DIFFERENTIAL GEOMETRY BASED REPRESENTATIONS FOR MESHES AND CORRESPONDENCES

By

Yuanzhen Wang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science - Doctor of Philosophy

2013

ABSTRACT

DIFFERENTIAL GEOMETRY BASED REPRESENTATIONS FOR MESHES AND CORRESPONDENCES

By

Yuanzhen Wang

Geometry processing is one of the major subfields of computer graphics, which deals with the representation, display, and animation of virtual scenes. Geometric models are involved not only in the digitization of three-dimensional (3D) objects, but also in rendering and animation, i.e., the processes of creating digital imagery from such models and deforming shapes over time. The representations of shapes and their relations are crucial to the accuracy and efficiency of geometry processing. Thus improved representations can potentially benefit a wide range of applications, including computer aided design, visualization, simulation, education, training, and electronic entertainment.

Despite the rapid progress in the past few decades, several known challenges remain in geometry processing, including: Storage in runtime volumetric data structure has a space complexity that is cubic in resolution, or worse when the shape to represent changes dynamically; Automatic hexahedral meshing with well-shaped elements, i.e., decomposing a shape into cube-like cells, is highly desirable in finite element approaches in physical simulation, but is still considered as an open problem; Coordinate-system-independent representations of curved surfaces representing the boundary of 3D objects are indispensable in robust 3D model acquisition through scanners and animation, but are often highly redundant, or inefficient in conversion to explicit representations or deformation; Shape correspondence is becoming increasingly important with the rapid growing number of digitized 3D models, but can be hard to establish under large deformation.

We address these seemingly diverse issues with methods based on a uniform framework called

discrete differential geometry, which strives to preserve desired structures of smooth shapes in their digitized representations, through carefully designed discretization. In this dissertation, we present novel meshing and correspondence inference techniques, designed for generating high quality representations of both the geometric objects and the mappings among these objects, which benefit subsequent geometry processing, modeling, and scientific computing.

First, we use edge lengths and dihedral angles of polyhedra to represent the first and second fundamental forms, two continuous quantities that measure the lengths and curvatures of curved surfaces, respectively. Mirroring the fundamental theorem of surfaces involving these fundamental forms, a discrete condition for the existence of an immersion of the surface in the discrete representation is also provided, which leads to an efficient method applicable in shape deformation.

Second, in order to express correspondence maps between two shapes, we propose a vector field map representation, extended from a spectral representation called functional map. The generalized representation is capable of handling the constraints for correspondence inference in the important angle-preserving case. The extension also makes it possible to handle function transfer between shapes with different topology.

Third, we present an automatic hexahedral meshing tool, based on a systematic treatment that eradicates common singularities that would lead to degenerate volumetric cells. Such singularities could be abundant in automatically generated edge direction fields guiding the interior and boundary layouts of the hexahedra in the final result.

Finally, we develop a compact data structure for volumetric shapes of arbitrary topology. It offers all commonly used incidence queries among its elements with constant time complexity, and can be adapted to work with dynamically changing topology.

We demonstrate the numerical test results, discuss future work, and point out the potential applications and techniques.

To my wife Qiong Wu.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Yiying Tong, for his tremendous help during my doctoral study, which is the most cherishable time in my life. I would not have been able to go through these four years without his continuous encouragement and inspiring guidance. From him, I learned a great amount of knowledge in computer graphics, the systematical way of doing research and the earnest attitude of pursuing the truth. This dissertation would not have been possible without his efforts in careful reviews.

I would also like to express my gratitude and deep appreciation to Prof. Rong Jin, Prof. Charles Owen, and Prof. Guowei Wei for being my dissertation committee members and giving me insightful comments and guidance. I am also grateful to Prof. Guowei Wei for the enlightenment to me in the research of Biomedical Imaging. I would also like to thank Prof. Jin Huang and Prof. Kun Zhou for sharing their research experiences and softwares. I also wish to thank Beibei Liu for the collaboration on surface meshing and correspondence, Tengfei Jiang for the collaboration on automatic hexahedral meshing, Xin Feng for the collaboration on 3D volume data structures, and Xiaojun Wang for the discussion and codes on fluid simulations, among other colleagues. I also want to thank Lujin Wang, my mentor in NVIDIA, for her help in my improvement of coding and software development skills.

At personal level, I am indebted to my parents, for their endless love, encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The world we live in is often approximated as a three-dimensional (3D) Euclidean space. However, our visual system only creates two-dimensional (2D) projections of 3D shapes. Despite the emergence of stereo displaying technologies, such as volumetric displays, holographic displays, 2D displays dominate as the major visual output device from computer to human. *Computer Graphics* bridges the gap between 3D models of real-world objects and the 2D imagery created through the rendering process. In this process, the representations of geometric shapes in computer are of ultimate importance.

Unfortunately, the discretization of 3D objects is almost always plagued with loss of certain properties in the continuous world. It is thus a fundamental task in computer graphics to find the proper representations for the target application. In computer aided geometric design and physical simulation, many representation schemes for solid objects have been developed. For example, using *cell decomposition* techniques, a solid object is represented by a collection of disjoint cells, whose union forms the original 3D volume occupied by the object. The adjacency relationship among the decomposed cells is usually represented by combinatoric descriptions. However, in applications from animation and rendering, in order to display objects for human visual perception

efficiently, usually only the boundaries of objects are relevant to generating the images, since the interior of opaque objects is invisible to human eyes. In this case, *boundary representation* (or b-rep) is another scheme developed to represent objects by their boundary surfaces alone. It improves the efficiency both in memory space and computational time tremendously, roughly speaking, from $O(n^3)$ to $O(n^2)$, with n being the resolution of the model along each dimension.

In both volume and surface cases, concrete representations and actual data structures are necessary. *Mesh* is the most widely used to represent both surfaces and volumes. A mesh is formed by the cell decomposition process, but restricted to using simple cells, with certain adjacency rules. For example, a triangle mesh approximates surfaces using flat triangles, such that each pair of triangles only intersect at one vertex or one edge, or are otherwise disjoint. Similarly, in the volumetric case, a tetrahedral mesh approximates a solid object using tetrahedra only.

The pipeline in geometric modeling in computation is often called Digital Geometry Processing (DGP), seen as a special form of the more generic digital signal processing (DSP). However, DGP poses important challenges as the sampling in meshes is usually highly irregular unlike the regular 1D audio, 2D image, or 3D video in DSP. Again, how the geometric signals are represented is fundamental to addressing this issue. In the following sections, we first go over the topics in DGP, noting those that would directly benefit from our work; we then introduce the elementary concepts in the theory of discrete differential geometry (DDG), which strives to preserve important structure during the discretization process, as the geometric signals come from digitizing smooth 3D surfaces and volumes; finally, we summarize the main contributions, which focus on using a DDG-based approach to address the question of how shapes should be represented in DGP.

1.1 Digital Geometry Processing

We provide the context of our work here, by presenting the usual pipeline of geometry processing and noting the relevance of our work within the overall structure.

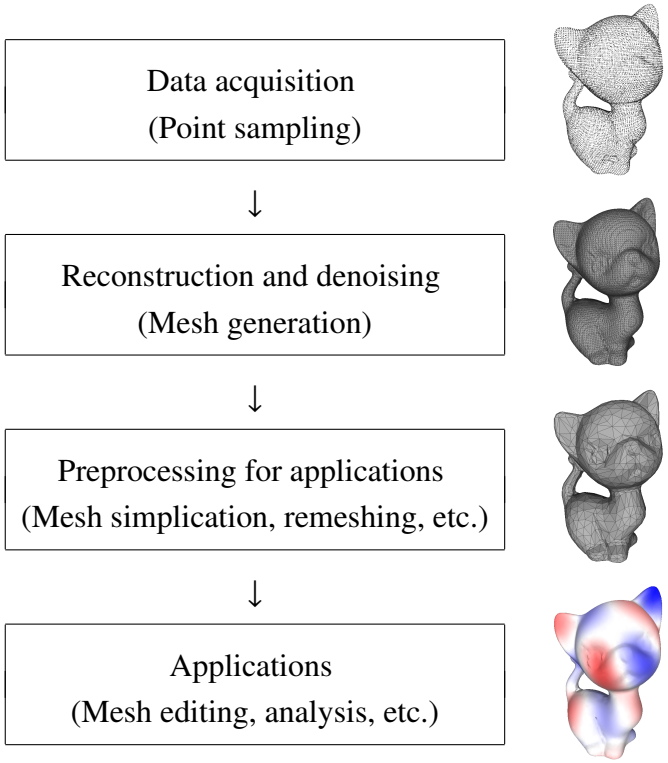


Figure 1.1: Main stages in digital geometry processing. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

In computation, especially in applications in Finite Element Methods (FEM), Computational Fluid Dynamics (CFD), and Computer-aided Design (CAD), to represent the shapes of real world objects, we first need to construct or acquire discrete geometric models. 3D scanners based on various techniques have now become common-place. They can provide us with the information of sampling points on the geometric shapes. Processing raw dataset from scanners leads to the research on point-based geometry processing [59], e.g., Pauly et al. [89] proposed a shape modeling algorithm based solely on sampling points. However, a point cloud without additional structure

may not be adequate for most other applications.

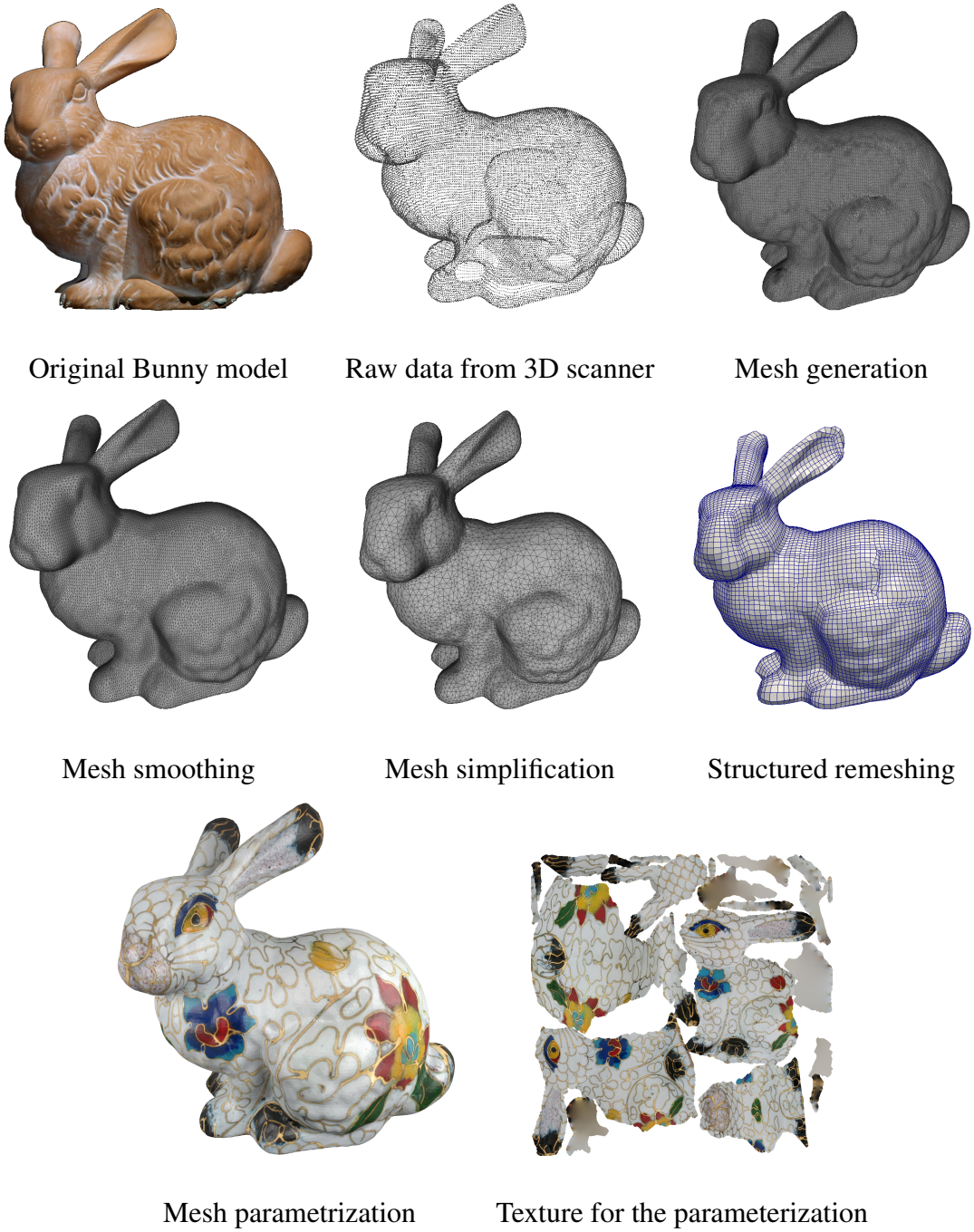


Figure 1.2: Example stages in digital geometry processing pipeline.

1.1.1 Mesh Generation and Surface Reconstruction

In practice, the knowledge about point locations only is insufficient to fulfill the tasks from most applications, including regular rendering and physical simulation. To establish the complete surface, additional information is required, such as how the points are connected with each other and how the gaps between points are filled with facets. More formally, the *topology* of those points is necessary for practical purposes, such as ray tracing, interference detection, inside/outside query.

