbescu et al. 2005; Peng et al. 2004; Lengyel et al. 2001; Chen et al. 2004; Huang et al. 2007; Jin et al. 2019]. The requirement is to have a thin shell around the surface that can be used to map an explicit mesh copied from a texture, or a volumetric density field used for ray marching. Our shells are naturally equipped with a 2.5D parametrization that can be used for these purposes, and have the advantage of allowing users to generate coarse shells which are efficient to evaluate in real-time. The bijectivity of our map ensures that the volumetric texture is mapped in the shell without discontinuities. We show one example in Section **??**.

2.3 Attribute Association and Transfer

Common Domains Or cross parameterizations?

A different approach to transfer attributes is to map both the source and the target to a common parametrization domain, and to compose the parametrization of the source domain with the inverse parametrization of the target domain to define a map from source to target. In the literature, there are methods that map triangular meshes to disks [Tutte 1963; Floater 1997], region of a plane [Maron et al. 2017; Aigerman et al. 2015, 2014; Schüller et al. 2013; Smith and Schaefer 2015; Rabinovich et al. 2017; Jiang et al. 2017; Weber and Zorin 2014; Campen et al. 2016; Müller et al. 2015; Gotsman and Surazhsky 2001; Surazhsky and Gotsman 2001; Zhang et al. 2005; Fu and Liu 2016; Litke et al. 2005; Schmidt et al. 2019], a canonical coarse polyhedra [Kraevoy and Sheffer 2004; Praun et al. 2001], orbifolds [Aigerman and Lipman 2015; Aigerman et al. 2017; Aigerman and Lipman 2016], Poincare disk [Springborn et al. 2008; Stephenson 2005; Kharevych et al. 2006; Jin et al. 2008], spectral basis [Ovsjanikov et al. 2012; Shoham et al. 2019; Ovsjanikov et al. 2017], and abstract domains [Kraevoy and Sheffer 2004; Schreiner et al. 2004; Pietroni et al. 2010]. While these approaches allow mappings between completely different surfaces, this is a hard problem to tackle in full generality fully automatically, with guarantees on the output (even some instances of the problem of global parametrization, i.e. maps from a specific type of almost everywhere flat domains to surfaces, lack a fully robust automatic solution).

Our approach uses a coarse triangular domain embedded in ambient space as the parametrization domain, and uses a vector field-aligned projection within an envelope to parametrize close-by surfaces bijectively to the coarse triangular domain. Compared to the methods listed above, our approach has both pros and cons. Its limitation is that it can only bijectively map surfaces that are similar to the domain, but on the positive side, it: (1) is efficient to evaluate, (2) guarantees an exact bijection (it is closed under rational computation), (3) works on complex, high-genus models, even with low-quality triangulations, (4) less likely to suffer from high distortion (and the

related numerical problems associated with it), often introduced by the above methods.

We see our method not as a replacement for the fully general surface-to-surface maps (since it cannot map surfaces with large geometric differences), but as a complement designed to work robustly and automatically for the specific case of close surfaces, which is common in many geometry processing algorithms, as well as serve as a foundation for generating such close surfaces (e.g., surface simplification and improvement, see Section **??**)

2.3.1 Shell Overview

We review works in computer graphics spanning both the realization of maps for attribute transfer (Section 2.3.2), and the explicit generation of boundary cages (Section **??**), which are closest to our work.

2.3.2 Attribute Transfer

Transferring attributes is a common task in computer graphics to map colors, normals, or displacements on discrete geometries. The problem is deeply connected with the generation of UV maps, which are piecewise maps that allow to transfer attributes from the planes to surfaces (and composition of a UV map with an inverse may allow transfer between surfaces). We refer to [Floater and Hormann 2005; Sheffer et al. 2006; Hormann et al. 2007] for a complete overview, and we review here only the most relevant works.

Projection Modifying the normal field of a surface has roots in computer graphics for Phong illumination [Phong 1975], and tessellation [Boubekeur and Alexa 2008]. Orthographic, spherical, and cage based projections are commonly used to transfer attributes, even if they often leads to artifacts, due to their simplicity [Community 2018; Nguyen 2007]. Projections along a continuously-varying normal field has been used to define correspondences between neighbouring surfaces [Kobbelt et al. 1998; Lee et al. 2000; Panozzo et al. 2013; Ezuz et al. 2019], but it is often discontinuous and non-bijective. While the discontinuities are tolerable for certain graphics applications (and they can be reduced by manually editing the cage), these approaches are not usable in cases where the procedure needs to be automated (batch processing of datasets) or when bijectivity is required (e.g., transfer of boundary conditions for finite element simulation). These types of projection may be useful for some remeshing applications to eliminate surface details [Ebke et al. 2014], but it makes these approaches not practical for reliably transferring attributes. Our shell construction, algorithms, and associated projection operator, can be viewed as guaranteed continuous bijective projection along a field.

Attribute Tracking In the specific context of remeshing or mesh optimization, algorithms have been proposed to explicitly track properties defined on the surface [Garland and Heckbert 1997; Cohen et al. 1997; Dunyach et al. 2013] after every local operation. By following the operations in reverse order, it is possible to resample the attributes defined on the input surface. These methods are algorithm specific, and provide limited control over the distortion introduced in the mapping.

Our algorithm provides a generic tool that enables any remeshing technique to obtain such a map with minimal modifications.

2.3.3 Envelopes

Explicit [Cohen et al. 1996, 1997] or implicit [Hu et al. 2016] envelopes have been used to control geometric error in planar [Hu et al. 2019b], surface [Guéziec 1996; Hu et al. 2017; Cheng et al. 2019], and volumetric [Hu et al. 2018, 2019c] remeshing algorithms. Our shells can be similarly used to control the geometric error introduced during remeshing, but they offer the advantage of providing a bijection between the two surfaces, enabling to transfer attributes between them without explicit tracking [Cohen et al. 1997]. We show examples of both surface and volumetric remeshing in Section **??**. Also, [Barnhill et al. 1992; Bajaj et al. 2002] utilize envelopes for function interpolation and reconstruction, where our optimized shells can be used for similar purposes.

2.3.4 Mesh Arrangements/Boolean Operations

Boolean operations are basic algorithms often used in geometry processing applications. Recently, [Zhou et al. 2016] proposed a robust way to compute them by constructing a space arrangement, and then filtering the result using the generalized winding number [Jacobson et al. 2013]. A similar approach, using an approximated meshing algorithm, has been extended in [Hu et al. 2019c], using a tetrahedral mesher to create the initial arrangement. The reimplementation of TetWild introduced in this paper (Section **??**) can be extended for a similar purpose.

2.3.5 Collision and Animation

While not directly related to our approach, bijective maps inherently involve some form of collision detection and response to avoid overlaps. The field on collision detection is vast, and we refer the reader to a survey [Jimenez et al. 2001]. In terms of simulations, methods such as asynchronous contact mechanics [Harmon et al. 2009; Harmon 2010; Ainsley et al. 2012] ensure the bijective property but are very expensive and designed to operate as part of a simulation. Our approach in Chapter 3 is specialized for geometric optimization, where we are interested in a quasi-static solution (i.e. we do not want to explicitly simulate a dynamic system, but only find an equilibrium solution).

The work that is closer to Chapter 3 in term of application (but very different in term of formulation) is [Harmon et al. 2011], where collision detection and response is used to interactively deform shapes while avoiding self-intersections. Similarly to the previous methods, the explicit detection and iterative response is expensive when many collisions happen at the same time. Our work avoids these expensive computations, and can robustly handle hundreds of simultaneous collisions while still making large steps in the optimization.

In Chapter 4 and 5 Converting triangle meshes into coarse cages is useful for many applications in graphics [Sacht et al. 2015], including proxies for collision detection [Calderon and Boubekeur 2017] and animation cages [Thiery et al. 2012]. While not designed for this application, our shells can be computed recursively to create increasingly coarse nested cages. We hypothesize that a

bijective map defined between all surfaces of the nested cages could be used to transfer forces from the cages to the object (for collision proxies), or to transfer handle selections (for animation cages). [Botsch et al. 2006; Botsch and Kobbelt 2003] uses a prismatic layer to define volumetric deformation energy, however their prisms are disconnected and only used to measure distortion. Our prisms could be used for a similar purpose since they explicitly tesselate a shell around the input surface.

Curved tetrahedral meshes have also gained popularity in the context of fast animation. With fewer degrees of freedom and preserved geometric fidelity, [Mezger et al. 2007] observe the benefit of quadratic tetrahedra in the pipeline of physically based animation. [Suwelack et al. 2013] further investigate the transfer problem when using curved meshes as a proxy.

2.4 Mesh Generation

2.4.1 Curved Surface Meshes

There are many algorithms for fitting curved *surfaces* to dense 3D triangle meshes. The most popular approaches fit spline patches, usually on top of a quadrangular grid. Since generating quadrilateral meshes is a challenging problem for which robust solutions do not exist yet, we refer to [Bommes et al. 2012] for an overview, and only review in this section algorithms for unstructured curved mesh generation, which are more similar to our algorithm. We note that the focus of our paper is volumetric meshing: while we generate an intermediate curved surface mesh, this is not the goal of our algorithm, especially since the generated surface is only C^0 on edges.

[Hoppe et al. 1994] fits a smooth surface represented by a point cloud to a curved triangle mesh based on a subdivision surface scheme and an interleaving mesh simplification and fitting pipeline that preserves sharp features. The algorithm does not provide explicit correspondence to the input: they are defined using distance closest point, which is not bijective far from the surface.

[Krishnamurthy and Levoy 1996] converts dense irregular polygon meshes of arbitrary topology into coarse tensor product B-spline surface patches with accompanying displacement maps. Based on the work [Lin et al. 2007] that fits triangle surface meshes with Bézier patches, [Zhang et al. 2011] fits triangle surface meshes with high-order B-spline quadrilateral patches and adaptively subdivide the patches to reduce the fitting error. These methods produce smooth surfaces but do not have feature preservation.

Another related topic is the definition of smooth parametric surfaces interpolating triangle meshes. We refer to [Zorin 2000] for an overview of subdivision methods and discuss here the approaches closer to our contribution.

[Hahmann and Bonneau 2003] proposed to use triangular Bézier patches to define smooth surfaces over arbitrary triangle meshes ensuring tangent plane continuity by relaxing the constraint of the first derivatives at the input vertices. Following Hahmann's work, [Yvart et al. 2005b] presents a more complete pipeline: perform QEM simplifications, trace the coarse mesh onto the dense one and perform parameterization relaxing and smoothing. Then it fits a hierarchical triangular spline [Yvart et al. 2005a] to the surface. More recent work [Tong and Kim 2009] approximates the triangulation of an implicit surface with a G^1 surface. These schemes are usually

designed to interpolate existing meshes rather than simplifying a dense linear mesh into a coarse curved mesh and are thus orthogonal to our contribution.

2.4.2 Curved Tetrahedral Mesh Generation

High-order meshes are used in applications in graphics [Bargteil and Cohen 2014; Mezger et al. 2009; Suwelack et al. 2013] and engineering analysis [Jameson et al. 2002] where it is important to reduce the geometric discretization error [Babuska and Guo 1988; Babuška and Guo 1992; Bassi and Rebay 1997; Luo et al. 2001; Oden 1994], while using a low number of degrees of freedom. The creation of high-order meshes is typically divided into three steps: (1) linear meshing of the smooth input surface, (2) curving of the linear elements to fit the surface, and (3) optimization to heal the elements inverted during curving. We first cover steps 2 and 3, and postpone the overview of linear tetrahedral algorithms to Section 2.4.4.

Direct methods. Direct methods are the simplest family of curving algorithms, as they explicitly interpolate a few points of the target curved surface or project the high-order nodes on the curved boundary [Dey et al. 1999; Ghasemi et al. 2016; Moxey et al. 2015; Abgrall et al. 2012; Sherwin and Peiró 2002; Turner 2017; Marcon et al. 2019]. The curved elements are represented using Lagrange polynomials, [Dey et al. 1999; Peiró et al. 2008], quadratic or cubic Bézier polynomials [George and Borouchaki 2012; Lu et al. 2013; Luo et al. 2002], or NURBS [Engvall and Evans 2016, 2017]. [Shephard et al. 2005; Sherwin and Peiró 2002] further optimizes the high-order node distribution according to geometric quantities of interest, such as length, geodesic distance, and curvature. In the case where no CAD information is available, [Wang et al. 2016], [Jiao and Wang 2012] use smooth reconstruction to compute high-order nodes and perform curving.

Deformation methods Deformation methods consider the input linear mesh as a deformable, elastic body, and use controlled forces to deform it to fit the curved boundary. Different physical models have been employed such as linear, [Abgrall et al. 2014, 2012; Dobrzynski and El Jannoun 2017; Xie et al. 2013], and (variants of) non-linear elasticity [Persson and Peraire 2009; Moxey et al. 2016; Fortunato and Persson 2016]. A comparison between different elasticity and distortion energies is presented in [Poya et al. 2016; Turner et al. 2016; Dobrev et al. 2019].

Direct and deformation methods have been tested on small collections of simple models, and, to the best of our knowledge, none of them can provide guarantees on the validity of the output or has been tested on large collections of models. There are also no reference implementations we could compare against.