In computational science, the term *mesh*, refers to the structure that not only describes the points locations but also tells us about the connections between them. There are a multitude of methods in mesh generation designed for various application purposes [71]. Our work [35] on volume mesh data structure can be seen as an extension of the connectivity representation of surfaces to volumes. It can also be used as an efficient representation for the domain within which the surface reconstruction can be performed for scanner output, through methods such as marching tetrahedra.

1.1.2 Mesh Smoothing

As mentioned before, we can use the sample point locations obtained from 3D scanners or vision techniques to construct the raw 3D models of objects. However, this process usually produces noisy data. It is not that unexpected, since noise is unavoidable for almost any 3D data acquisition techniques, unless the original surface is digitally created. Consequently, one sub-research area of geometry processing is on smoothing out the noise in raw sampling point datasets, obtained directly from real objects, in order to produce a mesh with higher quality than the original one. Research on smoothing algorithms recently received intense attention. Diffusion approaches including spectral analysis, Laplacian smoothing and curvature flow, and global minimization for certain forms of

nonlinear energy, are developed. A lot of alternative approaches are still under active research for specific application purposes. Our work [110] on rigid motion-invariant surface construction using discrete fundamental forms can potentially provide new definitions on noise, and lead to denoising processes that are independent of possible translation or rotation. Moreover, our novel spectral vector field analysis tool can be potentially applied to mesh smoothing by relying only on low frequency signals.

1.1.3 Mesh Simplification/Subdivision

Once a low-noise mesh is obtained, a subsequent step in geometry processing is to make the mesh coarse/dense for different applications. The raw dataset from 3D scanners is often sampled with a limited range of resolution. It is therefore desirable to have a postprocessing tool (e.g., the one proposed by Daniels et al. [25]) to manually control the mesh density. In some applications, e.g., fluid simulation and physical simulation, a large number of points can make the computation extremely slow. Therefore, a coarse mesh is favorable. On the other hand, a finer mesh may be more suitable for other applications where fine geometric details are necessary [43], e.g. 3D model sculpture. Our compact data structure [35] may prove indispensable when the required sampling density is extremely high.

1.1.4 Structured Remeshing

In FEM and other computational applications, a structured mesh (e.g., an all-quadrilateral mesh with each vertex adjacent to exactly four vertices) may lead to stabler or more accurate simulation than unstructured mesh (e.g., triangle meshes or polygonal meshes.). Recently, the research on remeshing from triangulation to quadrangulation is actively explored and there are a lot of excellent

results obtained from researchers [32, 107, 55, 12, 46, 25, 90, 119]. However, a fully automated method of generating high quality hexahedral meshes from tetrahedral meshes is still missing. Our work [49] based on smooth frame field has made progress towards such a meshing tool.

1.1.5 Mesh Deformation/Editing

With the increasing computing capability and user demands on 3D models, mesh deformation/editing becomes a convenient tool to generate new shapes from existed ones. Intuitive (real-time) editing methods [114, 101, 69, 47] allow artists and designers to creatively sculpture models without interruption of inspiration. Our work [110], based on representing discrete differential forms as local surface descriptors, provides such an editing tool, with quality matched only by much slower non-linear methods.

1.1.6 Mesh Parametrization

Without textures, meshes look like gypsum sculptures. To put the vivid textures (and other material properties) stored as planar images onto 3D geometric objects, surface parametrization, the process of flattening patches of the surface, is necessary. Although its early uses in computer graphics had been merely for texture mapping, parametrization, as a fundamental concept in differential geometry, has since found applications in many other mesh processing routines [37], such as detail mapping, detail synthesis, detail transfer, mesh completion, mesh editing, remeshing, mesh compression, and surface fitting. Our work [110] on rigid-motion invariant surface reconstruction provides a simple method for a quasi-isometric parameterization, which introduces low distortion and produces a natural boundary condition.

1.1.7 Mesh/Shape Correspondence

In recent years, mesh/shape correspondence inference became an active research area in computer graphics and computer vision. With the rapid increase in 3D models repositories (e.g., Google 3D warehouse), an interface for efficiently exploring large sets of models is necessary for users. However, it is not an easy task when the exploration is under a certain user preference, for example, browsing all chairs from a collection of furniture. Traditionally, it is merely based on the semantic method, which requires manually associating each model with several descriptive labels. But the labeling process is usually inefficient and not scalable. Thus, an automatic, efficient and accurate mesh exploration method is highly in demand. The analysis of shapes and their relations is a necessary prerequisite to such applications as mesh exploration. Our vector-field map representation presents a tool for establishing such relations among different shapes within the collection, and for their further spectral analysis.

To summarize, with extensive applications in computer graphics and geometric modeling, geometry processing is a vast research area. As stated above, our work is relevant for multiple stages in the spectrum of DGP, ranging from acquisition to deformation and shape analysis. There are many other novel works for interesting specific application topics in the same context, such as constructive solid geometry, computational geometry, and collisions detections. However, it is beyond the scope of this dissertation to survey all of these topics.

1.2 Discrete Differential Geometry

Discrete Differential Geometry (DDG) has emerged as a rapidly developing research area to study the proper discretization in recent decade. As the basic tool and theoretical foundation of this dissertation, we first briefly introduce the basic concepts here.

From the mathematical theory point of view, DDG bridges the gap between classical differential geometry, which focuses on smooth manifolds, and discrete geometry, which is concerned more with discrete geometric objects (i.e. polygons in 2D, polyhedra in 3D, simplicial complexes, etc.). The goal of DDG is to build discrete theories that mirror their counterparts in the classical differential geometry, in the sense that they preserve the important structures defined directly on the discrete level even at a coarse resolution. More importantly, it should not only be characterized as a pure mathematics field but also an important computational tool to other disciplines that rely heavily on geometry processing, such as computer-aided design, physical simulation, and computer graphics.

DDG is ubiquitously involved in operations of geometry processing. A survey edited by Bobenko et al. [8] has compiled recent developments made by researchers from a wide range of fields, including mathematics, physics, computational geometry and computer graphics. It pointed out some major research directions in the current DDG community:

- Discretization of Surfaces. This topic mainly focuses on what discrete representation of surfaces, among triangle meshes, polygon meshes, circle patterns [8], etc., should be properly chosen for specific applications. Some surfaces are often defined through continuous procedures: e.g., constrained surface area minimization for finding minimal surfaces [84]; Willmore flow [9] minimizing an energy measuring how much a surface deviates from a sphere. In simulating such procedures, an arbitrary discretization may not be able to capture those defining properties accurately, since often there is no known "perfect" discretization that preserves all the properties from the continuous theory.

- Discrete Curvature for Curves and Surfaces. A related topic in determining important shape descriptors with a given discretization of surfaces is the definition and evaluation of the dis-

crete curvatures [40]. Curvature plays a central role in describing shapes since it describes how much curves or surfaces deviate from straight lines and flat planes, respectively. Furthermore, they are often the determining factors in geometry processing procedures and physical simulations. For example, computing the minimal surfaces depends on a discrete method to estimate mean curvatures [91]. In Chapter 2 of this dissertation, curvature estimates for triangle meshes will be discussed in detail.

- Geometric Realizations of Combinatorial Surfaces. This topic is about finding solutions for some classical problems such as "using n vertices to construct a *combinatorial* data with maximum genus for a surface." Ziegler [122]'s work shows that it is realizable to construct a triangle mesh with n vertices for a surface with genus $O(nlogn)$, although while it is still unknown for a surface with genus $O(n^2)$. Such problems are nontrivial, since we must examine the surface from at least three different points of view — geometrically, topologically and combinatorially. However, they are highly relevant in geometry processing, as even the basic data structure in geometric modeling depends on the combinatoric representation of the underlying topology of the object.

- Geometry Processing and Modeling with Discrete Differential Geometry. There are many important applications involving geometry processing, especially with the recent increase in the amount of geometric data collected. A lot of results from DDG [28, 9, 56, 31, 21, 75, 111, 15] have been applied to geometry processing. The main theme of this dissertation also lies within this topic.

In this dissertation, we show our main contributions on surface and volume meshing and shape correspondences. We extend the curvature discretization of surfaces to the local shape description based on a discrete version of fundamental forms, from which the curvatures are derived. This

representation benefits further operations in geometry processing, such as deformation and animation. The discretization of volumes can be seen as the natural next step of DDG approaches to the discretization of surfaces. Volumetric meshing has been a long-standing topic in the Finite Element Methods (FEM) and Finite Volume Methods (FVM) communities for its importance in describing solid objects and domains of fluid motions. We draw upon the recent developments in DDG-based quadrilateral meshing, and make progress towards the challenging problem of automatic high quality hexahedral meshing. In the process, we develop a novel compact volumetric mesh data structures as we have to deal with large data, based on combinatoric representations of the topology.

By extending correspondence inference using the functional map representation, we develop a representation of mappings between manifold surface meshes through a linear mapping between the spaces of tangential vector fields. The theoretical foundation of this representation is a spectral vector field analysis tool, which we develop to handle various vector fields on surfaces.

1.3 Surface and Volume Representations

As mentioned above, in 3D surface meshing, for further processing operations after obtaining the sampling points (e.g., from 3D scanners), a proper representation of the connectivity between points is often required. There are basically two categories of methods to solve this problem. One is to build combinatorial structures such as Delaunay triangulations by interpolating points [10, 98]. The other one is to compute an implicit scalar function in 3D from the sampling points, and then extract the iso-surface from the implicit function. For example, Marching Cubes, designed by Lorensen and Cline [73], is the most famous algorithm for extracting the polygonal mesh approximating an iso-surface from implicit functions. Both categories of methods discussed

above can be extended to 3D volume meshing; our research on 3D volume meshing is aligned with the combinatorial structures based methods. In 3D volume meshing and solid modeling, volumetric combinatorial representations [67], as extensions to surface combinatorial structures, are used for representing numerous topological models. Tournois et al. [108] provided a robust and high quality 3D tetrahedral volumetric mesh generation algorithm.

In this section, we introduce the following three research projects on surface and volume meshing:

- local and rigid-motion-invariant surface representations based on Differential Coordinates;
- compact data structure to represent polyhedral meshes;
- automatic hexahedral meshing tool with singularity corrections.

1.3.1 Surface Representation Based on Differential Coordinates

A rigid-motion-invariant mesh representation forms the basis for capturing the geometric details when the shapes undergo deformations. Yu et al. [114], Zayer et al. [115] define *differential coordinates* based on the gradients of original surface coordinates, while Zhou et al. [121], Lipman et al. [69], Sorkine et al. [101] manipulate the Laplacian, trace of the Hessian matrix (second order derivatives), of the spatial coordinates.[i] However, those methods are non-linear when dealing with rotations, since gradients and Laplacians of coordinates are not rotation-invariant.

[i] The terms such as "gradients" and "Laplacian" are defined in Chapter 2.

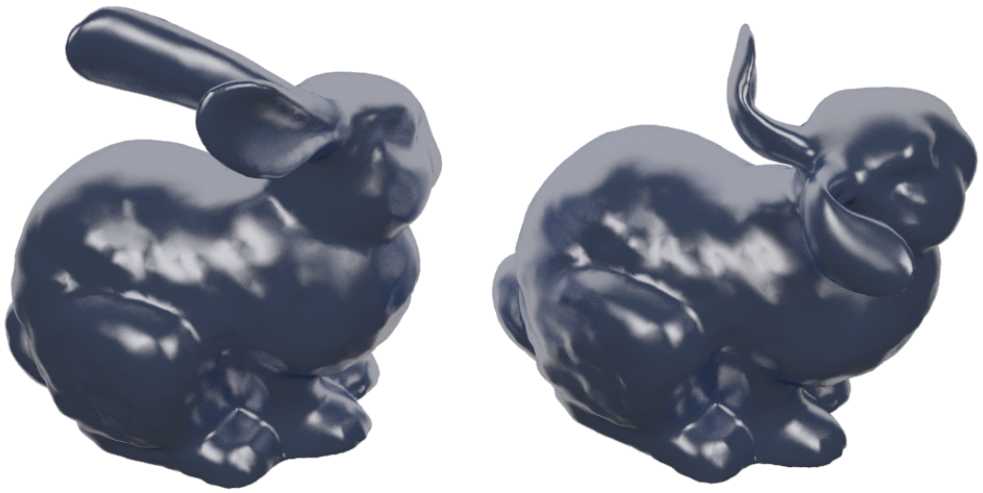


Figure 1.3: The Bunny model on the right side is deformed through our rigid-motion-invariant representation.

Unlike gradients of surface coordinates and Laplacian of the spatial coordinates, the first and second fundamental forms can provide both local and rotation-invariant information. Through integrating the surface equations that describe the relation between local shape descriptor and global shape (covered in section 2.1.2.6 in detail), we can reconstruct the original surface, up to an arbitrary rotation and translation, from the fundamental forms. Based on this fact, Lipman et al. [70] proposed a method of linear rotation-invariant differential coordinates for discrete triangle meshes.

A seemingly straightforward way is by directly attaching a local coordinate system (formed by 2 tangent vectors and the normal) at each vertex as suggested by Lipman et al. [70], and extracting the local information that can be analyzed to obtain first and second fundamental forms. This leads to a discretization of the *first discrete form* measuring the length on surface by a set of inner products between neighboring edges projected onto the tangent plane.

Besides the first discrete form, Lipman et al. [70] defined a *second discrete form* that plays a similar role as the second fundamental form, which provides the information about how the normal vector is changing in the neighborhood of each vertex. However, their discrete fundamental forms

are dependent on the choice of tangent planes, potentially degenerate in extreme cases, and highly redundant. The deformation technique based on these discrete fundamental forms cannot handle rotation and position constraints simultaneously. In contrast, our formulation uses tangent plane naturally associated with the triangles, and provides a concise representation, which can then be used to handle both rotational and positional constraints [110].

1.3.2 Compact Volume Mesh Data Structures

With current hardware technology, memory usage and operation efficiency can be well balanced for surface mesh data structures [100, 95]. For more information about surface mesh data structures, refer to two recent surveys Sieger and Botsch [100], Serna et al. [95]. Unfortunately, that is not the case in 3D volume mesh data structures. Developing a compact yet efficient volume data structure is still an active research topic. Compactness is a necessary requirement for processing large 3D volumetric objects. As shown in Table 1.1, using compact data structures for models can contribute to a memory usage reduction of more than 90%. On the other hand, efficiency in neighborhood access is important for applications in computational geometry and scientific computing, which rely on intense mesh operations such as incidence and adjacency queries. For example, the duality operators in discrete exterior calculus [28] involve both incidence and adjacent queries including vertex and edge neighborhood construction.

model name	OpenVolumeMesh	CGAL CM	Comapct Data
1mag	246,284k	334,028k	23,006k
Armadillo	502,028K	673,587k	44,634k
david	366,988k	483,532k	33,334k
dc-wt	1,402,900K	1,929,379k	125,702k
emd1590	54,696K	67,789k	6,110k
fertility	885,876K	1,205,862k	79,490k
neptune	921,616K	1,258,291k	83,442k

Table 1.1: Actual memory usage for the same meshes using OpenVolumeMesh library [83], CGAL's combinatorial maps [24] and compact data structures [35], respectively.

Alumbaugh and Jiao [3] have proposed an array-based compact data structure, which employs three lookup tables to store the incidence information between vertices and faces. However, it has limitations in providing efficient incidence and adjacent queries for edges. The main reason is that the three tables in their data structure do not provide direct information about edge indices. Thus, there are overheads in identifying a unique edge as the common member of different faces. This lack of efficient queries on edges may prevent this compact data structure from being used in a wider range of applications.

To address this issue, our approach [35] incorporates auxiliary lookup tables that contain edge incidence information. It improves the efficiency for commonly-used incidence and adjacency queries. We elaborate on this extended data structure in Chapter 6.

model name	Alumbaugh and Jiao [3]	Feng et al. [35]
1mag	19,858k	23,006k
Armadillo	38,103k	44,634k
david	29,486k	33,334k
dc-wt	111,286k	125,702k
emd1590	5,346k	6,110k
fertility	70,098k	79,490k
neptune	73,622k	83,442k

Table 1.2: Actual memory usage for a variety of meshes.

1.3.3 Automatic Hexahedral Meshing

Automatic high quality hexahedral mesh generation has sometimes been dubbed the "holy grail" in the meshing community, as such meshes benefit finite element methods due to their tensor product nature, leading to improvement in both speed and accuracy. Recent development in quadrangulation [32, 107, 55, 12, 46, 25] in computer graphics and geometric modeling has stirred up new research effort in this direction based on meshing methods guided by the cross-frame fields, fields of equivalence classes of local frames under the chiral cubic symmetry group (the set of 24 rotations that keep a cube centered at origin invariant without changing the right-handed frame to a left-handed one).

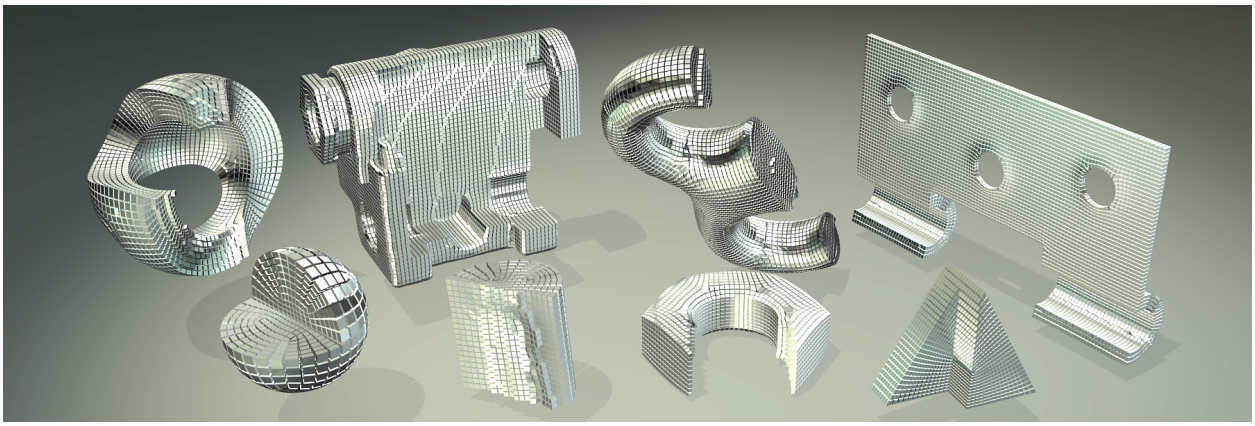


Figure 1.4: Models with hexahedral meshes generated by an automatic method.

In theory, CubeCover method [81], extending QuadCover method [55], formulated the *necessary conditions* for the volumetric parameterization, and laid down the foundation for automatic hexahedralization methods based on such parameterizations. They formulated the basic requirements of rotational and translational transitions on each interface between different charts or different parts of the same chart serving as the domains for parameterizations. The nontrivial transitions (cuts) of the domain are necessary to provide the flexibility in creating a mesh with high-quality element shape. Otherwise, a polycube-like topology would be enforced, potentially leading to huge angle distortion, undesirable for engineering or scientific computing purposes.

Huang et al. [48] proposed a method of automatically generating boundary aligned frame fields. It seems straightforward to combine this method with CubeCover. However, such a simple combination does not immediately yield a good automatic hexahedral meshing tool. Unless manually constructed or adjusted frame fields are used, there is no reason to assume that such automatically computed frame fields would satisfy these compatibility conditions [81] for degeneracy-free volumetric parameterization, and indeed they do not most of the time.

To generate high quality hexahedral meshes, we should eliminate the problem of the inconsistence caused by the automatically generated guidance field. By examining all the *singularity*

edges [81], we are able to detect the inadmissible ones and adjust them to obtain a singularity structure that would lead to a high quality hexahedral mesh [49]. However, it remains a challenging task to formulate the common global problematic cases mathematically.

1.4 Representation and Inference of Shape Correspondence

With the drastically increased number of available 3D models in the public domain, establishing correspondence between different shapes has been an ever more important research topic in computer graphics and computer vision in recent years. While the problem of shape correspondence is of great interests to the research community, the general correspondence of deformable shapes remains challenging even for similar complete shapes.

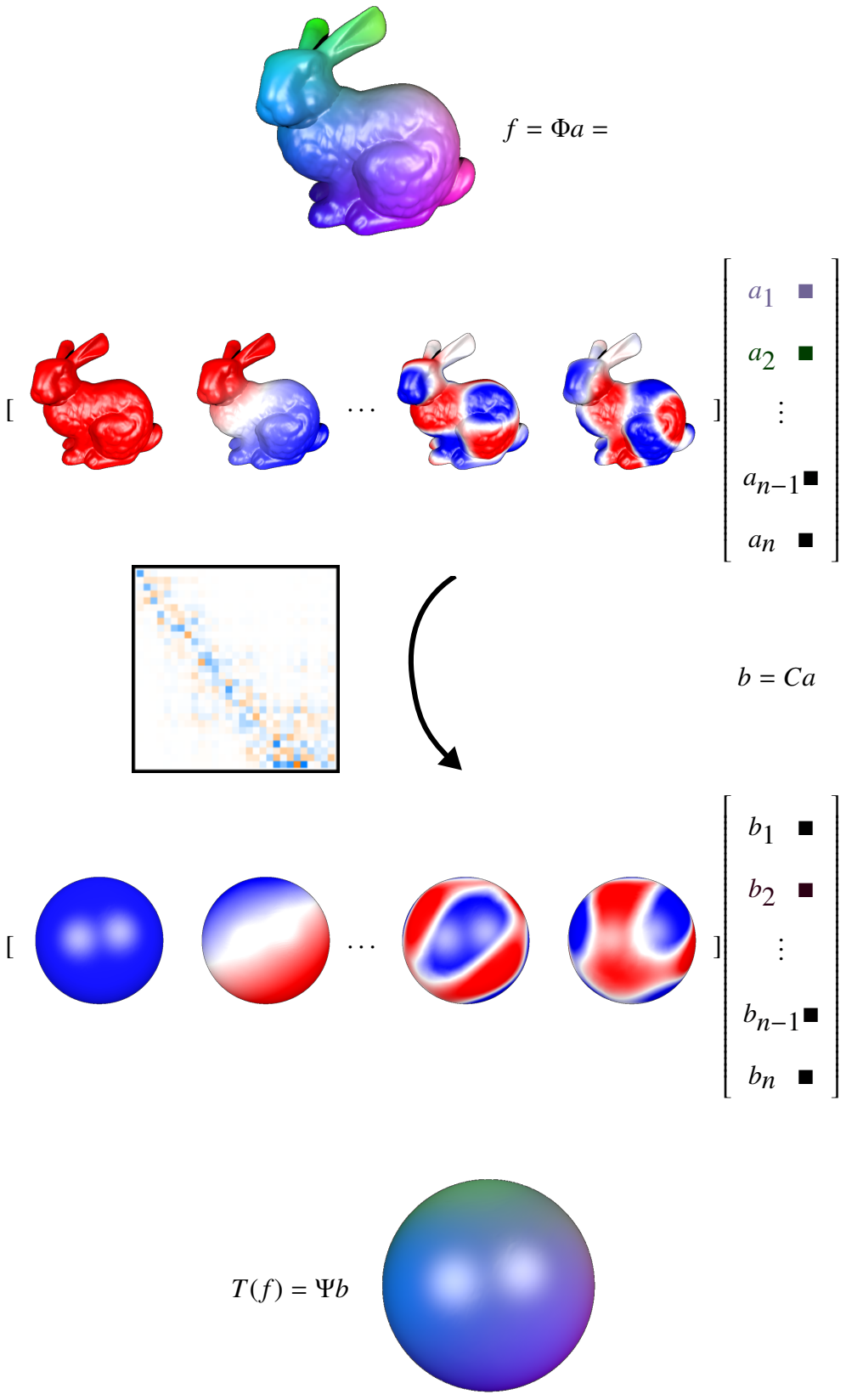


Figure 1.5: Illustration of function transfer through the matrix representation of functional map.

Inspired by the functional map representation using the linear map between functions defined on the two shapes [86], we propose to use the linear map between tangent vector field spaces as a more general representation to facilitate correspondence inference and other related applications. With our representation, it is possible to extend applicability in correspondence inference from near isometry to near conformal mapping. In applications such as joint analysis of shape collections, near isometry can be overly restrictive even for mappings between corresponding parts of the shapes. On the other hand, conformal mapping, or angle-preserving mapping, is a wider class of mappings, with isometry being a special case. For example, conformal mapping can always be achieved in theory, among any genus-0 closed surfaces, or disk-like surface patches, although it is not always compatible with landmark correspondence constraints in practice. However, even in that case, approximate angle preservation is still a desirable local property in many applications, as it leads to similar local shapes.

In correspondence inference using the functional map representation, the constraints and regularization terms often assume restricted types of correspondences. One important regularization constraint is the orthonormality of the functional map representation in their chosen bases, which restricts the map between functions to those induced by area preserving maps. It leads to significant improvement of the point-to-point correspondence quality in the near isometry case, but also to potential degradation of map quality when the deformation deviates from being area preserving. However, near area-preserving correspondences often forces even local features to distort severely. In our representation, it is replaced by the more commonly used near conformality requirement, while the implementation is still through orthonormality in vector field bases with identical sizes to function bases. Thus, our extension keeps the same compactness and efficiency while extending the applicability.

By representing the mapping in the gradient domain, we also provide a natural solution for

handling topological change when transferring scalar fields with landmarks. We first set the values at the landmark locations on the target mesh with the values on the corresponding landmark locations on the source mesh. Then, we transfer the gradient fields, and solve for the scalar fields by integrating the transferred gradient fields with the landmark values as the boundary condition. Thus, we can have precise control of the scalar field values at landmark locations, including those near the topological changes (pinching/merging points), although the transferred gradient fields may only contain the leading low frequencies in the spectral domain.

We note that our method is fully compatible with the functional map representation. Thus, any function preservation constraints for inference, including geometric descriptor, segmentation, and landmarks, can be directly incorporated into our system. With the new flexibility, we expect our compact representation to find application in improving results of point-to-point maps, mapping among collections of shape with large deformation, as well as facilitating transfer of novel types of properties, including principal curvature direction fields or even tensor fields.

1.5 Dissertation Organization

We first introduce the necessary concepts of discrete differential geometry in Chapter 2. After setting up the basic mathematical background, we discuss our discrete first and second fundamental forms and the surface reconstruction algorithm based on them in Chapter 3. Next, Chapter 4 is on the topic of surface correspondence. We propose a new way to represent the mapping between two shapes and introduce our spectral vector field analysis tool to deal with the discretization on meshes. After the topics of surface meshes, we start the discussion on volume meshes in the following two chapters. Chapter 5 presents our latest results on singularity optimization for fully automatic hexahedral mesh generations. Chapter 6 describes a novel compact, yet efficient and

easy to implement 3D volume data structure. Finally, we conclude the contributions from this dissertation and the future works in chapter 7.