Inversions and Intersections. Most of these methods introduce inverted elements during the curving of the high-order elements. Inverted elements can be identified by extending Jacobian metrics for linear elements [Knupp 2002, 2000] to high-order ones [Engvall and Evans 2018; Peiró et al. 2014; Johnen et al. 2013; Poya et al. 2016; Roca et al. 2012]. Various untangling strategies have been proposed, including geometric smoothing and connectivity modifications [Cardoze et al. 2004; Dey et al. 1999; Gargallo Peiró et al. 2013; George and Borouchaki 2012; Lu et al. 2013; Luo

et al. 2002; Peiró et al. 2008; Shephard et al. 2005; Dobrzynski and El Jannoun 2017; Gargallo-Peiró et al. 2015; Geuzaine et al. 2015; Roca et al. 2012; Ruiz-Gironés et al. 2017, 2016a,b; Stees and Shontz 2017; Steve L. Karman and Stefanski 2016; Toulorge et al. 2013, 2016; Ziel et al. 2017; Dobrev et al. 2019; Turner 2017; Luo et al. 2008; Lu et al. 2014]. None of these techniques can guarantee to remove the inverted elements.

An alternative approach is to start from an inversion-free mesh and slowly deform it [Persson and Peraire 2009; Ruiz-Gironés et al. 2017], explicitly avoiding inversions at the cost of possibly inaccurate boundary reproductions. These methods cannot however guarantee that the boundary will not self-intersect. Our approach follows a similar approach but uses a geometric shell to ensure element validity and prevention of boundary self-intersections.

Curved Optimal Delaunay Triangulation [Feng et al. 2018] generalize optimal delaunay triangulation paradigm to the high-order setting, through iteratively update vertices and connectivity. Their algorithm starts with a point cloud sampled from triangle meshes. However, the success of the method depends on the choice of final vertex number and sizing field, where insufficient vertices may result in broken topology or invalid tetrahedral meshes.

Software Implementation. Despite the large literature on curved meshing generation, there are very few implementations available. Nektar [Moxey et al. 2018] is a finite element software with a meshing component, which can generate high-order elements. Gmsh [Geuzaine and Remacle 2009] is an open source software that supports the curved meshing of CAD models, but it does not support dense linear meshes as input. Despite the difference in the input type, we provide a comparison with Gmsh in Section **??**, as it is the only method that we could run on a large collection of shapes. To the best of our knowledge, the commercial software that support curved meshing (Pointwise [Pointwise 2018; Steve L. Karman and Stefanski 2016]) are also requiring a CAD model as input.

2.4.3 Curved Structured Mesh Generation

The use of a hexahedral mesh as a discretization for a volume, allows to naturally define C^k splines over the domain, which can be used as basis functions for finite element methods: this idea has been pioneered by IsoGeometric Analyisis (IGA), and it is an active research area. The generation of volumetric, high-order parametrizations that conform to a given input geometry is an extremely challenging problem [Sorger et al. 2014; Peiró et al. 2015]. Most of the existing methods rely on linear hexahedral mesh generation, which is on its own a really hard problem for which automatic and robust solutions to generate coarse meshes are still elusive [Li et al. 2012; Gao et al. 2019; Guo et al. 2020; Palmer et al. 2020; Zhang et al. 2020; Marschner et al. 2020] due to the inherently global nature of the problem. The current state of the art for IGA meshing is a combination of manual decomposition of the volume and semi-automated geometrical fitting [Yu et al. 2020; Coreform 2020].

In contrast, our approach is automatic, i.e. can automatically process thousands of models without any manual intervention, while providing explicit guarantees on both the validity of

the elements and the maximal geometric error. Its downside is the C^0 continuity of the basis on the elements' interfaces. However, it is unclear to us if this is a real limitation: in many common settings in computer graphics and mechanical engineering (Poisson problems, linear and non-linear elasticity) the higher smoothness offered by spline functions does not make noticeable difference experimentally [Schneider et al. 2019] (and it actually leads to much worse conditioning of the system matrix, which is problematic for iterative solvers), and we thus believe that curved tetrahedral meshing is a very exciting alternative as it dramatically simplifies both the meshing and fitting of high-order elements.

2.4.4 Boundary Preserving Tetrahedral Meshing

We refer to [Hu et al. 2018] for a detailed overview of linear tetrahedral meshing, and we focus here only on the techniques that target boundary preserving tetrahedral meshing.

The most popular linear tetrahedral meshing methods are based on *Delaunay refinement* [Chew 1993; Shewchuk 1998; Ruppert 1995], i.e. the insertion of new vertices at the center of the circumscribed sphere of the worst tetrahedron in term of radius-to-edge ratio. This approach is used in the most popular tetrahedral meshing implementations [Si 2015; Jamin et al. 2015], and, in our experiments, proved to be consistently successful as long as the boundary is allowed to be refined. A downside of these approaches is that a 3D Delaunay mesh, unlike the 2D case, might still contain "sliver" tetrahedra, thus requiring mesh improvement heuristics [Cheng et al. 2000; Du and Wang 2003; Alliez et al. 2005; Tournois et al. 2009]. [Alexa 2019] discusses this issue in detail and provides a different formulation to avoid it without the use of a postprocessing. [Alexa 2020] introduces an approach that does not allow insertion of Steiner points, making it not suitable for generic polyhedra domains.

There are many variants of Delaunay-based meshing algorithms including *Conforming Delaunay tetrahedralization* [Murphy et al. 2001; Cohen-Steiner et al. 2002], *constrained Delaunay tetrahedralization* [Chew 1989; Si and Gärtner 2005; Shewchuk 2002; Si and Shewchuk 2014], and *Restricted Delaunay tetrahedralization* [Cheng et al. 2008; Boissonnat and Oudot 2005; Engwirda 2016].

To the best of our knowledge, all these methods are designed to allow some modifications of the input surface (either refinement, resampling, or approximation). One exception is the constrained Delaunay implementation in TetGen [Si 2015] that allows disabling any modification to the boundary. However, this comes at the cost of much lower quality and potential robustness issues, as we show in Appendix B.3.

A different tetrahedral meshing approach has been proposed in [Hu et al. 2018], and its variants [Hu et al. 2020, 2019b], where the problem is relaxed to generate a mesh that is close to the input to increase robustness. However, these approaches are not directly usable in our setting, as we require boundary preservation.

Due to these issues, we propose a novel boundary-preserving tetrahedral meshing algorithm specifically tailored for the shell mesh generated by our curved meshing algorithm.

2.4.5 Tetrahedral Mesh Generation

Tetrahedral meshing algorithms heavily rely on mesh editing operations. The most common approaches are Delaunay methods [Shewchuk 1998; Ruppert 1995; Remacle 2017; Du and Wang 2003; Alliez et al. 2005; Tournois et al. 2009; Murphy et al. 2001; Cohen-Steiner et al. 2002; Chew 1989; Si and Gärtner 2005; Shewchuk 2002; Si and Shewchuk 2014; Cheng et al. 2008; Boissonnat and Oudot 2005; Jamin et al. 2015; Dey and Levine 2008; Chen and Xu 2004; Shewchuk 1996; Cheng et al. 2012; Bishop 2016; Busaryev et al. 2009; Boissonnat et al. 2002; Si 2015], which strive to generate meshes satisfying the Delaunay condition, grid methods [Yerry and Shephard 1983; Bern et al. 1994; Molino et al. 2003; Bronson et al. 2013; Labelle and Shewchuk 2007; Doran et al. 2013; Bridson and Doran 2014], which start from a regular lattice or with a hierarchical space partitioning and optionally intersect the background mesh with the input surface, and front-advancing methods [Sadek 1980; Cuilliere et al. 2013; Alauzet and Marcum 2014; Haimes 2014], which insert one element at a time, growing the volumetric mesh (i.e. marching in space), until the entire volume is filled .

These algorithms rely on local operations on mesh data-structures, and benefit from our framework to simplify the implementation and gain automatic parallelization. We discuss an implementation of one the more recent algorithms [Hu et al. 2018, 2019c] in Section **??**. Note that some of these algorithms use local operation that are not implemented yet (such as 5-6 swap), but they could be added to our framework.

2.4.6 Constraints in Meshing.

Downstream applications often require meshes to satisfy either quality (avoidance of zero volume elements) or geometric (distance to the input surface) constraints. For example, Mandad et al. [2015] creates a surface approximation within a tolerance volume, the TetWild algorithms [Hu et al. 2018, 2019c] use an envelope [Wang et al. 2021] to restricts the geometry of the boundary of the tetrahedral mesh, [Brochu et al. 2012] adds constraints to local remeshing to avoid interpenetrations in simulations, and [Gumhold et al. 2003] extends mesh simplification [Garland and Heckbert 1999; Popović and Hoppe 1997] to ensure a non self-intersecting result.

These criteria are explicitly modeled as invariants in our framework, and they can be easily swapped in and out existing implementations, as we demonstrate in Section **??**.

2.4.7 Mesh Improvement.

Mesh improvements modifies an existing mesh by changing its connectivity and position of the vertices to improve the quality of its elements [Canann et al. 1996, 1993; A. Freitag and Ollivier-Gooch 1998; Lipman 2012; Chen and Xu 2004; Alliez et al. 2005; Feng et al. 2018; Hu et al. 2018; Alexa 2019; Klingner and Shewchuk 2007]. We show in Section **??** a reimplementation of [Alexa 2019] in IDAS form.

2.4.8 Dynamic Remeshing and Refinement

Simulations involving large deformations are common in computer graphics, and if the surface or volume deformed is represented by a mesh, it is inevitable that after a large deformation the quality of the elements will deteriorate, and the mesh will have to be updated. Additionally, it is often required to concentrate more elements in regions of interest whose location is changing during the simulation, for example to capture a fold in a cloth simulation, or a fracture in a brittle material. These two challenges are tackled in elastoplastic and viscoplastic simulations [Hutchinson et al. 1996; Bargteil et al. 2007; Wicke et al. 2010; Wojtan and Turk 2008], in fluid simulations [Misztal and Bærentzen 2012; Klingner et al. 2006; Ando et al. 2013; Chentanez et al. 2007; **?**], in cloth simulation [Villard and Borouchaki 2002; Bender and Deul 2013; Li and Volkov 2005; Narain et al. 2012, 2013; Pfaff et al. 2014; Simnett et al. 2009], and fracture simulation [Busaryev et al. 2013]. All these algorithms could benefit from our contribution, to simplify their implementation and obtaining speedup due to the automatic parallelization offered by our approach.

A different approach is discussed in [Grinspun et al. 2002], where the refinement is performed on the basis to avoid the difficulties with explicit remeshing. However, this approach cannot coarsen a dense input, and also cannot increase the quality of elements, making it usable only for specific scenarios [Grinspun et al. 2002]. Our approach aims at lowering the barrier for integrating explicit remeshing algorithms in simulation applications, thus allowing to directly use standard simulation methods on adaptive meshes without having to pay the high implementation cost for the mesh generation.

When remeshing is paired with algorithms simulating contacts that do not tolerate interpenetrations (for example [Li et al. 2020]), it is necessary to ensure that adaptive remeshing does not break this invariant. This can be achieved adding non-penetration constraints to each local mesh editing operations, as proposed in [Brochu et al. 2012]. Our framework is ideal for developing such methods, as additional constraints can be added to existing mesh editing algorithms with minimal modifications, as we demonstrate in Section **??**.

2.4.9 Parallel Meshing

To meet the demand of generating large meshes, a number of popular mesh generation algorithms have been redesigned to leverage modern parallel computing hardware, both in a shared memory and distributed memory setting. Typically a divide-and-conquer strategy is adopted where a mesh is partitioned to run local processing operations on each subdomain in parallel. There are two key challenges involved: (1) how to handle operations involving elements shared by multiple partitions; (2) how to ensure load stay balanced across different processors as the mesh evolves.

One way to mitigate both challenges is to ensure mesh is partitioned into similar sized patches with high area to boundary ratio. A large number of partitioning strategies are available, including clustering-based approaches [Mahmoud et al. 2021], spacial-hierarchy-based approach [Loseille et al. 2017; Lo 2012], space-filling-curve-based approach [Marot et al. 2019; Borrell et al. 2018], and general purpose graph partitioning [Karypis and Kumar 1998]. Many variations of space-filling curves have also been used to construct mesh partitions [Chrisochoides 2006; Aluru and Sevilgen 1997]. To handle potential conflicts that may arise at partition boundaries, various

synchronization strategies have been proposed [Okusanya and Peraire 1996; Chrisochoides and Nave 2003; Chrisochoides 2006] to minimize the amount of communication.

After generating the submeshes, some methods allow each compute node to work on them independently without synchronization. Once all threads are done, the meshes are merged [Cignoni et al. 1993; Chen 2010; Funke and Sanders 2017; Blelloch et al. 1999]. However these methods require complicated merge steps since the tetrahedra in the intermediate boundaries may not align. There are some techniques that compromise the Delaunay condition in some cases, so that the merging operation can be simpler [Lachat et al. 2014]. To avoid the tricky merge operations, other parallel strategies maintain a single complete Delaunay tetrahedralization and use synchronization techniques to avoid race conditions when working on a partition boundary [Okusanya and Peraire 1997; Chrisochoides and Nave 2003]. The parallel constrained Delaunay meshing algorithm [Chew et al. 1997] cleverly defines the boundary and edge constrains to reduce the variable and unpredictable communication patterns. Some other techniques use locks for handling conflicts and data races [Blandford et al. 2006; Batista et al. 2010; Foteinos and Chrisochoides 2011].

Another set of methods use recursive divide-and-conquer techniques for parallel implementation on shared memory machines [Marot and Remacle 2020]. All threads independently work on the internal parts of the mesh and skip the operations at the boundary. After this phase, processing of only the boundary elements becomes the new problem. This technique is then recursively used until all the mesh elements are processed. A similar set of techniques use clever space-filling curves for re-partitioning the mesh boundaries after each recursive phase [Chrisochoides 2006; Aluru and Sevilgen 1997].