Chapter 2

Rudiments of Discrete Differential Geometry

In computation, geometric objects cannot be represented by arbitrary smooth functions as often assumed in science and engineering. Piecewise polynomial approximations are often the method of choice. With the advent of parallel computational structure capable of handling a large number of simple objects efficiently, the simplest piecewise polynomial approximation, triangle meshes (linear within each triangle), has been the most popular representation in both geometric modeling and physical simulation.

Given the choice of piecewise flat surface as the digital representation of geometric shapes, important theorems or even concepts in the continuous theory often cannot be directly applied. To address this issue, the emerging research field of discrete differential geometry (DDG) aims at treating geometric objects used in computation as intrinsically discrete, with the continuous setting regarded as the limit when the discretization resolution goes to infinity. Thus the important structures modeled after the continuous theory are directly defined on the discrete objects, and

preserved at any resolution instead of only at the continuous limit. Therefore, unlike the idea of discretization through approximation to the continuous setting, DDG strives to discretize not only specific equations but also the principles in differential geometry theory. For example, if derivatives and integrals can be defined on discrete meshes, in a fashion consistent with a discrete version of Stokes' theorem (an extension of Leibniz's rule), a number of fundamental theorems derived based on Stokes' theorem from the continuous theory can be consequently applied to discrete meshes directly.

One such discrete theory on differential and integral calculus performed on meshes is *Discrete Exterior Calculus* (DEC) [28]. It provides such a systematic structure-preserving treatment on differential operators, and represents scalar and vector functions using a differential geometric concept called differential form. We employ the tools from DEC and extend some of its notions in Chapter 3 and 4.

In the rest of the chapter, we give a brief overview of the DDG concepts that are closely related to our work. We start with a discussion on the basics of the continuous theory in differential geometry of curves and surfaces, and elaborate on the corresponding discrete terms that are necessary to describe our work.

2.1 Basic Concepts from Differential Geometry

The classical theory of differential geometry studies curves and surfaces in 3D Euclidean space through techniques in differential and integral calculus. In this dissertation, we focus mainly on such low-dimensional geometry. We introduce the basic concepts without formal definition in this section. We attempt to provide the reader (presumably with at least elementary calculus knowledge and a computer science background) with the intuition behind these concepts, without involving

much differentiation despite the name "differential geometry". See, e.g., Gray et al. [41] for a rigorous introduction.

The basic goal in differential geometry is to describe geometric shapes using differential quantities in a neighborhood of each point. As the shapes of two objects are defined to be locally the same if a neighborhood in one object coincide with a neighborhood of the other after some rigid motion (translation and rotation), these quantities should be rigid motion invariant. Another goal in differential geometry is to provide the concepts of calculus for quantities defined on curved shapes, e.g., in vector field analysis.

In the following, we discuss some basic types of such differential geometric quantities, including curvature of curves, mean and Gaussian curvatures of surfaces, and the first and second fundamental forms. Then, we briefly introduce some differential operators for surfaces.

2.1.1 Differential Geometry of Curves

For simplicity, we restrict our discussion to planar curves.

2.1.1.1 Tangent vector

A straight line segment is the simplest object to locally approximate a segment of a smooth curve containing a point. More formally, given a point p on curve C, we can pick another point q near p and construct a straight line by connecting point p and q. As the point q approaches p, we obtain a line in the limit, which is called the *tangent* to the curve at point p, i.e., a line with one intersection point with the curve locally (within a neighborhood).

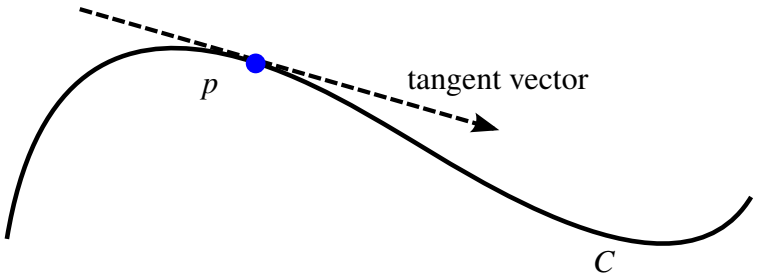


Figure 2.1: Tangent at a point p on a planar curve C.

2.1.1.2 Curvature

The tangent gives us the first order approximation of a local neighborhood of point p along the curve, which deviates from the curve with an error quadratic to the distance away from the point. However, tangent lines are insufficient to distinguish the local shapes around two different points, each located on a curve, because we can always perform a rigid-body transformation to align the two lines approximating the two neighborhoods. In other words, the straight line does not provide us with an invariant quantity to distinguish shapes.

In order to obtain more information about the shape of the neighborhood, instead of one other point, we pick two other points, infinitesimally close to p. Then we can construct a circle, which goes through those two points and p. The limit circle as the two points approach p is called the osculating circle. Its radius r tells us how much the local neighborhood of p is approximated by an arc. The value $\kappa = \frac{1}{r}$ is defined as the *curvature* at point p, which describes how fast the tangent direction is rotating when moving along the curve.[i] This quantity will only be the same if two segments of curves can be aligned through rigid motion with second order accuracy. This quantity is also rigid-motion-invariant since the radius of the osculating circle is not changed under translation or rotation.

[i] As a convention, the radius is considered infinite with the curvature considered 0, when the three points remain on a straight line.

For two curve segments to be the same, it is obvious that they must have the same curvature description. The converse is actually also true, when the curvature is described as a function of arc length distance from one of the end points. This fact is known as the fundamental theorem of planar curves. Thus, the global shape of curves can also be represented by curvature.

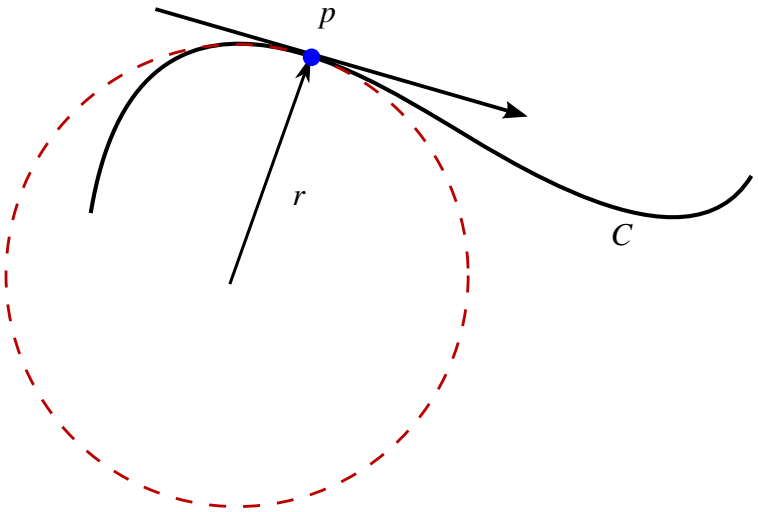


Figure 2.2: Defining curvature of a planar curve.

2.1.2 Differential Geometry of Surfaces

We now generalize the concept of curvature to the surface case, to mean curvature and Gaussian curvature. However, these two curvatures are insufficient in determining the global shape. Thus, we introduce two matrix functions, the first and second fundamental forms, with which the global surface can be determined by solving differential equations, the surface equations involving these functions. These local shape descriptors for a neighborhood of a point on the surface are crucial to our representation for the surface in Chapter 3.

2.1.2.1 Tangent plane and Normal vector

As with the curve case, a necessary concept to describe the surface locally near a point is the *tangent plane*. Given a surface S in 3D Euclidean space and a point p on S, if we follow the same procedure in defining the tangent for curves, we can find a set of tangents, which are the limits of approaching p from the other neighbouring points. It can be proved that all such tangents form a plane going through point p, which is called *tangent plane*.[ii]

Alternatively, we only need two additional points on the surface to determine a plane through points p, and take the limiting process with the two points approaching p to define the tangent plane at point p, as it is well known that a plane can be defined through three non-collinear points. (See Figure 2.3.) The vector **N** orthogonal to the tangent plane is defined as the normal of the surface at point p. If the surface describes the boundary of a 3D object, as a convection, we always pick the direction pointing from the inside of the object to the outside.

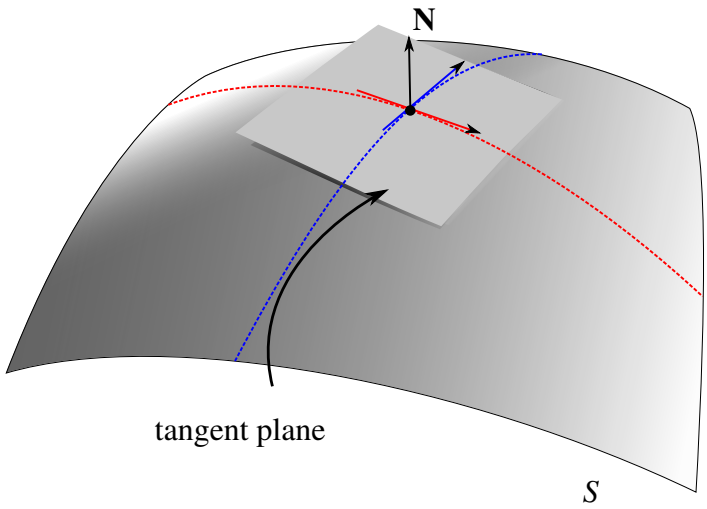


Figure 2.3: Tangent plane and normal vector **N**.

[ii] The strict mathematical proof is again omitted here. For interested readers, refer to Gray et al. [41].

2.1.2.2 Curvatures on Surface

Unlike the curve case, the curvature at a point p on surface is not unique if we construct osculating circles on the surface going through p. The definition of curvature on surface thus requires additional restrictions to produce a unique number. Instead of arbitrarily choosing two points near p, we first fix a plane going through point p, spanned by a tangent direction t and its normal vector. The intersection of the plane and the surface is a planar curve which also lies on the surface. We define the *normal curvature* (also known as the directional curvature) at p along t to be the curvature of this curve. (See Figure 2.4.) In other words, the normal curvature is determined by a given tangent direction. For curved 2D surfaces, we often refer to these normal curvatures by curvatures for simplicity.

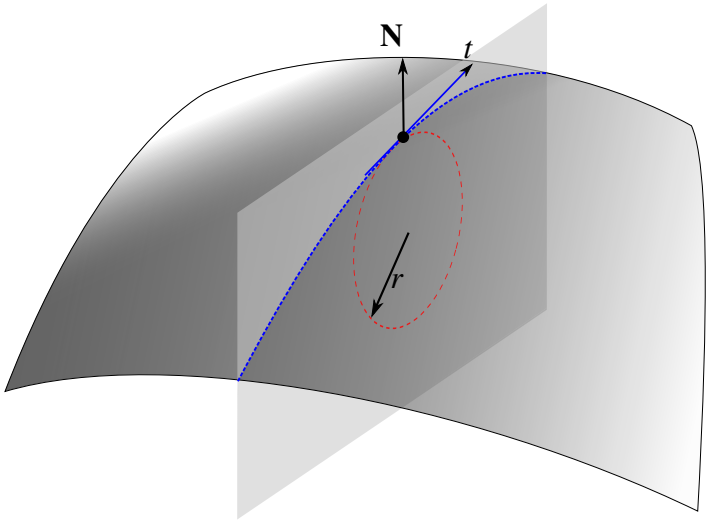


Figure 2.4: Normal Curvature of a 3D curved surface.

2.1.2.3 Principal Curvatures

Through rotating the plane in the definition of normal curvature around normal **N**, we can find all possible curvature values. Among those values, the maximum and minimum of the curvatures

values are called the *principal curvatures* at p, often denoted as κ_1 and κ_2. The tangent directions associated with these two curvatures are called the *principal directions*.

2.1.2.4 Mean Curvature and Gaussian Curvature

The *mean curvature* κ_H can be considered as the average of the curvature values along all directions at point p. Using $\kappa(\theta)$ to denote the curvature after rotating the normal plane by an angle of θ starting from an arbitrary normal plane, the mean curvature at p is defined as

$$\kappa_H = \frac{1}{2\pi} \int_0^{2\pi} \kappa(\theta) d\theta \tag{2.1}$$

Simple evaluation of the integral shows that $\kappa_H = \frac{\kappa + \kappa}{2}$.

The *Gaussian curvature* at p, usually denoted as κ_G, is defined as the product of the two principal curvatures,

$$\kappa_G = \kappa_1 \kappa_2 \tag{2.2}$$

These two curvatures, κ_H and κ_G, are of great importance since they contain essential local information of the surface geometry. In fact, for any two points with the same set of mean and Gaussian curvatures, we can align their local neighborhoods through a 3D rigid motion with second order accuracy. An important fact about κ_G is that it is an *intrinsic* property of surfaces, i.e. it is determined by geodesic distances, or lengths measured when moving only on the surface[iii], as indicated by Gauss's Theorema Egregium. This is an important observation in mathematics and computation. For instance, as a direct consequence, we cannot draw a map of the Earth on a flat sheet without any distortion.

[iii] More formally, the metric on the surface.

An example about the importance of mean curvature is the minimal surfaces, which have $\kappa_H = 0$ at every point. The word "minimal" in the name is due to the fact the surface area, with some boundary constraints, is minimized if and only if the mean curvature is zero everywhere. This fact is related to *mean curvature flow*, which gradually moves the surface points according to the mean curvature normal, the normal with a magnitude determined by the mean curvature, as it only stops when mean curvatures are zero. This is related to how soap bubbles or membranes assume their characteristic shapes, and various shapes used in product design and architecture, as well as denoising process for geometric signals.