Since the submesh boundaries are the main areas of concern, some methods entirely avoid any operations on these boundaries while ensuring the correctness of the result [Galtier and George 1996; Linardakis and Chrisochoides 2006]. These methods precompute the domain separators such that their facets are Delaunay admissible. This completely eliminates synchronization overheads, but only applies for Delaunay meshing.

Another conflict handling strategy is to simply reject the offending operations and try executing them later with a new domain partitioning [Marot et al. 2019]. This reject-and-repartition strategy may not guarantee algorithm termination, thus special care is needed to handle this case.

As the domain mesh evolves, keeping load balanced across processors becomes critical. Typically, this is done by periodically repartitioning the updated mesh. Zhou et al. [2012] proposes a predictive load balancing method to keep partitions balanced. Marot et al. [2019] uses simple rescaling of the space-filling curve to repartition the domain.

In this work, we are targeting only shared-memory parallelism, thus making the problem of reducing communications between processors less relevant. We use a graph-based space partitioning technique [Karypis and Kumar 1998] due to its simplicity and availability as open-source code (METIS), but we use it only to reduce the risk of conflicts. To avoid conflicts, we use a shared memory locking mechanism. This approach is only possible for shared-memory parallelism but has the major advantage of not requiring rebalancing and to respect, to a certain degree, the execution order prescribed by the user-code. This approach is possible thanks to the availability of efficient parallel atomic instructions, and parallel libraries based on them (oneTBB).

2.4.10 Boundary Layer

Boundary layers are commonly used in computational fluid dynamics simulations requiring highly anisotropic meshing close to the boundary of objects. Their generation is considered challenging [Aubry et al. 2015, 2017; Garimella and Shephard 2000]. These methods generate shells around a given surface, but do not provide a bijective map suitable for attribute transfer.

2.5 Robust Geometry Processing

The closest works, in terms of applications, to our contribution are the recent algorithms enabling black-box geometry processing pipelines to solve PDEs on meshes *in the wild*.

[Dyer et al. 2007; Liu et al. 2015] refines arbitrary triangle meshes to satisfy the Delaunay mesh condition, benefiting the numerical stability of some surface based geometry processing algorithms. These algorithms are orders of magnitude faster than our pipeline, but, since they are refinement methods, cannot coarsen dense input models. While targeting a different application, [Sharp et al. 2019] offers an alternative solution, which is more efficient than the extrinsic techniques [Liu et al. 2015] since it avoids the realization of the extrinsic mesh (thus naturally maintaining the correspondence to the input, but limiting its applicability to non-volumetric problems) and it alleviates the introduction of additional degrees of freedom. [Sharp and Crane 2020] further generalizes [Sharp et al. 2019] to handle non-manifold and non-orientable inputs, which our approach currently does not support.

TetWild [Hu et al. 2018, 2019c] can robustly convert triangle soups into high-quality tetrahedral meshes, suitable for FEM analysis. Their approach does not provide a way to transfer boundary conditions from the input surface to the boundary of the tetrahedral mesh. Our approach, when combined with a tetrahedral mesher that does not modify the boundary, enables to remesh low-quality surface, create a tetrahedral mesh, solve a PDE, and transfer back the solution (Figure **??**). However, our method does not support triangle soups, and it is limited to manifold and orientable surfaces.

3 | Bijective Maps with Triangulation Augmentation

4 | Bijective Projection in the Shell

5 | Bijective and Coarse High-Order Tetrahedral Meshes

6 Declarative Specification for Unstructured Mesh Editing Algorithms

7 | Conclusion

In this dissertation, we have examined problems with meshes, and mappings. By robust association of attributes between different representations, our constructions have benefit robust geometric modeling and scalable physical simulation.

However, there are still much to be done.

Fast boolean.

This chapter relates to the outlook and conclusions.

Post publication trivia Bijective Parameterization has received follow ups.

Geometry + Simulations

Geometry + Vision construction

Finally, the availability of data is essential to algorithm verification and data-driven algorithm development. In collaboration with OnShape Inc. and colleagues at NYU, we collected and curated large-scale computed-aided design (CAD) models created by mechanical engineers. The dataset contains one million geometry shapes with rich information. Physical simulation on a large scale of datasets allows for machine learning techniques to take advantage, in order to assist the generative design or practical simulations.

A | Shell Specifics

A.1 Proofs

A.1.1 Theorem ??

Consider the unique (up to symmetry) piece-wise affine map T deforming Δ into a reference prism $\hat{\Delta} = \{u, v \geq 0 | u + v \leq 1\} \times [-1, 1]$, identifying the bottom surface of $\hat{\Delta}$ as $z = -1$, middle surface as $z = 0$, and top surface as $z = 1$. Let T_T be the affine map mapping tetrahedron $\boldsymbol{T}$ to the reference prism. Since the volume of all $\boldsymbol{T}$ is positive by assumption, the map T is bijective and orientation preserving. In particular, T transforms $\mathcal{V}(p)$ to the const vector $e_z = (0, 0, 1)$ since the edges of the prism are mapped to axis aligned edges of the reference prism. The projection operator $\mathcal{P}(p)$ can thus be equivalently defined as $\mathcal{P}(p) = T^{-1}(\mathcal{P}_z(T(p)))$, where $\mathcal{P}_z$ is the projection over the z-axis in the reference prism. $\mathcal{P}_z$ is bijective if the piecewise linear mesh intersecting Δ is composed of triangles with positive area (and the boundaries are mapped to the boundaries) [Lipman 2014], after being mapped to the reference Δ and projected by $\mathcal{P}_z$. In the reference domain, having positive area after projection (with a fixed boundary) is equivalent to that the dot product between the projection direction and the normal of every face is positive. In the top half of Δ, $\mathcal{P}_z$ is bijective if for every point $T(p), p \in f\ T(n_p) \cdot [0, 0, 1] = T(n_p) \cdot T(\mathcal{V}(p)) > 0$ which holds since $n_p \cdot \mathcal{V}(p) > 0$ (from the definition of section) and the fact that T is orientation preserving and thus not changing the sign of the dot product.

A.1.2 Proposition ??

For simplicity, we will show that a uniform lower bound δ for all vertices exists. By consider a triangle of T, extruded over the displacement field $\mathcal{N}$, we obtain a generalized half prism with six vertices 0, e_1, e_2, δn_0, $e_1 + \delta n_1$, $e_2 + \delta\ n_2$, with δ a positive constant. The indices of the tetrahedra are:

(0 ,2, 3, 4), (0 ,3, 4, 5), (0 ,1, 5, 3), (0 ,1, 2, 3), (1 ,2, 3, 4), (0 ,1, 2, 4), (0 ,1, 5, 4), (1 ,3, 4, 5), (1 ,2, 3, 5), (0 ,2, 5, 4), (0 ,1, 2, 5), (2 ,3, 4, 5).

For a sufficiently small δ, the linear term will dominate the higher power of δ, which we omit. The dominant volume terms are listed in the following table: ($e_{ij} := e_j - e_i$)

(0 ,2, 3, 4)	$\text{vol1}(0, e_2, \delta n_0, e_1 + \delta n_1)$	$\delta\langle e_2, n_0, e_1\rangle$
(0 ,3, 4, 5)	$\text{vol2}(0, \delta n_0, e_1 + \delta n_1, e_2 + \delta n_2)$	$\delta\langle n_0, e_1, e_2\rangle$
(0 ,1, 5, 3)	$\text{vol3}(0, e_1, e_2 + \delta n_2, \delta n_0)$	$\delta\,\langle e_1, e_2, n_0\rangle$
(0 ,1, 2, 3)	$\text{vol4}(0, e_1, e_2, \delta n_0)$	$\delta\langle e_1, e_2, n_0\rangle$
(1 ,2, 3, 4)	$\text{vol5}(e_1, e_2, \delta n_0, e_1 + \delta n_1)$	$\langle e_{12}, \delta n_0 - e_1, \delta n_1\rangle$ $= \delta\langle e_2, -e_1, n_1\rangle$
(0 ,1, 2, 4)	$\text{vol6}(0, e_1, e_2, e_1 + \delta n_1)$	$\delta\langle e_1, e_2, n_1\rangle$
(0 ,1, 5, 4)	$\text{vol7}(0, e_1, e_2 + \delta n_2, e_1 + \delta n_1)$	$\langle e_1, e_2 + \delta n_2, \delta n_1\rangle$ $= \delta\langle e_1, e_2, n_1\rangle$
(1 ,3, 4, 5)	$\text{vol8}(e_1, \delta n_0, e_1 + \delta n_1, e_2 + \delta n_2)$	$\delta\langle \delta n_0 - e_1, n_1, e_{12} + \delta n_2\rangle$ $= \delta\langle -e_1, n_1, e_2\rangle$
(1 ,2, 3, 5)	$\text{vol9}(e_1, e_2, \delta n_0, e_2 + \delta n_2)$	$\langle e_{12}, \delta n_0 - e_1, e_{12} + \delta n_2\rangle$ $= \delta\langle e_{12}, \delta n_0 - e_1, n_2\rangle$ $= \delta\langle e_2, -e_1, n_2\rangle$
(0 ,2, 5, 4)	$\text{vol10}(0, e_2, e_2 + \delta n_2, e_1 + \delta n_1)$	$\delta\langle e_2, n_2, e_1\rangle$
(0 ,1, 2, 5)	$\text{vol11}(0, e_1, e_2, e_2 + \delta n_2)$	$\delta\langle e_1, e_2, n_2\rangle$
(2, 3, 4, 5)	$\text{vol12}(e_2, \delta n_0, e_1 + \delta n_1, e_2 + \delta n_2)$	$\langle \delta n_0 - e_2, \delta n_1 - e_{12}, \delta n_2\rangle$ $= \delta\langle -e_2, e_1, n_2\rangle$

For all the 12 tetrahedra the linear term is multiplied by a determinant containing the edges of the prism. We check directly that all these determinants are positive due to the assumption that $\mathbf{C}\mathcal{N}_i > 0$.

A.1.3 Theorem ??

Consider a ball around a point p of the middle face M of the prism. If the radius $r(p) > 0$ is sufficiently small, it contains, at most, a single vertex of M, and parts of incident faces. As M is compact, there is a minimal value of r on M, and it is positive. The line along the normal, passing through v, intersects incident faces at v so it cannot intersect them at any other point. Thus, the top and bottom prism vertices can be obtained by displacing V by r in either direction along the normal. Consider the r-neighborhood of the M, i.e. the union of all balls of radius r centered at points of M. This is a convex set, containing only M and parts of vertex-adjacent faces. All top and bottom prism vertices are in this set. Thus tetrahedra connecting these vertices are also in the set, i.e., complete prisms. The fact that tetrahedra are nondegenerate is established in Proposition **??**.

A.1.4 Theorem ??

To show that $\mathcal{T}$ is a section of $\mathcal{S}$ we need to show that (1) $M_\Delta = \mathcal{T} \cap \Delta$ is a simply connected patch for any prism of $\mathcal{S}$ and (2) for each prism Δ and every point $p \in M_\Delta$ the dot product between the face normal $n(p)$ and the pillars $\mathcal{V}_i, i = 1, 2, 3$ is strictly positive.

(1) is trivial since for every prism Δ of $\mathcal{S}$, Δ contains exactly one triangle of the topological bevel refinement face of $\mathcal{T}$.

To prove (2), consider the dot products of the normals of each of the triangles in the beveled region sharing at least a vertex with M_Δ, and the vectors along the pillars of M_Δ. We use colors to refer to the triangles of beveled prisms as shown in Figure) A.1 left.

The prism corresponding to the pink triangle covers only the interior of the original triangle.

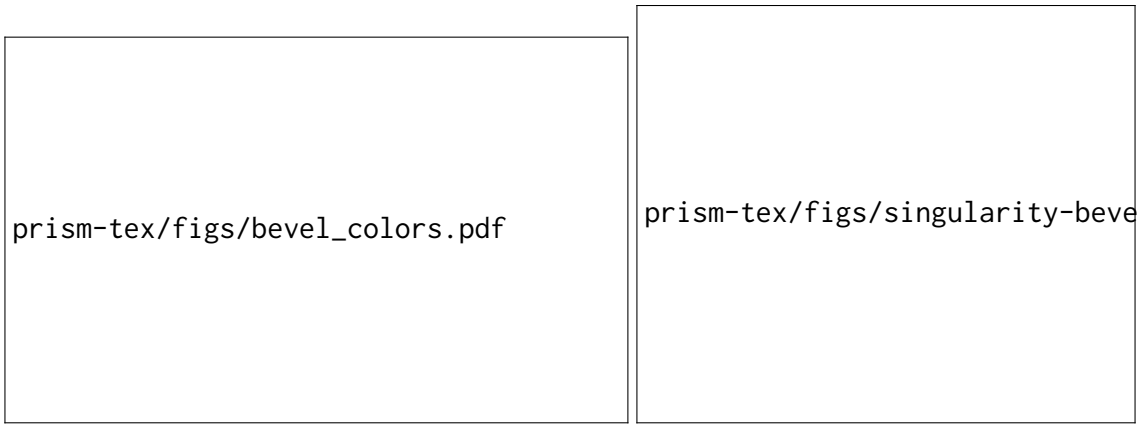

Figure A.1: Bevel patterns used in the proofs in Appendix A.1.4 and Appendix A.2

It satisfies the dot product condition since its pillars are copied from the solution of Problem (**??**) at each vertex. A similar argument can be made for the green prisms: each prism gets its pillars from the edges incident at two of the vertices, which are compatible with the normal of the adjacent triangle (e.g., red and blue pillars have positive dot product with the normal of T_1). Finally, the prisms corresponding to orange triangles cover the same one ring that was used to compute the original pillars (e.g., red pillar is compatible with T_1 and T_2). Since the same pillar is used for each of the 3 vertices of the original prism, the dot product is unchanged.

A.1.5 Theorem ??

We use A° to denote the interior of a set A. We first assert that the topological disk patch $\mathcal{T} \cap C$ coincides with $\mathcal{T} \cap C'$. To make the notation more concise, we define $\mathcal{T}_C = \mathcal{T} \cap C$ and $\mathcal{T}_{C'} = \mathcal{T} \cap C'$.

We show that $\mathcal{T}_C \subset C'$ by contradiction: assume there is $p \in \mathcal{T}_C$ but $p \notin C'$. Choose a point $q \in \partial\mathcal{T}_C$, that belongs to single prisms in C and C' and denote the enclosing prism in C as Δ_q and the corresponding one in C' as Δ'_q. It follows from the positivity of the volumes of Δ_q and Δ'_q (Assumption 1), that there is $\varepsilon > 0$ such that the half-ball (for sufficiently small ε) $\mathrm{Ball}(q, \varepsilon) \cap \Delta_q$ coincides with $\mathrm{Ball}(q, \varepsilon) \cap \Delta'_q$. Futhermore, the validity of the initial shell guarantees that there exists an interior point $\bar{q} \in \mathrm{Ball}(q, \varepsilon) \cap \Delta_q^\circ \cap \mathcal{T}$, and $\bar{q} \in \Delta_q'^\circ \cap \mathcal{T} \subset C'^\circ \cap \mathcal{T}$.

Since neither p or $\bar{q}$ lies on the boundary side triangles of $\partial C'$ (since $p, \bar{q} \notin \partial\mathcal{T}_C$), therefore there exists an interior path P (i.e., a path both in $\mathcal{T}_C$ and in $\mathcal{T}_{C'}$) that connects p and $\bar{q}$. By continuity the path P it will cross $\partial C'$ at some point $p_i \in \mathcal{T}_C$. Since the $\partial\mathcal{T}_{C'} = \partial\mathcal{T}_C$ is a fixed boundary loop (by the definition of $\mathbb{O}$), and does not contain interior points, p_i can only cross on the top surface or bottom surface of C' which contradicts Assumption 2. The roles of C and C' can be inverted to complete the proof.

We then map M_C to a regular planar n-gon D, with vertices on ∂M_C mapped cyclically to the vertices of ∂D. It follows from [Floater 2003, Corollary 6.2] that such a convex combination mapping $\varphi\colon M_C \to D$ is bijective. Then, similarly to the construction in Theorem **??**, we define $\phi\colon C \to D \times [-1, 1]$ through affine mapping induced by the thetrahedralization of the prisms Δ_i. Similarly, we define $\psi\colon C' \to D \times [-1, 1]$. The mapping ψ is also bijective because of Assumption 1 and the definition of $\mathbb{O}$ it follows that the codomain of ψ is the same as ϕ. Note that in both

cases, the map transforms the vector field $\mathcal{V}(p)$ to the constant vector field $e_z = (0, 0, 2)$, and the projection is $\mathcal{P}_z$, as defined in Appendix A.1.1.

The dot product condition (Assumption 3) ensures that $\mathcal{P}_z$ defines a bijection between D and $\psi(\mathcal{T}_C)$; and further, the image satisfies $\mathcal{P}_z(\psi(\Delta'_i \cap \mathcal{T})) = \psi(M'_i)$ where M'_i is the middle surface (a single triangle) of Δ'_i. Therefore, since ψ and $\mathcal{P}_z$ are both continuous and bijective, they are homeomorphisms which preserve topology, thus $\Delta'_i \cap \mathcal{T}$ is a simply connected topological disk.

A.2 Extension to meshes with Singularities

Most of the proofs and definition in Section **??** easily extends to meshes with singularity or pinched shells. For instance, both Theorem **??** and Definition **??** apply to pinched shells by just considering a prism made of 4 (2 for the top and 2 for the bottom slab) instead of 6. Propositions **??** and **??** both rely on per-vertex properties; by just excluding singular vertices (and neighboring prism) from the statements the proofs (and our algorithm) still holds.

Theorem **??** requires some minor additional considerations. As a consequence of beveling, no two singularities are adjacent. Therefore, we can always find a point $q \in \partial\mathcal{T}_C$ that belongs to a single prism (Appendix A.1.5) since every prism will have a positive volume because no singularities are adjacent.

Finally Proposition **??** requires a new proof since the beveling pattern used for singularities is different.

Proof. In Figure A.1 right, we highlight in red and orange the triangles not covered by the discussion in Appendix A.1.4. The prism corresponding to an orange middle triangle intersects the edge-adjacent triangle. And since the new pillars are computed as the average of the normals of the two adjacent triangles with dihedral angle strictly less than 360° (Section **??** inset), each dot product is positive. The prism for a pink middle triangle intersects only the interior of the original triangle, and the dot product between the pillars and the triangle normal are positive by construction. □

A.3 Reduction to a Standard QP

With slack variable t, Problem **??** can be rewritten as

$$\max_t t;\ d_v n_f \geq t, \|d_v\| = 1, t \geq \epsilon.$$

In the admissible region, substituting $t = 1/s$,

$$\min_s s;\ s d_v n_f \geq 1, \|d_v\| = 1, 0 < s \leq 1/\epsilon.$$

Substituting $x = s d_v$ (thus $\|x\| = s\|d\| = s$), we obtain the QP problem **??**. Note that the lower bound on s is implied from $s d_v n_f \geq 1$, and the upper bound $\|x\| < 1/\epsilon$ is checked a posteriori.

A.4 Bijectivity of the Nonlinear Prismatic Transformation

In this section, we show that the isoparametric transformation of prismatic finite element [Ciarlet 1991] is also bijective if I1 holds.

We follow the notation from Appendix A.1.2. The map is defined as (for the top slab)

$$\begin{aligned} f(u,v,\eta) &= ue_1 + ve_2 + \eta n_0 + u\eta n_{01} + v\eta n_{02}, \\ \det J_f &= \langle (1-u-v)n_0 + un_1 + vn_2, e_1 + \eta n_{01}, e_2 + \eta n_{02} \rangle, \end{aligned}$$

where η is the variable corresponding to the thickness direction, and $n_{ij} = n_j - n_i$. Observe that $\det J_f(u,v,\eta_0)$ is linear in u,v for fixed η_0, the extrema will only be achieved at the corner points [Knabner and Summ 2001]. Therefore, it is sufficient to check the signs at three edges $(0,0,\eta)$, $(1,0,\eta)$, $(0,1,\eta)$ for $\eta \in [0,1]$.

$$\begin{aligned} &\det J_f(0,0,\eta) \\ &= \langle n_0, (1-\eta)e_1 + \eta(e_1 + n_1 - n_0), (1-\eta)e_2 + \eta(e_2 + n_2 - n_0) \rangle \\ &= (1-\eta)^2 \langle n_0, e_1, e_2 \rangle + (1-\eta)\eta \langle n_0, e_1, e_2 + n_2 \rangle \\ &+ (1-\eta)\eta \langle n_0, e_1 + n_1, e_2 \rangle + \eta^2 \langle n_0, e_1 + n_1, e_2 + n_2 \rangle \\ &= (1-\eta)^2 \mathrm{vol}_4 + (1-\eta)\eta(\mathrm{vol}_3 + \mathrm{vol}_1) + \eta^2 \mathrm{vol}_2 > 0. \end{aligned}$$

By symmetry, it is easy to verify that the following are positive,

$$\begin{aligned} \det J_f(1,0,\eta) &= (1-\eta)^2 \mathrm{vol}_6 + (1-\eta)\eta(\mathrm{vol}_7 + \mathrm{vol}_5) + \eta^2 \mathrm{vol}_8, \\ \det J_f(0,1,\eta) &= (1-\eta)^2 \mathrm{vol}_{11} + (1-\eta)\eta(\mathrm{vol}_9 + \mathrm{vol}_{10}) + \eta^2 \mathrm{vol}_{12}. \end{aligned}$$

A.5 Double Slab

In Section **??**, we explain that we want to ensure that our shell validity is independent from the enumeration of the faces; for this reason, we require that the conditions I1 and I2 are satisfied for all possible decompositions. Without using the double slab, different decompositions of the prism will result in different intersections with the middle surface. In other words, the projection operator will project the input mesh to a different set of triangles on the middle surface. The double slab makes the middle surface independent of the decomposition by construction. We note that Theorem **??** is valid only for double-slab prisms since the statement requires that one prism contains only one triangle.

A.6 Variants

(Disclaimer: this section is unpublished material. And serves as post publish notes for future references.)

A.6.1 Pointless Variant

We extend the framework, by allowing the relaxation of definition of a section. Originally, any intersection between the triangle and the (different decomposition of the) prism is counted. Here, we explicitly forgive when the intersection is only one point on the vertex of mid surface and the triangle. This predicate can be easily implemented, making use of the bitwise equal coordinates. Therefore, the initial tracking section only follows edge-adjacency relationship, instead of the vertex adjacent one in the main text. With such adjusted condition for section, the optimization can be trivially accomodated. The only thing that is different (simpler) is the part for bevel, since we have less triangles to consider. The existing one is still valid, however, we can get away with simpler red-green intersection

A.6.2 Dynamic Intersection Check

To make easier the guarantee of a self-intersection-free shell, substitue the AABB tree collision check with a dynamic hashgrid. We initialize two hashgrids at the beginning, one for top surface and another one for bottom surface. This is followed by additional shrinking to make sure they are clear. Each local opeartion is responsible to trial and keep the validity of the surfaces. Furthermore, the top/bottom surface should not interfere with the reference input surface, this can be accelerated through the maintainance of the tracking list, and is trial checked before a local operation is performed.

A.6.3 Feature Snapping Shell

We can separate the definiition of reference surface and input surface to make a feature snapping shell.

A.6.4 Positive Detereminant for Natural Prism Map

This section follows and extends Appendix A.4 In fact, we observe that, for a quadratic bezier curve piece $at^2 + bt(1-t) + c(1-t)^2, t \in [0,1]$, the sufficient and necessary condition is $a > 0, c > 0, b > -2\sqrt{ac}$, and the last relation is equivalent to (but more clearly stated as) $b > 0 || b^2 - 4ac < 0$. The current text (all 12 positive tetra) impose an unnatural constraint on edge splits: in the case of an agressively positive prism (all decompositions are all-positive), sub-prism may not be positive in the same sense. While this will not break anything, forbidding edge split will lead to unexpected lower quality in the optimization procedure.

B | Curve Meshing related

B.1 The geometric map is bijective

Consider a connected 3-dimensional compact manifold (curved) tetrahedral mesh $\mathcal{M} = \{\sigma_i\}, i = 1, \ldots, n$ with $\sigma_i = g_i(\hat{\tau})$, $\hat{\tau}$ a regular unit tetrahedron, $\det(J_{g_i}) > 0$ at all points including the boundary, and σ_i and σ_j agree on a shared face. Let $\mathcal{M}_D$ be the domain obtained by copies of $\hat{\tau}$ and identifying $\hat{\tau}_i$ along common faces. We then define the map

$$\sigma \colon M_D \to \mathbb{R}^3,$$

by setting $\sigma|_{\hat{\tau}_i} = \sigma_i$.

Proposition B.1. *Suppose $\sigma|_{\partial\mathcal{M}_D}$ is injective. Then σ is injective on the whole domain $\mathcal{M}_D$.*

Proof. Our argument closely follow from [Aigerman and Lipman 2013, Appendix B] and [Lipman 2014, Theorem 1].

We consider point $y \in \mathbb{R}^3$ in general position, but not in the faces, edges, vertices, or any plane spanned by a linear face of $\mathcal{M}$. For each tetrahedron τ, we construct the map $\hat{\Psi}$ as a composition of $\sigma|_{\partial\hat{\tau}_i}$ (restricted to the triangular faces of the regular tetrahedron) and the projection map χ to the unit sphere centered around y.

We parametrize the image of $\sigma|_{\partial\hat{\tau}_i}$ as $x(u, v)$ on the projective plane (note that since $\det J_{g_i} > 0$, the image is a non-degenerate surface homeomorphic to a sphere). Since $x(u, v)$, and its normal field $n(u, v) = \frac{\partial}{\partial u}x \times \frac{\partial}{\partial v}x$ are polynomial functions, Consider the parametric curve $n(u, v) \cdot (x(u, v) - y) = 0$, the algebraic curve partitions the surface into finite number of patches, and on each patch the orientation of $\hat{\Psi}$ is constant. We can further triangulate the partitions to create a (abstract simplicial) complex.