We can view the mean curvature normal as the gradient of surface area with respect to the point position, which leads to the following formula,

$$2\kappa_H \mathbf{N} = \lim_{A \to 0} \frac{\nabla A}{A} \tag{2.3}$$

where ∇ is the gradient notation denoting the rate of change with respect to the components of the coordinates of the chosen point, and A is a small area around the point on the surface and $\kappa_H \mathbf{N}$ is the mean curvature normal.

This definition of mean curvature suggests a similar geometric definition for Gaussian curvature, in which, Gaussian curvature can be defined by the same limiting process through the concept of the Gauss map. For a surface, the Gauss map is a mapping from a surface point to a point on the sphere, such that both points have the same normal direction, i.e. the surface point is mapped to its normal. Then, the Gauss curvature can be defined as the following limit,

$$\kappa_G = \lim_{A \to 0} \frac{A_G}{A} \tag{2.4}$$

where A_G is the area of the image of A on the unit sphere under the Gauss map. Instead of

using the formula $\kappa_G = \kappa_1\kappa_2$, the above definition is often the practical choice for implementing discrete Gaussian curvature. We provide the details in later sections.

2.1.2.5 First and Second Fundamental Forms

The mean curvature and Gauss curvature determine the local shapes, but this combination does not provide enough information on how to assemble these local pieces to a global shape of a larger piece. In other words, there are completely different surfaces sharing the same mean curvature and Gaussian curvature. Thus, to create representations for surfaces, it is necessary to introduce other quantities. The first and second fundamental forms are one set of surface functions that fit the task.

To make the discussion easier to follow, assume that the surface S is represented as a mapping $\mathbf{x}$ from a planar region with two coordinates (parameters) (u^1, u^2) to the 3D space $\mathbb{R}^3$. This description of a surface patch through the mapping $\mathbf{x}$ with two parameters is called a *parameterization*. Given the parameterization, we can formulate the surface as $\mathbf{x}(u^1, u^2) = (x^1(u^1, u^2), x^2(u^1, u^2), x^3(u^1, u^2))^T$, where x^i's are the coordinate functions. For simplicity, we use subscripts after comma to denote the partial derivatives.

$$\mathbf{x}_{,\alpha} = (\frac{\partial x^1}{\partial u_\alpha}, \frac{\partial x^2}{\partial u_\alpha}, \frac{\partial x^3}{\partial u_\alpha})^T, \quad \mathbf{x}_{,\alpha\beta} = (\frac{\partial^2 x^1}{\partial u_\alpha \partial u_\beta}, \frac{\partial^2 x^2}{\partial u_\alpha \partial u_\beta}, \frac{\partial^2 x^3}{\partial u_\alpha \partial u_\beta})^T,$$

where $\alpha, \beta \in \{1, 2\}$. The first fundamental form I is then defined in these notations as,

$$I = \begin{pmatrix} \langle \mathbf{x}_{,1}, \mathbf{x}_{,1} \rangle & \langle \mathbf{x}_{,1}, \mathbf{x}_{,2} \rangle \\ \langle \mathbf{x}_{,2}, \mathbf{x}_{,1} \rangle & \langle \mathbf{x}_{,2}, \mathbf{x}_{,2} \rangle \end{pmatrix} \tag{2.5}$$

Let $\mathbf{N}$ denotes the normal vector, obtained by $\mathbf{N} = \frac{\mathbf{x}_{,1} \times \mathbf{x}_{,2}}{\|\mathbf{x}_{,1} \times \mathbf{x}_{,2}\|}$. The second fundamental form II is defined as

$$\mathit{II} = \begin{pmatrix} \langle \mathbf{x}_{,11}, \mathbf{N} \rangle & \langle \mathbf{x}_{,12}, \mathbf{N} \rangle \\ \langle \mathbf{x}_{,21}, \mathbf{N} \rangle & \langle \mathbf{x}_{,22}, \mathbf{N} \rangle \end{pmatrix} \tag{2.6}$$

Intuitively speaking, the first fundamental form I measures all lengths on the surface, which can also be used to determine angles and areas, and the second fundamental form II measures how the normal vector is changing locally along different tangential directions. Given a 3D surface S with the (u^1, u^2) parameterization, we can obtain the fundamental forms straightforwardly.

They seem to encode the same information as the mean and Gaussian curvatures up to a reparameterization and a rotation within the tangent plane. In fact, for any point on the surface, we can always find a new mapping through an affine transformation such that I is turned into the identity matrix. Then, with a rotation, we can turn the symmetric matrix II into a diagonal matrix, with the principal curvatures on the diagonal. Alternative, we can directly compute mean and Gaussian curvature as the (half) trace and determinant of $I^{-1}\mathit{II}$, respectively. However, it is a fundamental result in differential geometry, that we cannot create a region around a point, within which I is always identity. Thus, the fundamental forms do provide more information for a region than the curvatures. By linking the curvature information with the parameterization, they provide a means to piece together the local shapes to create the global shape.

2.1.2.6 Surface Equations

We first introduce the notations used here, which follow common conventions in differential geometry. By using indices for the matrix elements, the first fundamental form (metric tensor) I can be written as $I_{\alpha\beta} = \langle \mathbf{x}_{,\alpha}, \mathbf{x}_{,\beta} \rangle$, and the matrix of second fundamental form II as $\mathit{II}_{\alpha\beta} = \langle \mathbf{x}_{,\alpha\beta}, \mathbf{N} \rangle$, where lowercase Greek letters are still used for indices $\in \{1, 2\}$, and an index after a comma in the subscript denotes partial derivative with respect to the parameter associated to that index.

In the following discussion, we assume Einstein's summation convention, in which indices that appear twice will be considered as dummy indices for summation, e.g.

$$\Gamma^{\gamma}_{\alpha\beta}\mathbf{x}_{,\gamma} = \sum_{\gamma=1}^{2} \Gamma^{\gamma}_{\alpha\beta}\mathbf{x}_{,\gamma}.$$

We also use superscripts and subscripts to denote different quantities related through I or its inverse. More precisely, inverse of the first fundamental is denoted by $I^{\alpha\beta}$, which is defined from $I_{\alpha\beta}$ by

$$I^{\alpha\gamma}I_{\gamma\beta} = \delta^{\alpha}_{\beta},$$

where δ^{α}_{β} is called Kronecker's symbol, which is equal to 1 when $\alpha = \beta$, and 0 otherwise. With $I^{\alpha\beta}$ defined, we can denote $I^{-1}I\!I$ by $I\!I^{\alpha}_{\beta}$, i.e.

$$I\!I^{\alpha}_{\beta} = I^{\alpha\gamma}I\!I_{\gamma\beta}.$$

Gauss's surface equations describe how the local quantities, the fundamental forms, can be used to construct the surface. These equations give expressions of the derivatives of the tangent vectors $\mathbf{x}_{,\alpha}$, which can be used to solve for the embedding $\mathbf{x}$.

It is actually a straightforward decomposition of $\mathbf{x}_{,\alpha\beta}$, a 3D vector when α and β are fixed, in the local frame formed by $\{\mathbf{x}_{,1}, \mathbf{x}_{,2}, \mathbf{N}\}$ as

$$\mathbf{x}_{,\alpha\beta} = \Gamma^{\gamma}_{\alpha\beta}\mathbf{x}_{,\gamma} + I\!I_{\alpha\beta}\mathbf{N}.$$

Note here that the decomposition coefficients $\Gamma^{\gamma}_{\alpha\beta}$ are called the Christoffel symbols, which turn

out to be functions of only I,

$$\Gamma^{\gamma}_{\alpha\beta} = \frac{1}{2} I^{\gamma\tau}(I_{\tau\alpha,\beta} + I_{\tau\beta,\alpha} - I_{\alpha\beta,\tau}).$$

Since $\mathbf{x}_{,\alpha\beta}$, $\mathbf{x}_{,\gamma}$ and $\mathbf{N}$ are all combinations of derivatives of $\mathbf{x}$, the above equations are a set of partial differential equations for $\mathbf{x}$ given I and $I\!I$.

Roughly speaking, to find a solution given a symmetric positive definite matrix function I and a symmetric matrix function $I\!I$, if we start from a seed point and a chosen pair of tangential directions $\mathbf{x}_{,1}$ and $\mathbf{x}_{,2}$, we can propagate these directions along any paths on the surface by integrating these equations. We can then construct an immersion by integrating $\mathbf{x}_{,1}$ and $\mathbf{x}_{,2}$.

I and $I\!I$ provide 6 numbers per point while the 3D embedding requires only 3 numbers (coordinates) per point, so there must be 3 equations per point that I and $I\!I$ must satisfy. In fact, they can be easily derived as follows. As the left-hand side of the above equation describes the gradient of $\mathbf{x}_{,\alpha}$, it has to be curl-free, which means

$$\begin{aligned} 0 = \mathbf{x}_{,\alpha\beta\gamma} - \mathbf{x}_{,\alpha\gamma\beta} &= (R^{\tau}_{\alpha\gamma\beta} - (I\!I^{\tau}_{\gamma} I\!I_{\alpha\beta} - I\!I^{\tau}_{\beta} I\!I_{\alpha\gamma}))\mathbf{x}_{,\tau} \\ &+ (\nabla_{\gamma} I\!I_{\alpha\beta} - \nabla_{\beta} I\!I_{\alpha\gamma})\mathbf{N}, \end{aligned} \tag{2.7}$$

where R denotes the Riemann curvature tensor determined by I,

$$R^{\tau}_{\alpha\gamma\beta} = \Gamma^{\tau}_{\alpha\beta,\gamma} - \Gamma^{\tau}_{\alpha\gamma,\beta} + \Gamma^{\tau}_{\gamma\mu}\Gamma^{\mu}_{\alpha\beta} - \Gamma^{\tau}_{\beta\mu}\Gamma^{\mu}_{\alpha\gamma};$$

and ∇ denotes the covariant derivative defined by I through Γ, in particular, for $I\!I$, it is

$$\nabla_{\gamma} I\!I_{\alpha\beta} = I\!I_{\alpha\beta,\gamma} - \Gamma^{\tau}_{\alpha\gamma} I\!I_{\tau\beta} - \Gamma^{\tau}_{\beta\gamma} I\!I_{\alpha\tau}.$$

The coefficient in front of $\mathbf{x}_{,\tau}$ must be 0,

$$R^{\tau}_{\alpha\gamma\beta} - (I\!I^{\tau}_{\gamma} I\!I_{\alpha\beta} - I\!I^{\tau}_{\beta} I\!I_{\alpha\gamma}) = 0,$$

which is called *Gauss's equation*. The coefficient in front of $\mathbf{N}$ must also be 0,

$$\nabla_{\gamma} I\!I_{\alpha\beta} - \nabla_{\beta} I\!I_{\alpha\gamma} = 0,$$

which are called the *Mainardi-Codazzi equations*. Note that on surfaces, $R^{\tau}_{\alpha\gamma\beta}$ has only one independent component $R^{1}{}_{212}$; thus Gauss's equation simply states that the Gaussian curvature $det(I^{-1} I\!I)$ is the intrinsic curvature.

The fundamental theorem of surfaces states that Gauss's equation and the Codazzi equations are not only a necessary condition for I and $I\!I$ to be induced by a local embedding, but also a sufficient one, as proven using the Frobenius theorem, which is explained further in Chapter 3.

In this section 2.1, we go over some necessary concepts very briefly. However, the above definitions can be easily found in many textbooks [41, 51] in great detail.

2.1.3 Differential Operators

We concentrate on calculus done in curved 2D for simplicity and relevance to our work. In planar differential calculus, three types of differential operators are essential. They are gradient, curl, and divergence. They can be combined to generate other useful operators, such as the Laplace operator, or Laplacian, which is crucial to spectral analysis, since the eigenfunctions of Laplacian form the basis for the usual spectral analysis.

The gradient operator on a scalar field f is defined by

$$(\nabla f)^{\alpha} = f_{,\alpha},$$

where we again use superscripts or subscripts before comma to index components, and those after comma to denote partial derivative with respect to that coordinates. It indicates the direction along which f increases fastest.

The curl operator on a vector field $\mathbf{v}$ is defined by

$$\nabla \times \mathbf{v} = v^{2}_{,1} - v^{1}_{,2}.$$

It measures local rotations contained in the vector field, for example, curl of velocity fields provides information on the strength of vortices.

The divergence operator on a vector field $\mathbf{v}$ is defined as

$$\nabla \cdot \mathbf{v} = v^{\alpha}_{,\alpha},$$

where we again use Einstein's summation notation.

The Laplace operator is then defined as the second order operator,

$$\triangle f = \nabla \cdot \nabla f.$$

On surfaces endowed with a first fundamental form I, the operators are modified as

$$
\begin{aligned}
(\nabla f)^{\alpha} &= I^{\alpha\beta} f_{,\beta} && (2.8)\\
\nabla \times \mathbf{v} &= \frac{1}{\sqrt{|I|}}[(I^{2}_{\alpha} v^{\alpha})_{,1} - (I^{1}_{\alpha} v^{\alpha})_{,2}] && (2.9)\\
\nabla \cdot \mathbf{v} &= \frac{1}{\sqrt{|I|}}(\sqrt{|I|} v^{\alpha})_{,\alpha}, && (2.10)\\
\Delta f &= \nabla \cdot \nabla f = \frac{1}{\sqrt{|I|}}(\sqrt{|I|} I^{\alpha\beta} f_{,\alpha})_{,\beta}, && (2.11)
\end{aligned}
$$

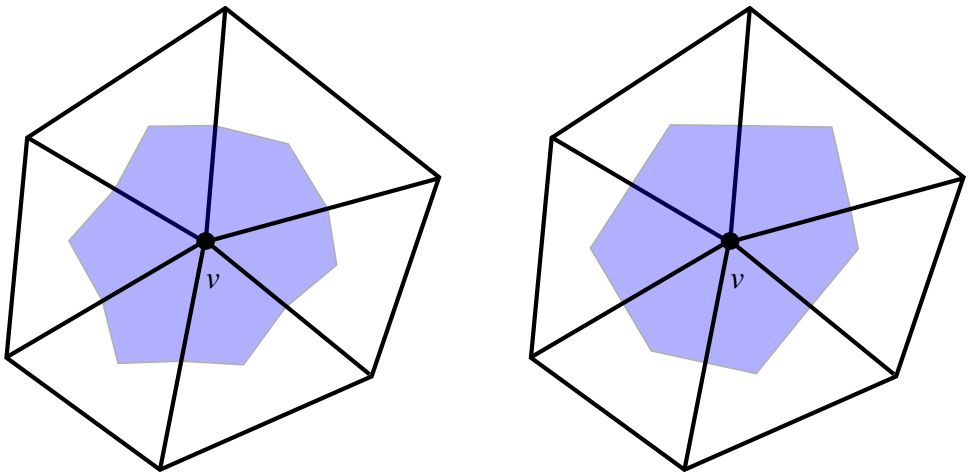
where $|I|$ denotes the determinant of the first fundamental form I. While it is easy to see that the above operators reduce to the planar counterparts when I is the identity matrix, these operators are seemingly coordinate-dependent. However, they are indeed coordinate-system (parameterization)-independent, which can be more obviously revealed with the language of differential forms discussed in the discrete calculus section.

2.2 Discrete Curvature Estimates for Triangle Meshes

As example concepts in discrete differential geometry, which are also closely related to the discrete fundamental forms that we introduce in Chapter 3, we show how the continuous definitions on curvatures can be mirrored in the discrete settings, in particular, on triangle meshes. The principles of taking integral (or spatial average) and defining quantities on carefully chosen mesh entities used in the definitions of curvatures are explained here, and they also apply to our definition of discrete fundamental forms.

As mentioned earlier, triangle meshes are widely used in computer graphics to approximate surfaces piecewise-linearly. However, properties derived from taking differentials in continuous theory become nontrivial to define in such piecewise flat structure. For example, normal vectors and

curvatures cannot be defined everywhere directly as in the continuous case. Nevertheless, these basic geometric attributes are crucially important and ubiquitously needed in numerous applications, such as mesh simplification, surface modeling, mesh editing and parametrization [44, 2, 113].



(a) (b)

Figure 2.5: Spatial average region (a) and Voronoi region (b) around a point v.

The work from Meyer et al. [75] gives definitions on some discrete differential operators for triangle meshes. The idea behind their discrete definitions is *spatial average* (as shown in Figure 2.5a). In order to define a property on a vertex[iv], we can specify a local region around it, which is influenced by this vertex. Thus, the property value stored on this vertex represents an average of the property on all points within this region. For a vertex v, denoting the local region as Ω_v, we can formally express the spatial average as

$$f|_v = \frac{1}{A_\Omega} \iint_{p \in \Omega} f|_p dA, \tag{2.12}$$

where f is the value of some property of interest, and A_Ω is the area of region Ω_v. If the local region is chosen consistently for each vertex, increasing sampling density with a high-quality mesh

[iv] We use the term "vertex" for corner points of the polyhedron representing the discrete surface.

will eventually lead to the property converging to the continuous definition under mild assumptions on the smoothness of f.

The way of defining the spatial averaging region around a vertex is non-unique. For Finite Element with linear basis functions or other piecewise linear methods, the spatial average region is sometimes chosen to be the region bounded by a loop going through the midpoints of every edges in one ring, with those midpoints connected through a certain "center" point chosen from each triangle (see Figure 2.5b). The most commonly used two such regions are the Voronoi region, which corresponds to choosing circumcenters as the triangle centers, and the barycenter region, which corresponds to choosing barycenters as the triangle centers.

2.2.1 Discrete Mean Curvature Normal

Recall that we can consider the mean curvature from area limit as equation (2.3). For a vertex v_i on a triangle mesh, the discrete mean curvature H_i can be defined through

$$2H_i\mathbf{N}_i = \frac{1}{A_\Omega} \sum_{T_j \in Ring(v_i)} \nabla A(T_j) \tag{2.13}$$

where, $H_i\mathbf{N}_i$ corresponds to the mean curvature normal at v_i; $Ring(v_i)$ is the *one-ring* of v_i, the region formed by the triangles incident to v_i; $A(T_j)$ is the area of the intersection of triangle j and the spatial region Ω; and $A_\Omega = \sum_{T_j \in Ring(v_i)} A(T_j)$. Note that $\nabla A(T_j)$ becomes trivial to compute in the discrete setting, which is just the normal direction of the triangle j.

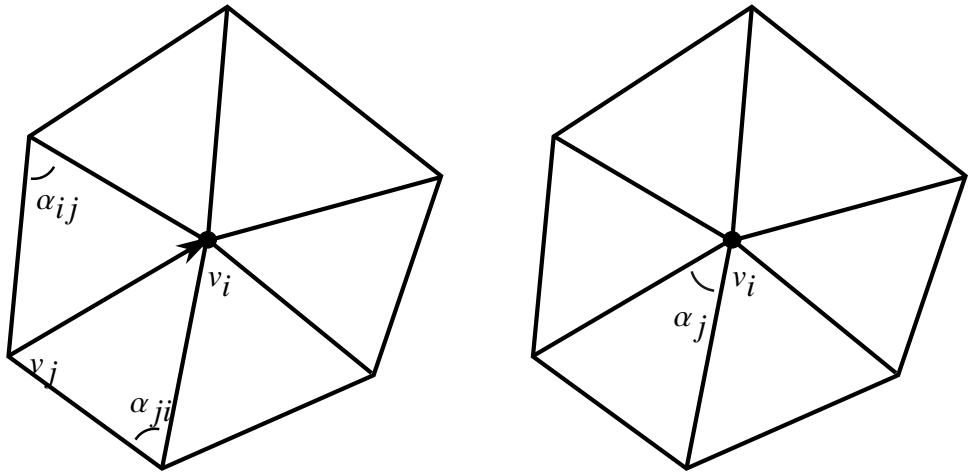
We can see that the discrete mean curvature normal shares the same form of the integral version of in the continuous case Eq. 2.3,

$$\int\int_A 2\kappa_H\mathbf{N} = \nabla A.$$

Expanding the gradient of triangle angles leads the following equation, known as the *cotan-formula* applied to the vertex coordinates:

$$2H_i\mathbf{N}_i = \frac{1}{A_\Omega} \sum_{v \in Ring(v)} (\cot \alpha_{ij} + \cot \alpha_{ji})(v_i - v_j) \tag{2.14}$$

where the α_{ij} and α_{ji} are the opposite angles of edge e_{ij} within its two adjacent triangles as showed in Figure 2.6a. This formula derived from area minimization coincides with the one derived from differential approach when the differential operator involved is discretized as specified in the next section, another evidence of the structure-preserving property of the spatial averaging approach.



(a) (b)

Figure 2.6: Discretization of mean and Gaussian curvatures at vertex v_i.

2.2.2 Discrete Gaussian Curvature

The definition of Gaussian curvature in the continuous case is involved with the second order derivative of spatial coordinates as with the definition of mean curvature. Thus, we use the spatial averaging approach as in the mean curvature case. We start with the alternative definition involving the area ratios between a neighborhood of a point and its image under the Gauss map.

For a piecewise flat surface, the Gauss map image of a region covers a non-zero area on the sphere only when it contains a vertex. Otherwise, it is only a point or a curve without interior on the sphere. Thus, a commonly used geometric way to define Gaussian curvature for a vertex v_i is

$$G_i = 2\pi - \sum_{T_j \in Ring(v_i)} \alpha_j, \tag{2.15}$$

where α_j is the tip angle of triangle T_j in the one-ring of vertex i, as illustrated in Figure 2.6b. If we sum up the Gauss curvature at each vertex on a triangle mesh, we get a discrete version of the Gauss-Bonnet theorem, which says that the integral of Gauss curvature is $2\pi\chi$, where χ is the Euler characteristic, a topological invariant that can be evaluated based on an alternating sum of the number of cells of each dimension. For a closed triangle mesh M, assuming V, E and F denote the number of vertices, edges and faces , the Euler characteristic of M is $V - E + F$. The sum of all Gauss curvatures are $2\pi V - \pi F$[v], which means $2\pi(V - \frac{1}{2}F) = 2\pi(V - E + F) = 2\pi\chi$[vi].

The above definition is in fact just the integral of Gaussian curvature over a region that contains a single vertex. Thus, if we take the spatial average region into account, the following equation is used to get the average value

$$G_i = \frac{1}{A_\Omega}(2\pi - \sum_j \alpha_j), \tag{2.16}$$

which is precise the Gauss map image area divided by the original area. Our discrete definitions in Chapter 3 on the fundamental forms are consistent with this popular discrete Gauss curvature, in the sense that they satisfy a version of Gauss's equation linking I and K.

[v] Each angle from each face is subtracted exactly once.

[vi] For a closed surface triangle mesh, each edge is shared by exactly two triangles, which implies$2E = 3F$.

2.3 Discrete Exterior Calculus

A systematic treatment on differential operators (e.g., gradient, divergence and Laplace operator) is rather straightforward through *discrete exterior calculus* (DEC). DEC mimics the smooth exterior calculus, which provides common operators, including the differential operators, on entities called *differential forms*. With a first fundamental form, differential forms can be used to represent scalar fields and vector fields; and exterior calculus provides all necessary operators for vector calculus on surfaces and volumes.

The most important operator in exterior calculus is the exterior derivative, which is a unifying concept for various differential operators in the study of geometry, topology and physics. It has a long history, involving mathematicians such as, Hermann Grassmann, Sophus Lie, Élie Cartan, W. V. D. Hodge and many others, although the systematic treatment of discrete exterior calculus (DEC) was devised only a decade ago. Due to the needs in numerous applications from computational methods (e.g., elasticity, plasticity, electromagnetics, and fluids dynamics), computer vision and computer graphics, the applications and extensions of DEC have been intensely explored.

2.3.1 Smooth Exterior Calculus

Before introducing DEC, we briefly explain the terminology used in smooth exterior calculus. Again, the discussion is mainly on the intuition behind each concept. For a comprehensive introduction to exterior calculus, the interested readers are referred to Frankel [38].

2.3.1.1 Manifold

Informally, an n-manifold is a space that locally looks like an n-dimensional linear space. For example, the surface of the Earth can be modeled as a 2-manifold. Imagine that standing on

the ground, one sees the ground as approximately a flat plane (2-dimensional linear space), even though the Earth globally cannot be flattened in maps without first being decomposed into (potentially overlapping) pieces.

A manifold can be described by such pieces through continuous maps called *charts*, which describe the inverses of continuous parameterizations of such pieces of the manifold defined on subsets of $\mathbb{R}^n$. The linear space thus provides local coordinate systems for the manifold. Most manifolds (e.g., a sphere) must be described by more than one charts. The collection of charts that covers the entire manifold is called an *atlas*. A *differentiable* manifold is the type of manifold with an atlas containing "compatible" charts. Roughly speaking, the word "compatible" means that, in the intersection region of three charts (e.g. U, V, and W), the change of coordinates are differentiable, and the change from U to W is the same as the compound of the change from U to V and V to W. Unless specified otherwise, the manifolds mentioned in this dissertation are differentiable.

In this dissertation, the word "surface" implies a 2-manifold, and "volume" a 3-manifold. For simplicity, the concepts to be introduced are based on 2-manifolds in following discussion.

2.3.1.2 Vector Field

As mentioned in 2.1.2, for each point on a curve, there is a tangent vector going through it (as shown in 2.1); for each point on a surface, there is a tangent plane passing through it (as shown in 2.3). For high-dimensional manifolds, the similar concept is generalized as a *tangent space* to describe the space composed by all tangents passing through a point. It allows the definition of *vector field* on manifolds. Informally, a vector field is seen as a continuous assignment of a tangent vector, belonging to the tangent space, to each point of the manifold.

2.3.1.3 Differential Forms

The concept of a *k-differential form* provides a unified way to represent the integrands for k dimensional integrations on k-dimensional submanifolds (subsets of manifolds inheriting the manifold structures). It simplifies the representation of integrands, especially when they are defined with multiple variables in differential geometry and tensor analysis. More precisely, a k-form (ω) is the integrand of a definite integral on a k-dimensional region (Ω).

$$\int_{\Omega} \omega \tag{2.17}$$

For example, a scalar field (regular function) $s(x, y)$ can be considered a 0-form (integrated at a point) in a plane or a chart of a surface with coordinates (x, y); $f(x)dx$ in the definite integral $\int_a^b f(x)dx$ on the interval $[a, b]$ along the real line is a 1-form; more generally, in $\int_C F(x, y, z)dx + G(x, y, z)dy + H(x, y, z)dz$ integrated along a 3D curve, $F(x, y, z)dx + G(x, y, z)dy + H(x, y, z)dz$ is also a 1-form; similarly, the integrand in the integral $\iint_S F(x, y, z)dxdy + G(x, y, z)dydz + H(x, y, z)dzdx$ is a 2-form.