Similar to [Aigerman and Lipman 2013, Appendix B], we count the number of pre-images of y, which equals to the degree in general positions,

$$\deg(\sigma)(y) = \sum_{i=1}^{n} \deg(\hat{\Psi}|_{\tau_i}) = \deg(\hat{\Psi}|_{\partial\mathcal{M}}).$$

Since $\sigma|_{\partial\mathcal{M}_D}$ is injective, and furthermore

$$\deg(\hat{\Psi}|_{\partial\mathcal{M}}) = \deg\chi|_{\partial\mathcal{M}} = \begin{cases} 1, & y \in \mathcal{M} \\ 0, & y \notin \mathcal{M}. \end{cases}$$

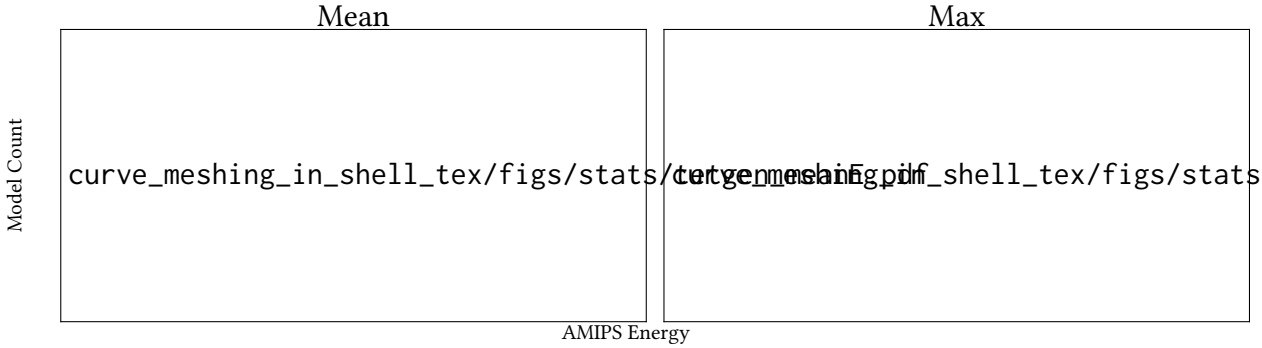

Figure B.1: Histogram of mean and maximum conformal AMIPS energy [Rabinovich et al. 2017] of the output of our method and TetGen.

Thus we have shown the map is injective for the general positions for y, and it remains to be shown that the map is an open map, which follows from the same argument of [Lipman 2014, Lemma 2].

□

B.2 Local Operations

Figure **??** introduce a set of *valid* local operations to modify the shell, including edge split, edge collapse, edge flip and vertex smoothing. The operations are an analogue of the triangle mesh edit operations [Dunyach et al. 2013], by simultaneously editing the shared connectivity of bottom, middle and top surface of the shell. **??**Theorem 3.7]chp:shell outlines invariant conditions, which maintains the shell projection to be bijective.

Our algorithm adopts and extends the local operations therein to the high order setting. In addition to the existing conditions, we also validate the curved volumetric mesh in the shell. The algorithm maintains the global intersection free bottom (top) surface with a dynamic hash grid [Teschner et al. 2003]. Then for each prism, we check the positivity (defined by the determinant of Jacobian of the geometric map) of the prismatic element (each decomposed into three tetrahedra).

In the presence of feature annotation and feature straightening (Section **??**) more care is taken to maintain the valid correspondence between the grouped feature chains and the curved edges: edge flip is disabled on the edges annotated as features; collapse is only allowed when it does not degenerate the chain, and the two endpoints of the edge is on the same chain. Since we require a map from the original input edges, we insert additional degree of freedoms in the input mesh. When performing edge split, the insertion of the new vertex is queried among the pre-image of the current edge, as an existing vertex of the input. For vertex smoothing (more specifically pan), the target location is limited to the set of input vertices.

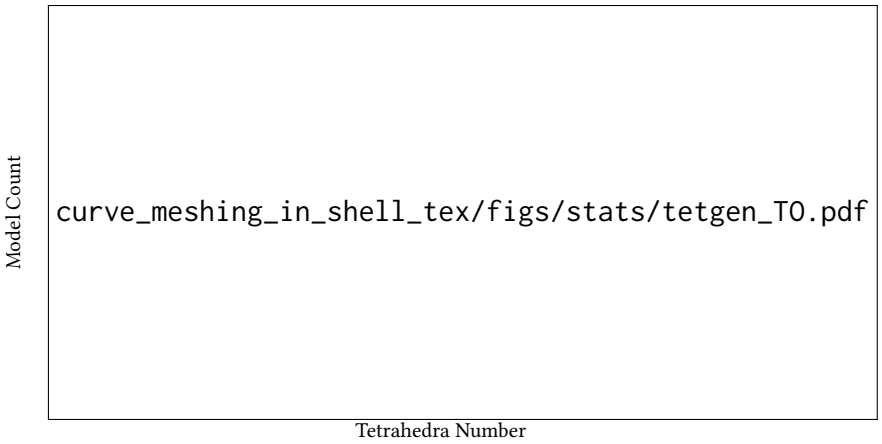

Figure B.2: Histogram of output tetrahedra number for TetGen and our method.

B.3 Boundary preserving TetGen comparison

We compared our conforming tetrahedral meshing algorithm (Section **??**) with TetGen on the linear shells (triangle meshes) of 3522 models from the Thingi10k dataset, giving each model sufficient computing resources (2 hours maximum running time and 32GB memory usage). Inheriting the robustness from TetWild, our method successfully processed all the inputs while preserving the triangulation, while TetGen fails on 224 models (215 models are not conforming, and 9 models have no output).

In Figure B.1, we show the average and maximum element quality of the output of our method and TetGen. Our method has a better average and maximum output quality than TetGen. Note that in the quality plot, the "tail" of TetGen's distribution is longer than ours. The maximum average energy of TetGen's output and ours are 3×10^8 and 3×10^5 respectively. The largest maximum energy of TetGen's output and ours are 3×10^{12} and 7×10^6 respectively. Our method generates denser output (Figure B.2), but our focus is on robustness instead of efficiency in this step.

Bibliography

A. Freitag, L. and Ollivier-Gooch, C. (1998). Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40.

Abgrall, R., Dobrzynski, C., and Froehly, A. (2012). A method for computing curved 2D and 3D meshes via the linear elasticity analogy: preliminary results. Research Report RR-8061, INRIA.

Abgrall, R., Dobrzynski, C., and Froehly, A. (2014). A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems. *International Journal for Numerical Methods in Fluids*, 76(4):246–266.

Aigerman, N., Kovalsky, S. Z., and Lipman, Y. (2017). Spherical orbifold tutte embeddings. *ACM Trans. Graph.*, 36(4).

Aigerman, N. and Lipman, Y. (2013). Injective and bounded distortion mappings in 3d. *ACM Transactions on Graphics (TOG)*, 32(4):1–14.

Aigerman, N. and Lipman, Y. (2015). Orbifold tutte embeddings. *ACM Trans. Graph.*, 34(6).

Aigerman, N. and Lipman, Y. (2016). Hyperbolic orbifold tutte embeddings. *ACM Trans. Graph.*, 35(6).

Aigerman, N., Poranne, R., and Lipman, Y. (2014). Lifted bijections for low distortion surface mappings. *ACM Trans. Graph.*, 33(4).

Aigerman, N., Poranne, R., and Lipman, Y. (2015). Seamless surface mappings. *ACM Trans. Graph.*, 34(4).

Ainsley, S., Vouga, E., Grinspun, E., and Tamstorf, R. (2012). Speculative parallel asynchronous contact mechanics. *ACM Trans. Graph.*, 31(6):151:1–151:8.

Alauzet, F. and Marcum, D. (2014). A closed advancing-layer method with changing topology mesh movement for viscous mesh generation. In *Proceedings of the 22nd International Meshing Roundtable*, pages 241–261. Springer International Publishing, Cham.

Alexa, M. (2019). Harmonic triangulations. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 38(4):54.

Alexa, M. (2020). Conforming weighted delaunay triangulations. *ACM Transactions on Graphics (TOG)*, pages 1–16.

Alliez, P., Cohen-Steiner, D., Yvinec, M., and Desbrun, M. (2005). Variational tetrahedral meshing. In *ACM Transactions on Graphics (TOG)*. ACM.

Aluru, S. and Sevilgen, F. E. (1997). Parallel domain decomposition and load balancing using space-filling curves. In *Proceedings fourth international conference on high-performance computing*, pages 230–235. IEEE.

Ando, R., Thürey, N., and Wojtan, C. (2013). Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. (Proc. SIGGRAPH 2013)*.

Aubry, R., Dey, S., Mestreau, E., and Karamete, B. (2017). Boundary layer mesh generation on arbitrary geometries. *International Journal for Numerical Methods in Engineering*, 112(2):157–173.

Aubry, R., Mestreau, E., Dey, S., Karamete, B., and Gayman, D. (2015). On the 'most normal'normal—part 2. *Finite Elements in Analysis and Design*, 97:54–63.

Babuska, I. and Guo, B. Q. (1988). The h-p version of the finite element method for domains with curved boundaries. *SIAM Journal on Numerical Analysis*, 25(4):837–861.

Babuška, I. and Guo, B. (1992). The h, p and h-p version of the finite element method; basis theory and applications. *Advances in Engineering Software*, 15(3):159 – 174.

Bajaj, C. L., Xu, G., Holt, R. J., and Netravali, A. N. (2002). Hierarchical multiresolution reconstruction of shell surfaces. *Computer Aided Geometric Design*, 19(2):89–112.

Bargteil, A. W. and Cohen, E. (2014). Animation of deformable bodies with quadratic bézier finite elements. *ACM Transactions on Graphics (TOG)*, 33(3):27.

Bargteil, A. W., Wojtan, C., Hodgins, J. K., and Turk, G. (2007). A finite element method for animating large viscoplastic flow. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 16–es, New York, NY, USA. Association for Computing Machinery.

Barnhill, R. E., Opitz, K., and Pottmann, H. (1992). Fat surfaces: a trivariate approach to triangle-based interpolation on surfaces. *Computer Aided Geometric Design*, 9(5):365–378.

Bassi, F. and Rebay, S. (1997). High-order accurate discontinuous finite element solution of the 2d euler equations. *Journal of Computational Physics*, 138(2):251 – 285.

Batista, V. H., Millman, D. L., Pion, S., and Singler, J. (2010). Parallel geometric algorithms for multi-core computers. *Computational Geometry*, 43(8):663–677.

Baumgart, B. G. (1972). Winged edge polyhedron representation. Technical report, Stanford, CA, USA.

Bender, J. and Deul, C. (2013). Adaptive cloth simulation using corotational finite elements. *Computers & Graphics*, 37(7):820–829.

Bern, M., Eppstein, D., and Gilbert, J. (1994). Provably good mesh generation. *Journal of Computer and System Sciences*, 48(3):384 – 409.

Bishop, C. J. (2016). Nonobtuse triangulations of pslgs. *Discrete & Computational Geometry*, 56(1):43–92.

Blandford, D. K., Blelloch, G. E., and Kadow, C. (2006). Engineering a compact parallel delaunay algorithm in 3d. In *Proceedings of the twenty-second Annual Symposium on Computational Geometry*, pages 292–300.

Blelloch, G. E., Miller, G. L., Hardwick, J. C., and Talmor, D. (1999). Design and implementation of a practical parallel delaunay algorithm. *Algorithmica*, 24(3):243–269.

Boissonnat, J.-D., Devillers, O., Pion, S., Teillaud, M., and Yvinec, M. (2002). Triangulations in cgal. *Computational Geometry*, 22:5–19.

Boissonnat, J.-D. and Oudot, S. (2005). Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451.

Bommes, D., Lévy, B., Pietroni, N., Puppo, E., a, C. S., Tarini, M., and Zorin, D. (2012). State of the art in quad meshing. In *Eurographics STARS*.

Borrell, R., Cajas, J. C., Mira, D., Taha, A., Koric, S., Vázquez, M., and Houzeaux, G. (2018). Parallel mesh partitioning based on space filling curves. *Computers & Fluids*, 173:264–272.

Botsch, M. and Kobbelt, L. (2003). Multiresolution surface representation based on displacement volumes. In *Computer Graphics Forum*, volume 22, pages 483–491. Wiley Online Library.

Botsch, M., Pauly, M., Gross, M. H., and Kobbelt, L. (2006). Primo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing*, number CONF, pages 11–20.

Boubekeur, T. and Alexa, M. (2008). Phong tessellation. In *ACM Transactions on Graphics (TOG)*, volume 27, page 141. ACM.

Bridson, R. and Doran, C. (2014). Quartet: A tetrahedral mesh generator that does isosurface stuffing with an acute tetrahedral tile. `https://github.com/crawforddoran/quartet`.

Brisson, E. (1989). Representing geometric structures in d dimensions: Topology and order. In *Proceedings of the Fifth Annual Symposium on Computational Geometry*, SCG '89, page 218–227, New York, NY, USA. Association for Computing Machinery.

Brochu, T., Edwards, E., and Bridson, R. (2012). Efficient geometrically exact continuous collision detection. *ACM Trans. Graph.*, 31(4).

Bronson, J. R., Levine, J. A., and Whitaker, R. T. (2013). Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *Proceedings of the 21st International Meshing Roundtable*, pages 191–209. Springer Berlin Heidelberg, Berlin, Heidelberg.

Busaryev, O., Dey, T. K., and Levine, J. A. (2009). Repairing and meshing imperfect shapes with delaunay refinement. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, SPM '09, pages 25–33. ACM.

Busaryev, O., Dey, T. K., and Wang, H. (2013). Adaptive fracture simulation of multi-layered thin plates. *ACM Trans. Graph.*, 32(4).

Calderon, S. and Boubekeur, T. (2017). Bounding proxies for shape approximation. *ACM Trans. Graph.*, 36(4).

Campen, M., Capouellez, R., Shen, H., Zhu, L., Panozzo, D., and Zorin, D. (2021). Efficient and robust discrete conformal equivalence with boundary. *ACM Trans. Graph.*, 40(6).

Campen, M., Shen, H., Zhou, J., and Zorin, D. (2019). Seamless parametrization with arbitrary cones for arbitrary genus. *ACM Trans. Graph.*, 39(1).

Campen, M., Silva, C. T., and Zorin, D. (2016). Bijective maps from simplicial foliations. *ACM Trans. Graph.*, 35(4):74:1–74:15.

Campen, M. and Zorin, D. (2017). Similarity maps and field-guided t-splines: a perfect couple. *ACM Trans. Graph.*, 36(4).

Canann, S. A., Muthukrishnan, S. N., and Phillips, R. K. (1996). Topological refinement procedures for triangular finite element meshes. *Engineering with Computers*, 12(3):243–255.

Canann, S. A., Stephenson, M. B., and Blacker, T. (1993). Optismoothing: An optimization-driven approach to mesh smoothing. *Finite Elements in Analysis and Design*, 13(2):185 – 190.

Cardoze, D., Cunha, A., Miller, G. L., Phillips, T., and Walkington, N. (2004). A bézier-based approach to unstructured moving meshes. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 310–319, New York, NY, USA. ACM.

Chazal, F. and Cohen-Steiner, D. (2005). A condition for isotopic approximation. *Graphical Models*, 67(5):390–404.

Chazal, F., Lieutier, A., Rossignac, J., and Whited, B. (2010). Ball-map: Homeomorphism between compatible surfaces. *International Journal of Computational Geometry & Applications*, 20(03):285–306.

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

Chen, M.-B. (2010). The merge phase of parallel divide-and-conquer scheme for 3d delaunay triangulation. In *International Symposium on Parallel and Distributed Processing with Applications*, pages 224–230. IEEE.

Chen, Y., Tong, X., Wang, J., Wang, J., Lin, S., Guo, B., Shum, H.-Y., and Shum, H.-Y. (2004). Shell texture functions. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 343–353. ACM.

Cheng, S.-W., Dey, T. K., Edelsbrunner, H., Facello, M. A., and Teng, S.-H. (2000). Silver exudation. *Journal of the ACM (JACM)*, 47(5):883–904.

Cheng, S.-W., Dey, T. K., and Levine, J. A. (2008). A practical delaunay meshing algorithm for a large class of domains. In *Proceedings of the 16th International Meshing Roundtable*, pages 477–494. Springer.

Cheng, S.-W., Dey, T. K., and Shewchuk, J. (2012). *Delaunay Mesh Generation*. Chapman and Hall/CRC, Boca Raton, Florida.

Cheng, X.-X., Fu, X.-M., Zhang, C., and Chai, S. (2019). Practical error-bounded remeshing by adaptive refinement. *Computers & Graphics*, 82:163 – 173.

Chentanez, N., Feldman, B. E., Labelle, F., O'Brien, J. F., and Shewchuk, J. R. (2007). Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, page 219–228, Goslar, DEU. Eurographics Association.

Chew, L. P. (1989). Constrained delaunay triangulations. *Algorithmica*, 4(1-4):97–108.

Chew, L. P. (1993). Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 274–280. ACM.

Chew, L. P., Chrisochoides, N., and Sukup, F. (1997). Parallel constrained delaunay meshing. *ASME APPLIED MECHANICS DIVISION-PUBLICATIONS-AMD*, 220:89–96.

Chrisochoides, N. (2006). Parallel mesh generation. In *Numerical solution of partial differential equations on parallel computers*, pages 237–264. Springer.

Chrisochoides, N. and Nave, D. (2003). Parallel delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering*, 58(2):161–176.

Ciarlet, P. G. (1991). Basic error estimates for elliptic problems. *Finite Element Methods (Part 1)*.

Cignoni, P., Montani, C., Perego, R., and Scopigno, R. (1993). Parallel 3d delaunay triangulation. *Computer Graphics Forum*, 12(3):129–142.

Cohen, J., Manocha, D., and Olano, M. (1997). Simplifying polygonal models using successive mappings. In *Proceedings. Visualization'97 (Cat. No. 97CB36155)*, pages 395–402. IEEE.

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., and Wright, W. (1996). Simplification envelopes. In *Siggraph*, volume 96, pages 119–128.

Cohen-Steiner, D., de Verdière, E. C., and Yvinec, M. (2002). Conforming delaunay triangulations in 3D. In *Proceedings of the eighteenth annual symposium on Computational geometry - SCG '02*, number 2, pages 217 – 233. ACM Press.

Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam.

Coreform (2020). Cubit. `https://coreform.com/products/coreform-cubit/`.

Cuilliere, J.-C., Francois, V., and Drouet, J.-M. (2013). Automatic 3D mesh generation of multiple domains for topology optimization methods. In *Proceedings of the 21st International Meshing Roundtable*, pages 243–259. Springer Berlin Heidelberg, Berlin, Heidelberg.

Degener, P., Meseth, J., and Klein, R. (2003). An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable*, pages 201–213.

Dey, S., O'bara, R. M., and Shephard, M. S. (1999). Curvilinear mesh generation in 3d. In *IMR*, pages 407–417. IMR.

Dey, T. K. and Levine, J. A. (2008). Delpsc: A delaunay mesher for piecewise smooth complexes. In *Proceedings of the twenty-fourth annual symposium on Computational geometry - SCG '08*, pages 220–221, New York, NY, USA. ACM Press.

DiCarlo, A., Paoluzzi, A., and Shapiro, V. (2014). Linear algebraic representation for topological structures. *Computer-Aided Design*, 46:269–274. 2013 SIAM Conference on Geometric and Physical Modeling.

Dobrev, V., Knupp, P., Kolev, T., Mittal, K., and Tomov, V. (2019). The target-matrix optimization paradigm for high-order meshes. *SIAM Journal on Scientific Computing*, 41(1):B50–B68.

Dobrzynski, C. and El Jannoun, G. (2017). High order mesh untangling for complex curved geometries. Research Report RR-9120, INRIA Bordeaux, équipe CARDAMOM.

Doran, C., Chang, A., and Bridson, R. (2013). Isosurface stuffing improved: Acute lattices and feature matching. In *ACM SIGGRAPH 2013 Talks on - SIGGRAPH '13*, pages 38:1–38:1, New York, NY, USA. ACM Press.

Du, Q. and Wang, D. (2003). Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International journal for numerical methods in engineering*, 56(9):1355–1373.

Dufourd, J.-F. (1991). An obj3 functional specification for boundary representation. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 61–72.

Dunyach, M., Vanderhaeghe, D., Barthe, L., and Botsch, M. (2013). Adaptive remeshing for real-time mesh deformation.

Dyer, R., Zhang, H., and Möller, T. (2007). Delaunay mesh construction.

Ebke, H.-C., Campen, M., Bommes, D., and Kobbelt, L. (2014). Level-of-detail quad meshing. *ACM Transactions on Graphics (TOG)*, 33(6):184.

Engvall, L. and Evans, J. A. (2016). Isogeometric triangular bernstein–bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 304:378 – 407.

Engvall, L. and Evans, J. A. (2017). Isogeometric unstructured tetrahedral and mixed-element bernstein–bézier discretizations. *Computer Methods in Applied Mechanics and Engineering*, 319:83 – 123.

Engvall, L. and Evans, J. A. (2018). Mesh quality metrics for isogeometric bernstein-bézier discretizations. *arXiv:1810.06975*.

Engwirda, D. (2016). Conforming restricted delaunay mesh generation for piecewise smooth complexes. *CoRR*.

Ezuz, D., Solomon, J., and Ben-Chen, M. (2019). Reversible harmonic maps between discrete surfaces. *ACM Trans. Graph.*, 38(2).

Feng, L., Alliez, P., Busé, L., Delingette, H., and Desbrun, M. (2018). Curved optimal delaunay triangulation. *ACM Transactions on Graphics (TOG)*.

Floater, M. (2003). One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, 72(242):685–696.

Floater, M. S. (1997). Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250.

Floater, M. S. and Hormann, K. (2005). Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization*, pages 157–186. Springer Verlag.

Fortunato, M. and Persson, P.-O. (2016). High-order unstructured curved mesh generation using the winslow equations. *Journal of Computational Physics*, 307:1 – 14.

Foteinos, P. and Chrisochoides, N. (2011). Dynamic parallel 3d delaunay triangulation. In *Proceedings of the 20th International Meshing Roundtable*, pages 3–20. Springer.

Fu, X.-M. and Liu, Y. (2016). Computing inversion-free mappings by simplex assembly. *ACM Trans. Graph.*, 35(6):216:1–216:12.

Fu, X.-M., Liu, Y., and Guo, B. (2015). Computing locally injective mappings by advanced mips. *ACM Transactions on Graphics (TOG)*, 34(4):71.

Funke, D. and Sanders, P. (2017). Parallel d-d delaunay triangulations in shared and distributed memory. In *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 207–217. SIAM.

Galtier, J. and George, P. L. (1996). Prepartitioning as a way to mesh subdomains in parallel. In *in 5th International Meshing Roundtable*. Citeseer.

Gao, X., Shen, H., and Panozzo, D. (2019). Feature preserving octree-based hexahedral meshing. *Computer Graphics Forum*, 38(5):135–149.

Gargallo-Peiró, A., Roca, X., Peraire, J., and Sarrate, J. (2015). Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 103(5):342–363.

Gargallo Peiró, A., Roca Navarro, F. J., Peraire Guitart, J., and Sarrate Ramos, J. (2013). High-order mesh generation on cad geometries. In *Adaptive Modeling and Simulation 2013*, pages 301–312. Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE).

Garimella, R. V. and Shephard, M. S. (2000). Boundary layer mesh generation for viscous flow simulations. *International Journal for Numerical Methods in Engineering*, 49(1-2):193–218.

Garland, M. and Heckbert, P. (1999). *Quadric-Based Polygonal Surface Simplification*. PhD thesis, USA. AAI9950005.

Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co.

George, P. and Borouchaki, H. (2012). Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 90(9):1156–1182.

Geuzaine, C., Johnen, A., Lambrechts, J., Remacle, J.-F., and Toulorge, T. (2015). *The Generation of Valid Curvilinear Meshes*, pages 15–39. Springer International Publishing, Cham.

Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331.

Ghasemi, A., Taylor, L. K., and Newman, III, J. C. (2016). Massively parallel curved spectral/finite element mesh generation of industrial cad geometries in two and three dimensions. *Fluids Engineering Division Summer Meeting*.

Gillespie, M., Springborn, B., and Crane, K. (2021). Discrete conformal equivalence of polyhedral surfaces. *ACM Trans. Graph.*, 40(4).

Gotsman, C. and Surazhsky, V. (2001). Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75.

Grinspun, E., Krysl, P., and Schröder, P. (2002). Charms: A simple framework for adaptive simulation. *ACM Trans. Graph.*, 21(3):281–290.

Gu, X., Guo, R., Luo, F., Sun, J., and Wu, T. (2018a). A discrete uniformization theorem for polyhedral surfaces ii. *Journal of Differential Geometry*, 109(3):431–466.

Gu, X., Luo, F., Sun, J., and Wu, T. (2018b). A discrete uniformization theorem for polyhedral surfaces. *Journal of Differential Geometry*, 109(2):223–256.

Guéziec, A. (1996). *Surface simplification inside a tolerance volume*. IBM TJ Watson Research Center.

Guibas, L. and Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2):74–123.

Gumhold, S., Borodin, P., and Klein, R. (2003). Intersection free simplification. *International Journal of Shape Modeling*, 9(02):155–176.

Guo, H.-X., Liu, X., Yan, D.-M., and Yang, L. (2020). Cut-enhanced polycube-maps for feature-aware all-hex meshing. *ACM Transactions on Graphics (TOG)*, 39(4):106:1–106:14.

Hahmann, S. and Bonneau, G. . (2003). Polynomial surfaces interpolating arbitrary triangulations. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):99–109.

Haimes, R. (2014). Moss: Multiple orthogonal strand system. In *Proceedings of the 22nd International Meshing Roundtable*, pages 75–91. Springer International Publishing, Cham.

Harmon, D. (2010). *Robust, efficient, and accurate contact algorithms*. PhD thesis, Columbia University.

Harmon, D., Panozzo, D., Sorkine, O., and Zorin, D. (2011). Interference-aware geometric modeling. *ACM Trans. Graph.*, 30(6):137:1–137:10.

Harmon, D., Vouga, E., Smith, B., Tamstorf, R., and Grinspun, E. (2009). Asynchronous contact mechanics. *ACM Trans. Graph.*, 28(3):87:1–87:12.

Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W. (1994). Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302.

Hormann, K. and Greiner, G. (2000). Mips: An efficient global parametrization method. Technical report, ERLANGEN-NUERNBERG UNIV (GERMANY) COMPUTER GRAPHICS GROUP.

Hormann, K., Lévy, B., and Sheffer, A. (2007). Mesh parameterization: Theory and practice. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA. ACM.

Hu, K., Yan, D., Bommes, D., Alliez, P., and Benes, B. (2017). Error-bounded and feature preserving surface remeshing with minimal angle improvement. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2560–2573.

Hu, K., Yan, D.-M., Bommes, D., Alliez, P., and Benes, B. (2016). Error-bounded and feature preserving surface remeshing with minimal angle improvement. *IEEE transactions on visualization and computer graphics*, 23(12):2560–2573.

Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F. (2019a). Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):1–16.

Hu, Y., Schneider, T., Gao, X., Zhou, Q., Jacobson, A., Zorin, D., and Panozzo, D. (2019b). Triwild: robust triangulation with curve constraints. *ACM Transactions on Graphics (TOG)*, 38(4):52.

Hu, Y., Schneider, T., Wang, B., Zorin, D., and Panozzo, D. (2019c). Fast tetrahedral meshing in the wild. *arXiv preprint arXiv:1908.03581*.