In 2D, we can see that 0-forms have one component, 1-forms have two components, and 2-forms have one component. When one changes the coordinate-based representation from one chart (parameterization) to another, 0-forms remain the same, so they represent functions (scalar fields); 1-forms v_α change in a way different from vector fields v^α, but the two can always be related through I, i.e. $v_\alpha = I^{\alpha\beta}v_\beta$; 2-forms change similarly to functions multiplied by $\sqrt{|I|}$, square root of the determinant of the first fundamental form. Thus, given a first fundamental form I, we can use both 0-forms and 2-forms to represent scalar fields, and 1-forms to represent vector fields.

2.3.1.4 Exterior Derivative

Given a k-form ω, the *exterior derivative* produces a $(k+1)$-form when applied on ω, such that

$$\int_{\Omega} d\omega = \int_{\partial\Omega} \omega, \tag{2.18}$$

where Ω is a $(k+1)$-submanifold, and $\partial\Omega$ is its k-D boundary. This definition can be shown as equivalent to a more complicated form expressed in coordinates, through the Stokes' theorem.

In a sense we can see d as an overloaded operator, and use d_k to explicitly denote the exterior derivative applied to k-forms. When it is clear from the context, we drop the subscript k in the following text.

Furthermore, this operator d corresponds to the gradient ∇ when applied to 0-forms representing the functions to get 1-forms representing the resulting vector fields, and the curl operator $\nabla\times$ when applied to 1-forms.

2.3.1.5 Hodge star

On an n-dimensional manifold, the *Hodge dual* or *Hodge star* operator, denoted as $\star_k$, converts a k-form into a $(n-k)$-form. We omit the mathematical definition here, and give only the 2D case. When $n = 2$, $\star$ applied to a 0-form as simply a multiplication of $\sqrt{|I|}$, to a 1-form as rotation of the corresponding vector field by 90°, and to a 2-form as division by $\sqrt{|I|}$.

As with d, $\star$ can be seen as an overloaded operator, and we omit the subscript when the context provides enough information about which type of differential forms that $\star$ is applied to.

Together with d, we can assemble the operator divergence $\nabla\cdot$ as $d\star_1$.

2.3.2 Discretization

For computation purposes, we need to find the discrete counterparts of the above concepts on smooth theory. Following the main principle of DDG, we must seek the preservation of the important structures in this calculus on surface. As a counter example, a regular finite-difference-based approach on 0-forms or 1-forms would not keep basic theorems (e.g., Stokes' theorem) even when the manifold is discretized to a regular grid.

We only present the most relevant concepts briefly here. For a detailed discourse on DEC, interested readers are referred to Hirani [45] and Desbrun et al. [29]. We loosely follow the notations in Desbrun et al. [29] to explain the discretization of the entities involved one by one: first, the discrete domain; second, fields defined on the domains; third, basic operators; and finally, the Laplacian operator.

2.3.2.1 Discrete Manifold and Simplicial complexes

As the domain on which exterior calculus is performed, we introduce the discretization of a manifold in this section. In practice, polygonal and polyhedral meshes are commonly used in applications. Theoretically, triangle and tetrahedral meshes are examples of *simplicial complex*, which is a collection of simple cells called *simplices*. A *k-simplex* is the k-dimensional element in the discretization of manifold. Roughly speaking, a k-simplex is the convex hull of $k + 1$ points in a generic position, i.e. when the convex hull does not degenerate to an m-dimensional set, with $m < k$.

As shown in the following figure, 0-simplices are vertices, 1-simplices are edges, 2-simplices are triangles, and 3-simplices are tetrahedra.

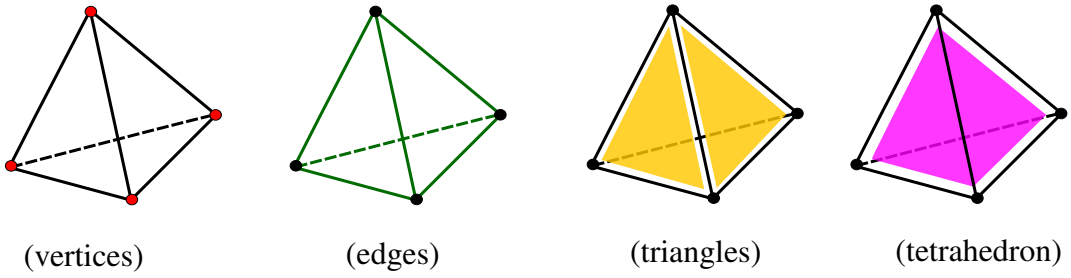


Figure 2.7: Examples of k-simplices.

Any k vertices from the $(k+1)$ [vii] vertices form a $(k-1)$-simplex. We call those $(k-1)$-simplices the $(k-1)$-*faces* of the k-simplex. For convenience, to allow faces to be defined for 0-simplices, the empty set $\emptyset$ is regarded as the (-1)-simplex.

A *simplicial complex* $\mathcal{K}$ is a collection of simplices under the following conditions:

1. Every face of a simplex from $\mathcal{K}$ is also in $\mathcal{K}$.

2. The intersection of any two simplices is a face of both simplices.

The second condition eliminates some invalid cases. For instance, as shown in Figure 2.8, the intersection of two 2-simplices is not a face of either of the 2-simplex.

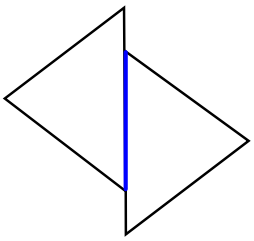
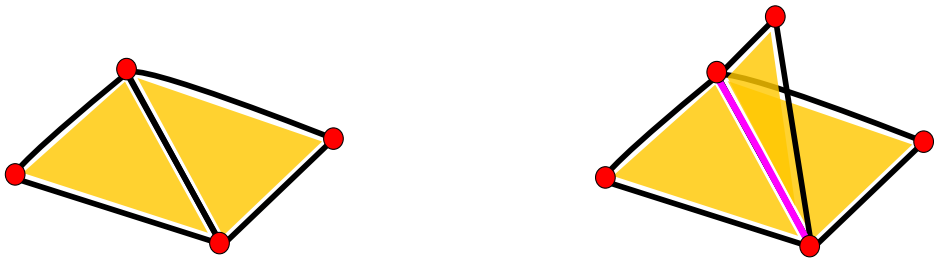


Figure 2.8: Example of invalid case for being 2-simplex.

Triangle and tetrahedral meshes are special cases of simplicial complexes, whose highest cell dimensions are 2 and 3, respectively.

[vii] Obviously, there are $k+1$ different ways.

Simplicial complexes can be used to represent manifolds. However, for a simplicial complex to be a *discrete manifold*, every point must have neighborhoods that are similar to n-D linear spaces. Specifically, an n-dimensional discrete manifold $\mathcal{M}$ is an n-D simplicial complex satisfying following condition: for each simplex, the union of all the incident n-simplices is topologically an n-D ball, i.e. it can be continuously deformed into an n-D ball. In practice, it suffices to check if the condition holds for each vertex.



(Discrete Manifold) (Not a discrete manifold)

Figure 2.9: Examples of discrete manifold and non-manifold cell complexes.

The Figure 2.9 shows two valid 2-D simplicial complexes. The left picture shows a discrete 2-dimensional manifold, while the purple edge on the right picture destroys the property of "locally linear", because all incident 2-simplices do not form a topological disk (2-D ball).

2.3.2.2 Discrete Differential Forms

As introduced in previous sections, discrete differential k-forms on surface are discretized as their integrals on k-D cells, which are the k-simplices in a simplicial complex representing a discrete n-manifold. In particular, on *orientable* surfaces, i.e. surfaces with a specific ordering of vertices for each triangle, once we select an orientation for each edge, we can represent the forms as follows:

- 0**-form**: scalar values on vertices
- 1**-form**: scalar values on edges

- 2**-form**: scalar values on facets

For a discrete surface M, let n_0, n_1 and n_2 be the number of vertices, edge and facets, respectively. Then the discrete k-forms are stored as vectors with size n_k.

2.3.2.3 Discrete Exterior Derivative

In the continuous setting the exterior derivative operator turns k-forms, linearly, into $(k+1)$-forms. Naturally, the exterior derivative operators are discretized as matrices, since matrices represent the linear transformations between two vector spaces. As d can be defined through Stokes' theorem, and a $(k+1)$-form $d\omega$ is stored as one value per $(k+1)$-simplex σ, we only need the values summed up over all the values for the ω on the k-D faces in $\partial\sigma$. Thus, the three discrete exterior derivative operators for 2-dimensional discrete manifold are merely incidence matrices with entries indicating the sign of mutual orientation: the exterior derivative d_k ($k \in \{0, 1\}$) for k-forms is a $n_{k+1} \times n_k$ matrix D, such that

$$D_{ij} = \begin{cases} 1 & \text{if } i\text{-th } (k+1)\text{-simplex is incident to } j\text{-th } k\text{-simplex with consistent orientation;} \\ -1 & \text{if } i\text{-th } (k+1)\text{-simplex is incident to } j\text{-th } k\text{-simplex with opposite orientation;} \\ 0 & \text{otherwise.} \end{cases}$$

2.3.2.4 Discrete Hodge Star

The discretization of more accurate Hodge star operators is still an active topic in computer graphics [78], finite element analysis, and other areas. A convenient way of visualizing the Hodge dual is to use a *dual mesh*. For simplicity, we use Voronoi diagrams as dual meshes. In 2D, such a dual mesh provides: a 2-D cell, the Voronoi region, for each vertex in the primal mesh (the original triangle mesh); a 1-D cell, Voronoi edge between two adjacent Voronoi regions of the end vertices,

for each edge in the primal mesh; and a 0-D cell, the intersection of three Voronoi regions for the three vertices, for each triangle in the primal mesh.

The discretization of the Hodge star operator relies on this duality. The primal k-dimensional Hodge star transfers a k-form (on the primal mesh), in a *linear* manner, to an $(n-k)$-form (on the dual mesh). Given a surface M with n_0 vertices, n_1 edges and n_2 facets, we can represent the Hodge star as an $n_k \times n_k$ matrix, since each primal k-simplex has exactly one corresponding dual $(n-k)$-simplex in the dual mesh. Such matrix representations are not unique, with complexity dependent on its accuracy (error in terms of the mesh resolution). For efficiency, in this dissertation, we use symmetric, positive definite and diagonal matrices to represent the Hodge stars. Those properties bring computation efficiency and numerical stability for our applications. Similarly to the discrete exterior derivative, we have three discrete Hodge star operators for 2-dimensional discrete manifold. The Hodge star mapping k-forms on the primal meshes to $n-k$-forms on the dual mesh can be represented as the matrix

$$S_{ij} = \delta^i_j \, |\star\sigma_i|/|\sigma_i|,$$

where δ^i_j is the Kronecker symbol, which is equal to 1 when $i = j$ and 0 otherwise, $|\sigma|$ is the size of a k-simplex σ (1 for vertex, length for edge, and area for polygon), and $\star\sigma$ is the dual $n-k$-cell. The Hodge star from k-forms on the dual mesh to the primal mesh, is simply the inverse of the corresponding primal Hodge star.

2.3.2.5 Discrete Laplace Operator from DEC

Now, we can re-examine Laplace operators from the perspective of DEC. For any k-form, it can be defined in general as

$$\triangle = d^* d + d d^*, \tag{2.19}$$

where $d_k^* = (-1)^k \star^{-1} d \star$.

In particular, for a function, f,

$$(\mathbf{L}f)_{p_i} = \sum_{p_j} ((\cot \alpha_{ij} + \cot \alpha_{ji})(f_{p_i} - f_{p_j})). \tag{2.20}$$

We can observe that this cotan formula coincides with the coefficients in the mean curvature estimate, preserving this important relation in the discrete setting, showing the benefits of proper discretization.

However, as concluded in Wardetzky et al. [111], different discrete strategies lead to different behaviors of the Laplacian in applications. It can be theoretically proved that there is "no free lunch", i.e., it is impossible to come up with a "perfect" discrete Laplace operator which preserves all the properties of a continuous Laplace operator.

In summary, we introduce in this chapter the basic terminology in the relevant parts of differential geometry, including curvature, fundamental forms, and simple calculus, as well as the principled discretization methods that we use throughout the dissertation: meshes, differential forms, and operators in DEC.

Chapter 3

Rigid-motion-invariant Discrete Surface Representation

Local rigid-motion-invariant quantities, such as the aforementioned curvatures and fundamental forms, often serve as a starting point for differential geometry of surfaces. According to the fundamental theorem of surfaces, the fundamental forms can even be used to represent the surface, since they determine the global shape. In the discrete setting, such representations are also of great interest in various geometry processing contexts. For geometric modeling, local encoding of geometric details facilitates intuitive surface manipulation and deformation. For the simulation of elastic thin-shell objects, the strain energy must only depend on invariants of rigid motions to ensure preservation of momenta. Even parameterization, geometric texture synthesis, shape analysis, or 3D model registration can potentially benefit from an efficient rotation-invariant description of the shape.