Hu, Y., Schneider, T., Wang, B., Zorin, D., and Panozzo, D. (2020). Fast tetrahedral meshing in the wild. *ACM Trans. Graph.*, 39(4).

Hu, Y., Zhou, Q., Gao, X., Jacobson, A., Zorin, D., and Panozzo, D. (2018). Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4):60–1.

Huang, J., Liu, X., Jiang, H., Wang, Q., and Bao, H. (2007). Gradient-based shell generation and deformation. *Computer Animation and Virtual Worlds*, 18(4-5):301–309.

Hutchinson, D., Preston, M., and Hewitt, T. (1996). Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, page 31–45, Berlin, Heidelberg. Springer-Verlag.

Jacobson, A., Kavan, L., and Sorkine-Hornung, O. (2013). Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.*, 32(4).

Jameson, A., Alonso, J., and McMullen, M. (2002). Application of a non-linear frequency domain solver to the euler and navier-stokes equations. In *40th AIAA Aerospace Sciences Meeting & Exhibit*.

Jamin, C., Alliez, P., Yvinec, M., and Boissonnat, J.-D. (2015). Cgalmesh: a generic framework for delaunay mesh generation. *ACM Transactions on Mathematical Software (TOMS)*, 41(4):23.

Jiang, Z. (2017). GitHub - jiangzhongshi/Scaffold-Map: Robust, efficient and low distortion bijective mapping in 2D and 3D — github.com. `https://github.com/jiangzhongshi/Scaffold-Map`.

Jiang, Z. (2021a). GitHub - jiangzhongshi/bichon: Robust Coarse Curved TetMesh Generation — github.com. `https://github.com/jiangzhongshi/bichon`. [Accessed 09-Jun-2022].

Jiang, Z. (2021b). GitHub - jiangzhongshi/bijective-projection-shell: A projection-based bijective map definition and a robust algorithm to construct the prismatic shell domain. https://cs.nyu.edu/ zhongshi/files/BijectivePrism.pdf — github.com. `https://github.com/jiangzhongshi/bijective-projection-shell`.

Jiang, Z., Schaefer, S., and Panozzo, D. (2017). Simplicial complex augmentation framework for bijective maps. *ACM Transactions on Graphics*, 36(6).

Jiao, X. and Wang, D. (2012). Reconstructing high-order surfaces for meshing. *Engineering with Computers*, 28(4):361–373.

Jimenez, P., Thomas, F., and Torras, C. (2001). 3d collision detection: a survey. *Computers & Graphics*, 25(2):269 – 285.

Jin, M., Kim, J., Luo, F., and Gu, X. (2008). Discrete surface ricci flow. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1030–1043.

Jin, Y., Song, D., Wang, T., Huang, J., Song, Y., and He, L. (2019). A shell space constrained approach for curve design on surface meshes. *Computer-Aided Design*, 113:24–34.

Johnen, A., Remacle, J.-F., and Geuzaine, C. (2013). Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 233:359–372.

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.

Kharevych, L., Springborn, B., and Schröder, P. (2006). Discrete conformal mappings via circle patterns. *ACM Trans. Graph.*, 25(2):412–438.

Kjolstad, F., Chou, S., Lugato, D., Kamil, S., and Amarasinghe, S. (2017). Taco: A tool to generate tensor algebra kernels. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 943–948. IEEE.

Kjolstad, F., Kamil, S., Ragan-Kelley, J., Levin, D. I., Sueda, S., Chen, D., Vouga, E., Kaufman, D. M., Kanwar, G., Matusik, W., et al. (2016). Simit: A language for physical simulation. *ACM Transactions on Graphics (TOG)*, 35(2):1–21.

Klingner, B. and Shewchuk, J. (2007). Aggressive tetrahedral mesh improvement. pages 3–23.

Klingner, B. M., Feldman, B. E., Chentanez, N., and O'Brien, J. F. (2006). Fluid animation with dynamic meshes. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, page 820–825, New York, NY, USA. Association for Computing Machinery.

Knabner, P. and Summ, G. (2001). The invertibility of the isoparametric mapping for pyramidal and prismatic finite elements. *Numerische Mathematik*, 88(4):661–681.

Knupp, P. M. (2000). Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i, a framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering*, 48(3):401–420.

Knupp, P. M. (2002). Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii, a framework for volume mesh optimization and the condition number of the jacobian matrix. *International Journal for Numerical Methods in Engineering*, 48(8):1165–1185.

Kobbelt, L., Campagna, S., Vorsatz, J., and Seidel, H.-P. (1998). Interactive multi-resolution modeling on arbitrary meshes. In *Siggraph*, volume 98, pages 105–114.

Kovalsky, S. Z., Galun, M., and Lipman, Y. (2016). Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.*, 35(4):134:1–134:11.

Kraemer, P., Untereiner, L., Jund, T., Thery, S., and Cazier, D. (2014). Cgogn: N-dimensional meshes with combinatorial maps. In *Proceedings of the 22nd International Meshing Roundtable*, pages 485–503. Springer.

Kraevoy, V. and Sheffer, A. (2004). Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics (TOG)*, 23(3):861–869.

Krishnamurthy, V. and Levoy, M. (1996). Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 313–324.

Labelle, F. and Shewchuk, J. R. (2007). Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. In *ACM SIGGRAPH 2007 papers on - SIGGRAPH '07*, page 57, New York, NY, USA. ACM Press.

Lachat, C., Dobrzynski, C., and Pellegrini, F. (2014). Parallel mesh adaptation using parallel graph partitioning. In *5th European conference on computational mechanics (ECCM V)*, volume 3, pages 2612–2623. CIMNE-International Center for Numerical Methods in Engineering.

Lee, A., Moreton, H., and Hoppe, H. (2000). Displaced subdivision surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94.

Lengyel, J., Praun, E., Finkelstein, A., and Hoppe, H. (2001). Real-time fur over arbitrary surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 227–232. ACM.

Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371.

Li, L. and Volkov, V. (2005). Cloth animation with adaptively refined meshes. In *Proceedings of the Twenty-Eighth Australasian Conference on Computer Science - Volume 38*, ACSC '05, page 107–113, AUS. Australian Computer Society, Inc.

Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C., and Kaufman, D. M. (2020). Incremental potential contact: Intersection- and inversion-free large deformation dynamics. *ACM Trans. Graph. (SIGGRAPH)*, 39(4).

Li, Y., Liu, Y., Xu, W., Wang, W., and Guo, B. (2012). All-hex meshing using singularity-restricted field. *ACM Trans. Graph.*, 31(6).

Lienhardt, P. (1994). N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry & Applications*, 4(03):275–324.

Lin, H., Chen, W., and Bao, H. (2007). Adaptive patch-based mesh fitting for reverse engineering. *Computer-Aided Design*, 39(12):1134 – 1142.

Linardakis, L. and Chrisochoides, N. (2006). Delaunay decoupling method for parallel guaranteed quality planar mesh refinement. *SIAM Journal on Scientific Computing*, 27(4):1394–1423.

Lipman, Y. (2012). Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.*, 31(4):108:1–108:13.

Lipman, Y. (2014). Bijective mappings of meshes with boundary and the degree in mesh processing. *SIAM Journal on Imaging Sciences*, 7(2):1263–1283.

Litke, N., Droske, M., Rumpf, M., and Schröder, P. (2005). An image processing approach to surface matching. In *Symposium on Geometry Processing*, volume 255.

Liu, Y.-J., Xu, C.-X., Fan, D., and He, Y. (2015). Efficient construction and simplification of delaunay meshes. *ACM Transactions on Graphics (TOG)*, 34(6).

Lo, S. (2012). Parallel delaunay triangulation in three dimensions. *Computer Methods in Applied Mechanics and Engineering*, 237:88–106.

Loseille, A., Alauzet, F., and Menier, V. (2017). Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes. *Computer-Aided Design*, 85:53–67.

Lu, Q., Shephard, M. S., Tendulkar, S., and Beall, M. W. (2013). Parallel curved mesh adaptation for large scale high-order finite element simulations. In Jiao, X. and Weill, J.-C., editors, *Proceedings of the 21st International Meshing Roundtable*, pages 419–436, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lu, Q., Shephard, M. S., Tendulkar, S., and Beall, M. W. (2014). Parallel mesh adaptation for high-order finite element methods with curved element geometry. *Engineering with Computers*, 30(2):271–286.

Luo, F. (2004). Combinatorial yamabe flow on surfaces. *Communications in Contemporary Mathematics*, 6(05):765–780.

Luo, X., Shephard, M. S., Lee, L.-Q., Ng, C., and Ge, L. (2008). Tracking adaptive moving mesh refinements in 3d curved domains for large-scale higher order finite element simulations. In *Proceedings of the 17th International Meshing Roundtable*, pages 585–601. Springer.

Luo, X., Shephard, M. S., and Remacle, J.-F. (2001). The influence of geometric approximation on the accuracy of high order methods. *Rensselaer SCOREC report*, 1.

Luo, X., Shephard, M. S., Remacle, J.-F., O'Bara, R. M., Beall, M. W., Szabó, B. A., and Actis, R. (2002). p-version mesh generation issues. In *IMR*.

Mahmoud, A. H., Porumbescu, S. D., and Owens, J. D. (2021). Rxmesh: A gpu mesh data structure. *ACM Trans. Graph.*, 40(4).

Mandad, M., Cohen-Steiner, D., and Alliez, P. (2015). Isotopic approximation within a tolerance volume. *ACM Transactions on Graphics (TOG)*, 34(4):64.

Mäntylä, M. (1987). *An introduction to solid modeling*. Computer Science Press, Inc.

Marcon, J., Peiró, J., Moxey, D., Bergemann, N., Bucklow, H., and Gammon, M. R. (2019). A semi-structured approach to curvilinear mesh generation around streamlined bodies. In *AIAA Scitech 2019 Forum*, page 1725.

Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V. G., and Lipman, Y. (2017). Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1.

Marot, C., Pellerin, J., and Remacle, J.-F. (2019). One machine, one minute, three billion tetrahedra. *International Journal for Numerical Methods in Engineering*, 117(9):967–990.

Marot, C. and Remacle, J.-F. (2020). Quality tetrahedral mesh generation with hxt. *arXiv preprint arXiv:2008.08508*.

Marschner, Z., Palmer, D., Zhang, P., and Solomon, J. (2020). Hexahedral mesh repair via sum-of-squares relaxation. In *Computer Graphics Forum*, pages 133–147. Wiley Online Library.

Mezger, J., Thomaszewski, B., Pabst, S., and Straßer, W. (2007). A finite element method for interactive physically based shape modelling with quadratic tetrahedra.

Mezger, J., Thomaszewski, B., Pabst, S., and Straśer, W. (2009). Interactive physically-based shape editing. *Computer Aided Geometric Design*, 26(6):680 – 694. Solid and Physical Modeling 2008.

Misztal, M. K. and Bærentzen, J. A. (2012). Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. Graph.*, 31(3).

Molino, N., Bridson, R., and Fedkiw, R. (2003). Tetrahedral mesh generation for deformable bodies. In *Proc. Symposium on Computer Animation*.

Moxey, D., Ekelschot, D., Keskin, U., Sherwin, S., and Peiro, J. (2016). High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design*, 72:130 – 139. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

Moxey, D., Green, M., Sherwin, S., and Peiró, J. (2015). An isoparametric approach to high-order curvilinear boundary-layer meshing. *Computer Methods in Applied Mechanics and Engineering*, 283:636 – 650.

Moxey, D., Turner, M., Marcon, J., and Peiro, J. (2018). Nekmesh: An open-source high-order mesh generator.

Müller, M., Chentanez, N., Kim, T.-Y., and Macklin, M. (2015). Air meshes for robust collision handling. *ACM Trans. Graph.*, 34(4).

Murphy, M., Mount, D. M., and Gable, C. W. (2001). A point-placement strategy for conforming delaunay tetrahedralization. *International Journal of Computational Geometry & Applications*, 11(06):669–682.

Narain, R., Pfaff, T., and O'Brien, J. F. (2013). Folding and crumpling adaptive sheets. *ACM Trans. Graph.*, 32(4).

Narain, R., Samii, A., and O'Brien, J. F. (2012). Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6).

Nguyen, H. (2007). *Gpu gems 3*. Addison-Wesley Professional.

Oden, J. (1994). Optimal h-p finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 112(1):309 – 331.

Okusanya, T. and Peraire, J. (1996). Parallel unstructured mesh generation.

Okusanya, T. and Peraire, J. (1997). 3-d parallel unstructured mesh generation. In *Proc. Joint ASME/ASCE/SES Summer Meeting*. Citeseer.

Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., and Guibas, L. (2012). Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4).

Ovsjanikov, M., Corman, E., Bronstein, M., Rodolà, E., Ben-Chen, M., Guibas, L., Chazal, F., and Bronstein, A. (2017). Computing and processing correspondences with functional maps. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17.

Palmer, D., Bommes, D., and Solomon, J. (2020). Algebraic representations for volumetric frame fields. *ACM Trans. Graph.*, 39(2).

Panozzo, D., Baran, I., Diamanti, O., and Sorkine-Hornung, O. (2013). Weighted averages on surfaces. *ACM Transactions on Graphics (TOG)*, 32(4):60.

Peiró, A. G., Gironés, E. R., Navarro, F. J., and Ramos, J. S. (2015). On curving high-order hexahedral meshes.

Peiró, A. G., Roca, X., Peraire, J., and Sarrate, J. (2014). Defining quality measures for validation and generation of high-order tetrahedral meshes. In Sarrate, J. and Staten, M., editors, *Proceedings of the 22nd International Meshing Roundtable*, pages 109–126, Cham. Springer International Publishing.

Peiró, J., Sherwin, S. J., and Giordana, S. (2008). Automatic reconstruction of a patient-specific high-order surface representation and its application to mesh generation for cfd calculations. *Medical & Biological Engineering & Computing*, 46(11):1069–1083.

Peng, J., Kristjansson, D., and Zorin, D. (2004). Interactive modeling of topologically complex geometric detail. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 635–643. ACM.

Persson, P.-O. and Peraire, J. (2009). Curved mesh generation and mesh refinement using lagrangian solid mechanics. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*.

Pfaff, T., Narain, R., de Joya, J. M., and O'Brien, J. F. (2014). Adaptive tearing and cracking of thin sheets. *ACM Trans. Graph.*, 33(4).

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317.

Pietroni, N., Tarini, M., and Cignoni, P. (2010). Almost isometric mesh parameterization through abstract domains. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):621–635.

Pointwise (2018). High order mesh generation at pointwise. `https://www.pointwise.com/theconnector/2016-Q3/High-Order-Mesh-Generation-at-Pointwise.html`. Accessed: 2018-11-14.

Popović, J. and Hoppe, H. (1997). Progressive simplicial complexes. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 217–224, USA. ACM Press/Addison-Wesley Publishing Co.

Poranne, R. and Lipman, Y. (2014). Provably good planar mappings. *ACM Trans. Graph.*, 33(4):76:1–76:11.

Porumbescu, S. D., Budge, B., Feng, L., and Joy, K. I. (2005). Shell maps. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 626–633. ACM.

Poya, R., Sevilla, R., and Gil, A. J. (2016). A unified approach for a posteriori high-order curved mesh generation using solid mechanics. *Computational Mechanics*, 58(3):457–490.

Praun, E., Sweldens, W., and Schröder, P. (2001). Consistent mesh parameterizations. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 179–184.

Rabinovich, M., Poranne, R., Panozzo, D., and Sorkine-Hornung, O. (2017). Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4):37a.

Ragan-Kelley, J., Barnes, C., Adams, A., Paris, S., Durand, F., and Amarasinghe, S. (2013). Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *Acm Sigplan Notices*, 48(6):519–530.

Remacle, J.-F. (2017). A two-level multithreaded delaunay kernel. *Computer-Aided Design*, 85:2–9.

Requicha, A. G. (1980). Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv.*, 12(4):437–464.

Roca, X., Gargallo-Peiró, A., and Sarrate, J. (2012). Defining quality measures for high-order planar triangles and curved mesh generation. In Quadros, W. R., editor, *Proceedings of the 20th International Meshing Roundtable*, pages 365–383, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ruiz-Gironés, E., Gargallo-Peiró, A., Sarrate, J., and Roca, X. (2017). An augmented lagrangian formulation to impose boundary conditions for distortion based mesh moving and curving. *Procedia Engineering*, 203:362 – 374. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.

Ruiz-Gironés, E., Roca, X., and Sarrate, J. (2016a). High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3d cad representation. *Computer-Aided Design*, 72:52 – 64. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

Ruiz-Gironés, E., Sarrate, J., and Roca, X. (2016b). Generation of curved high-order meshes with optimal quality and geometric accuracy. *Procedia Engineering*, 163:315 – 327. 25th International Meshing Roundtable.

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

Sacht, L., Vouga, E., and Jacobson, A. (2015). Nested cages. *ACM Transactions on Graphics (TOG)*, 34(6):170.

Sadek, E. A. (1980). A scheme for the automatic generation of triangular finite elements. *International Journal for Numerical Methods in Engineering*, 15(12):1813–1822.

Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. (2001). Texture mapping progressive meshes. In *ACM SIGGRAPH*, pages 409–416.

Schmidt, P., Born, J., Campen, M., and Kobbelt, L. (2019). Distortion-minimizing injective maps between surfaces. *ACM Transactions on Graphics (TOG)*, 38.

Schneider, T., Hu, Y., Gao, X., Dumas, J., Zorin, D., and Panozzo, D. (2019). A large scale comparison of tetrahedral and hexahedral elements for finite element analysis.

Schreiner, J., Asirvatham, A., Praun, E., and Hoppe, H. (2004). Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877.

Schüller, C., Kavan, L., Panozzo, D., and Sorkine-Hornung, O. (2013). Locally injective mappings. In *Symposium on Geometry Processing*, pages 125–135.

Sharp, N. and Crane, K. (2020). A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum (SGP)*, 39(5).

Sharp, N., Soliman, Y., and Crane, K. (2019). Navigating intrinsic triangulations. *ACM Transactions on Graphics (TOG)*, 38(4):55.

Sheffer, A., Praun, E., and Rose, K. (2006). Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171.

Shephard, M. S., Flaherty, J. E., Jansen, K. E., Li, X., Luo, X., Chevaugeon, N., Remacle, J.-F., Beall, M. W., and O'Bara, R. M. (2005). Adaptive mesh generation for curved domains. *Applied Numerical Mathematics*, 52(2):251 – 271. ADAPT '03: Conference on Adaptive Methods for Partial Differential Equations and Large-Scale Computation.

Sherwin, S. and Peiró, J. (2002). Mesh generation in curvilinear domains using high-order elements. *International Journal for Numerical Methods in Engineering*, 53(1):207–223.

Shewchuk, J. (1996). Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. *Applied computational geometry towards geometric engineering*, pages 203–222.

Shewchuk, J. R. (1998). Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95. ACM.

Shewchuk, J. R. (2002). Constrained delaunay tetrahedralizations and provably good boundary recovery. In *IMR*, pages 193–204.

Shoham, M., Vaxman, A., and Ben-Chen, M. (2019). Hierarchical functional maps between subdivision surfaces. *Computer Graphics Forum*, 38(5):55–73.

Si, H. (2015). Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36.

Si, H. and Gärtner, K. (2005). Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *IMR*, pages 147–163. Springer.

Si, H. and Shewchuk, J. R. (2014). Incrementally constructing and updating constrained delaunay tetrahedralizations with finite-precision coordinates. *Engineering with Computers*, 30(2):253–269.

Simnett, T. J. R., Laycock, S. D., and Day, A. M. (2009). An Edge-based Approach to Adaptively Refining a Mesh for Cloth Deformation. In Tang, W. and Collomosse, J., editors, *Theory and Practice of Computer Graphics*. The Eurographics Association.

Smith, J. and Schaefer, S. (2015). Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34(4):70:1–70:9.

Sorger, C., Frischmann, F., Kollmannsberger, S., and Rank, E. (2014). Tum.geoframe: automated high-order hexahedral mesh generation for shell-like structures. *Eng. Comput.*, 30(1):41–56.

Sorkine, O., Cohen-Or, D., Goldenthal, R., and Lischinski, D. (2002). Bounded-distortion piecewise mesh parameterization. In *Proceedings of the Conference on Visualization*, pages 355–362.

Springborn, B. (2020). Ideal hyperbolic polyhedra and discrete uniformization. *Discrete & Computational Geometry*, 64(1):63–108.

Springborn, B., Schröder, P., and Pinkall, U. (2008). Conformal equivalence of triangle meshes. *ACM Trans. Graph.*, 27(3):1–11.

Stees, M. and Shontz, S. M. (2017). A high-order log barrier-based mesh generation and warping method. *Procedia Engineering*, 203:180 – 192. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.

Stephenson, K. (2005). *Introduction to circle packing: The theory of discrete analytic functions*. Cambridge University Press.

Steve L. Karman, J T. Erwin, R. S. G. and Stefanski, D. (2016). High-order mesh curving using wcn mesh optimization. In *46th AIAA Fluid Dynamics Conference, AIAA AVIATION Forum*.

Sun, J., Wu, T., Gu, X., and Luo, F. (2015). Discrete conformal deformation: algorithm and experiments. *SIAM Journal on Imaging Sciences*, 8(3):1421–1456.

Surazhsky, V. and Gotsman, C. (2001). Morphing stick figures using optimized compatible triangulations. In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pages 40–49. IEEE.

Suwelack, S., Lukarski, D., Heuveline, V., Dillmann, R., and Speidel, S. (2013). Accurate surface embedding for higher order finite elements. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA'13, pages 187–192, New York, NY, USA. ACM.

Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M. H. (2003). Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, volume 3, pages 47–54.

Thiery, J.-M., Tierny, J., and Boubekeur, T. (2012). Cager: Cage-based reverse engineering of animated 3d shapes. *Comput. Graph. Forum*, 31(8):2303–2316.

Tong, W.-h. and Kim, T.-w. (2009). High-order approximation of implicit surfaces by g1 triangular spline surfaces. *Computer-Aided Design*, 41(6):441–455.

Toulorge, T., Geuzaine, C., Remacle, J.-F., and Lambrechts, J. (2013). Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8 – 26.

Toulorge, T., Lambrechts, J., and Remacle, J.-F. (2016). Optimizing the geometrical accuracy of curvilinear meshes. *Journal of Computational Physics*, 310:361 – 380.

Tournois, J., Wormser, C., Alliez, P., and Desbrun, M. (2009). Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics*, 28(3):Art–No.

Turner, M. (2017). *High-order mesh generation for CFD solvers*. PhD thesis, Imperial College London.

Turner, M., Peiró, J., and Moxey, D. (2016). A variational framework for high-order mesh generation. *Procedia Engineering*, 163:340 – 352. 25th International Meshing Roundtable.

Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 13(3):743–768.

Villard, J. and Borouchaki, H. (2002). Adaptive meshing for cloth animation. In *In Proceedings of the 11th International Meshing Roundtable (IMR 2002*, pages 243–252.

Wang, B., Ferguson, Z., Schneider, T., Jiang, X., Attene, M., and Panozzo, D. (2021). A large-scale benchmark and an inclusion-based algorithm for continuous collision detection. *ACM Trans. Graph.*, 40(5).

Wang, L., Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., and Shum, H.-Y. (2003). View-dependent displacement mapping. In *ACM Transactions on graphics (TOG)*, volume 22, pages 334–339. ACM.

Wang, R., Liu, L., Yang, Z., Wang, K., Shan, W., Deng, J., and Chen, F. (2016). Construction of manifolds via compatible sparse representations. *ACM Transactions on Graphics (TOG)*, 35(2):1–10.

Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., and Shum, H.-Y. (2004). Generalized displacement maps. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 227–233. Eurographics Association.

Weber, O. and Zorin, D. (2014). Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph.*, 33(4):75:1–75:12.

Wicke, M., Ritchie, D., Klingner, B. M., Burke, S., Shewchuk, J. R., and O'Brien, J. F. (2010). Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph.*, 29(4).

Wojtan, C. and Turk, G. (2008). Fast viscoelastic behavior with thin features. *ACM Trans. Graph.*, 27(3):1–8.

Xie, Z. Q., Sevilla, R., Hassan, O., and Morgan, K. (2013). The generation of arbitrary order curved meshes for 3d finite element analysis. *Computational Mechanics*, 51(3):361–374.

Ye, K., Ni, W., Krieger, M., Ma'ayan, D., Wise, J., Aldrich, J., Sunshine, J., and Crane, K. (2020). Penrose: from mathematical notation to beautiful diagrams. *ACM Transactions on Graphics (TOG)*, 39(4):144–1.

Yerry, M. A. and Shephard, M. S. (1983). A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications*, 3(1):39–46.

Yu, Y., Wei, X., Li, A., Liu, J. G., He, J., and Zhang, Y. J. (2020). Hexgen and hex2spline: Polycube-based hexahedral mesh generation and spline modeling for isogeometric analysis applications in ls-dyna.

Yvart, A., Hahmann, S., and Bonneau, G.-P. (2005a). Hierarchical triangular splines. *ACM Trans. Graph.*, 24(4):1374–1391.

Yvart, A., Hahmann, S., and Bonneau, G.-P. (2005b). Smooth adaptive fitting of 3d models using hierarchical triangular splines. In *International Conference on Shape Modeling and Applications 2005 (SMI'05)*, pages 13–22. IEEE.

Zayer, R., Steinberger, M., and Seidel, H.-P. (2017). A gpu-adapted structure for unstructured grids. *Comput. Graph. Forum*, 36(2):495–507.

Zhang, E., Mischaikow, K., and Turk, G. (2005). Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24(1):1–27.

Zhang, P., Vekhter, J., Chien, E., Bommes, D., Vouga, E., and Solomon, J. (2020). Octahedral frames for feature-aligned cross fields. *ACM Trans. Graph.*, 39(3).

Zhang, S., Li, Z., Zhang, H., and Yong, J. (2011). Multi-resolution mesh fitting by b-spline surfaces for reverse engineering. In *2011 12th International Conference on Computer-Aided Design and Computer Graphics*, pages 251–257.

Zhou, K., Synder, J., Guo, B., and Shum, H.-Y. (2004). Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 45–54, New York, NY, USA. ACM.

Zhou, M., Xie, T., Seol, S., Shephard, M. S., Sahni, O., and Jansen, K. E. (2012). Tools to support mesh adaptation on massively parallel computers. *Engineering with Computers*, 28(3):287–301.

Zhou, Q., Grinspun, E., Zorin, D., and Jacobson, A. (2016). Mesh arrangements for solid geometry. *ACM Transactions on Graphics (TOG)*, 35(4):1–15.

Ziel, V., Bériot, H., Atak, O., and Gabard, G. (2017). Comparison of 2d boundary curving methods with modal shape functions and a piecewise linear target mesh. *Procedia Engineering*, 203:91 – 101. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.

Zorin, D. (2000). Subdivision for modeling and animation. *SIGGRAPH2000 course notes*.