An obvious choice for such a representation for surface meshes would be the array of edge lengths and dihedral angles, which uses approximately twice as much storage space as the orig-

inal vertex coordinates, but is indeed rotation-invariant and local. Moreover, given an arbitrary parameterization, they can be used to construct the first and second fundamental forms, which describe the metric of the surface and the bending of normals respectively. However, constructing a surface embedding (or immersion) in a linear fashion from an *arbitrary* set of edge lengths and dihedral angles is far from straightforward—except when the set of values are known to come from an existing embedding, in which case a sequential reconstruction of the shape can be performed by determining the coordinates of the vertices one-by-one based on the edge lengths and dihedral angles, through traversing the mesh (e.g., in breadth-first search order) starting from an arbitrary seed vertex.

3.1 Discrete Fundamental Theorem of Surfaces

To mirror the fundamental forms from the smooth theories (mentioned in Section 2.1), we [110] propose to encode the local geometric information in edges instead of vertices. The edge lengths and dihedral angles look like an obvious choice for surface representation. In fact, Fröhlich and Botsch [39] have used edge lengths and dihedral angles in their mesh deformation method, which relies on *non-linear* minimization of a physics-based energy. However, it is not straightforward to reconstruct a 3D surface from an *arbitrary* edge lengths and dihedral angles in a *linear* manner. As mentioned above, the reconstruction is only trivial if the edge lengths and dihedral angles are known to come from an existing embedding. To introduce our [110] discrete fundamental forms, we employ a convenient mathematical tool called lift.

3.1.1 Lift of an Immersion.

The lift $F(u^1, u^2)$ of an immersion $\mathbf{x}(u^1, u^2)$ from the 3D Euclidean space to the 6D rigid transformation (or Euclidean motion) group $E(3)$ is expressed in the familiar 4×4-matrix form for homogeneous coordinates

$$F = \begin{pmatrix} f & \mathbf{x} \\ 0 & 1 \end{pmatrix}, \tag{3.1}$$

where $f = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ is a rotation matrix representing a local frame at $\mathbf{x}$. We further restrict the third basis vector $\mathbf{b}_3$ to be $\mathbf{N}$, and thus $(\mathbf{b}_1, \mathbf{b}_2)$ is an orthonormal frame of the tangent plane. This moving frame that we attach to each point on surfaces is similar to the Frenet-Serret frame for curves. The matrix 1-form dF provides the differential properties of the surface, including the tangential directions ($d\mathbf{x}$) and how the frames are rotating (df). To find the differential invariants, we describe these quantities in the frame with the axes aligned to f and the origin located at $\mathbf{x}$, i.e., we use $F^{-1}dF$. [i]

$$\Omega = F^{-1}dF = \begin{pmatrix} f^T & -f^T\mathbf{x} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} df & d\mathbf{x} \\ 0 & 0 \end{pmatrix} \tag{3.2}$$

$$= \begin{pmatrix} f^T df & f^T d\mathbf{x} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\omega_1^2 & -\omega_1^3 & \omega^1 \\ \omega_1^2 & 0 & -\omega_2^3 & \omega^2 \\ \omega_1^3 & \omega_2^3 & 0 & \omega^3 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.3}$$

[i] Technically, this turns out to be the pullback of a canonical Lie-algebra-valued 1-form on $E(3)$, the Maurer-Cartan form Ω. Roughly speaking, it just identifies the tangent space at any F to the tangent space at the identity of the group through the group action (i.e., matrix multiplication) of F^{-1}; refer to Ivey and Landsberg [51] for the mathematical definitions. In this context, the pullback of Ω onto the surface simply means that we restrict the application of the 1-forms to only tangent vectors of the surface. We omit the pullback notation below for simplicity.

which can be seen as a matrix with each entry being a 1-form. $\Omega_x = (\omega^1, \omega^2, \omega^3)^T$ denotes the change of $\mathbf{x}$ in the local frame. We have $\omega^3 = 0$ on the surface, since an infinitesimal displacement on the surface does not contain any normal component. ω^1 and ω^2 can be used as an orthonormal basis for 1-forms on the surface, which implies that they provide information about I; more precisely, $I = \omega^1 \otimes \omega^1 + \omega^2 \otimes \omega^2$. The upper left 3×3-block Ω_R denotes the rotation of f under infinitesimal motions on the surface, and is antisymmetric because f is orthonormal. ω_1^2 is exactly the metric connection form of I in the given orthonormal local frame field. We also have $\omega_\alpha^3 = I\!I_{\alpha\beta}\omega^\beta$, describing the rotation of the normal vector. Thus, Ω encapsulates all the information in I and $I\!I$, and prescribes the differential dF given F at any given point. Since $dF = F\Omega$,

$$d\mathbf{x} = f\,\Omega_x = \omega^1\mathbf{b}_1 + \omega^2\mathbf{b}_2 \tag{3.4}$$

$$df = f\,\Omega_R = f\begin{pmatrix} 0 & -\omega_1^2 & -\omega_1^3 \\ \omega_1^2 & 0 & -\omega_2^3 \\ \omega_1^3 & \omega_2^3 & 0 \end{pmatrix}. \tag{3.5}$$

The main difference between these equations and Gauss's surface equations is that here we choose an orthonormal frame field instead of the frame induced by the parameterization. Therefore, up to a change of frames, the integrability conditions are exactly the same as Gauss's equation and the Codazzi equations. However, by introducing the intermediate variables f, the integrability conditions in this form can be constructed straightforwardly. Briefly speaking, our discrete fundamental theorem of surface examines the restrictions on the values that Ω can assume to make dF integrable, and our discrete reconstruction deals with the problem of finding a least-squares solution minimizing the difference between $F^{-1}dF$ and the prescribed Ω, when the system is not exactly

integrable.

3.1.2 Discrete Fundamental Forms from Edge Lengths and Dihedral Angles

Given a fixed connectivity of the surface mesh M with vertex set V, edge set E, and triangle set T (which corresponds to fixed topology in the continuous case), we define the *discrete first fundamental form I* to be the assignment of edge lengths $\{l_e\}$ and the *discrete second fundamental form II* to be the assignment of angles between face normals $\{\phi_e\}$ (or ϕ_{ij}, if e is adjacent to triangles i and j). We will also sometimes refer to ϕ_e as the dihedral angle, although technically, the dihedral angle is $\pi - \phi_e$. Both forms are stored on edges; they are naturally invariant under rigid-body transformations, and independent of the parameterization.

Given an arbitrary parameterization, edge lengths can certainly be used to construct a piecewise-constant first fundamental form matrix. The dihedral angles are often used to compute mean curvatures at edges locally. However, together with the connectivity of the mesh, they provide more information than just the mean curvature, since they denote the change of the normal across the edges. In fact, given the change of normal in the three directions from each face center to its three adjacent faces, one can construct a unique second fundamental form matrix (represented by 3 numbers in any given parameterization) per face. Conversely, the edge lengths and dihedral angles can be computed from the fundamental form matrix per face. Note that no existing embedding is required in the process.

3.1.2.1 Local frames

Given the discrete first fundamental form I, we can define a 2D local orthonormal frame $(\mathbf{t}, \mathbf{t}^{\perp})$ in each triangle T. This is uniquely determined by the edges and an arbitrary rotation from, e.g., the first edge of the triangle. We show later that all our derivations and algorithms are independent of

this rotation. Each edge (seen as a 2D vector) can then be expressed as a linear combination of its local frame vectors, $\mathbf{x}_n - \mathbf{x}_m = a^1_{mn,T}\mathbf{t} + a^2_{mn,T}\mathbf{t}^\perp$, if the edge points from vertex m to vertex n, and $\mathbf{x}_m$ is the 2D coordinates of vertex m. Here, $a^\alpha_{mn,T}$'s are specifying a piecewise constant ω^1 and ω^2 given the choice of the 2D frame.

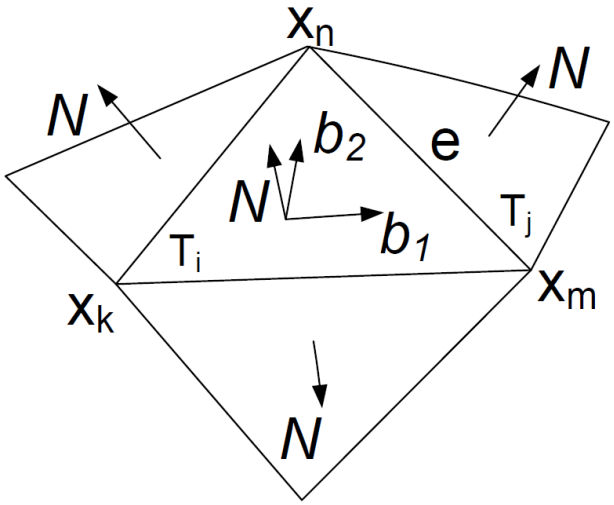


Figure 3.1: An example of local frame.

If there is an immersion, we can identify the 2D frame with $(\mathbf{b}_1, \mathbf{b}_2)$, and the inclusion map indicates $\mathbf{x}_n - \mathbf{x}_m = a^1_{mn,T}\mathbf{b}_1 + a^2_{mn,T}\mathbf{b}_2$ (see figure 3.1).

3.1.2.2 Transition rotation

In the continuous setting, a proper integration of the rotation part of $dF = F\Omega$ along a dual path from T_i to T_j would produce $f_j - f_i = f_i R_{ij} - f_i$, where R_{ij} is a rotation determined by Ω independent of the path if Ω comes from an integrable pair of first and second fundamental forms. In the discrete setting, we can use the *transition rotation* R_{ij} defined on each edge to express the rotation part of the discrete Ω as $R_{ij} - Id$. A discrete transition rotation that is completely determined by I and II (and consistent with an immersion when it exists) can be constructed as follows. A rotation angle that would align the first axis in the 2D orthonormal frame in a triangle T to one of its edge

e is denoted as $\theta_{T\to e}$. We define R_{ij} to be the 3D rotation matrix $R_z(\theta_{T\to e})R_x(\phi_e)R_z(-\theta_{T'\to e})$, where $R_u(\theta)$ represents the rotation of axis u and angle θ, e is the shared edge with triangle T_i to its left side, and ϕ_e changes sign if the edge orientation is reversed. Basically we align the first frame axis of T_i to the common edge, rotate around the edge by the dihedral angle, and then rotate within the next face to fit the local frame in T_j.

Given an immersion with a lift F, we also have $R_{ij} = f_i^{-1} f_j$. The rotation angle $\theta_{T\to e}$ in the tangent plane determines the discrete connection for tangential fields [21], corresponding to ω_1^2 in the continuous setting, while ϕ_e corresponds to ω_1^3 and ω_2^3, which contain the information of $I\!I$.

3.1.3 Discrete Surface Equations

In the last section, we provide the discretization of fundamental forms. It is natural to examine the other components of the continuous theory: the discretization of certain conditions that should be satisfied to recover the surface from fundamental forms.

3.1.3.1 Local Discrete Fundamental Theorem of Surfaces

Since we put f and $\mathbf{x}$ at different locations (faces and vertices, resp.), we need to process the two components of dF also at appropriate locations.

The frames of the lift must satisfy the following *frame equations*:

$$\forall e_{ij} \in E, \quad f_j - f_i - f_i(R_{ij} - Id) = f_j - f_i R_{ij} = 0 \tag{3.6}$$

For any triangle T containing vertex m and vertex n, we have the *edge equations* as follows:

$$\mathbf{x}_n - \mathbf{x}_m - (a^1_{mn,T}\mathbf{b}_{1,T} + a^2_{mn,T}\mathbf{b}_{2,T}) = 0. \tag{3.7}$$

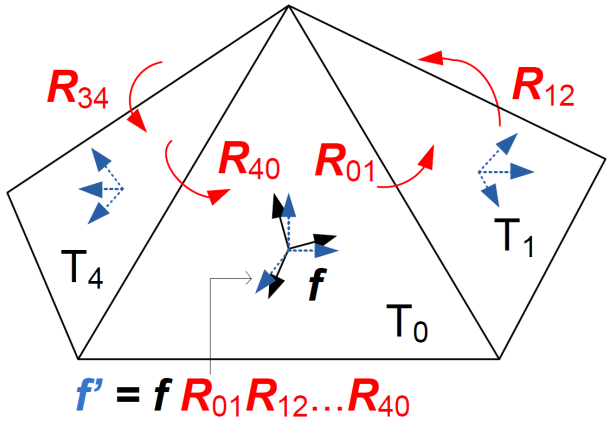


Figure 3.2: Transition rotation matrices.

3.1.3.2 Local discrete fundamental theorem of surfaces

Given a surface mesh with fixed connectivity and disk-like topology, if the transition rotation matrices $\{R_{ij}\}$ computed from a set of edge lengths (satisfying the triangle inequalities) and a set of dihedral angles satisfy the following condition for a counterclockwise loop of n triangles T_i around each interior vertex

$$\Pi_{i\in 0,\ldots,n-1} R_{i,(i+1) \bmod n} - Id = 0, \tag{3.8}$$

then there exists an immersion in 3D Euclidean space with these edge lengths and dihedral angles as its first and second fundamental forms.

Proof. The condition (Eq. 3.8) means that f can be updated by repeatedly applying transition rotations ($f_j = f_i R_{ij}$) along any minimal dual cycle (formed by the dual edges of the one-ring of a vertex, see Figure 3.2), and the result is the same as the starting frame. We prove next that integrating f using Eq. 3.6 along two *arbitrary* dual paths from face i to face j will produce the same frame f_j from the same f_i. Without loss of generality, we restrict the paths to be non-intersecting and without self-intersection. First notice that due to the trivial topology we assume

for now, such two paths form a counterclockwise boundary loop around a simply connected region if we reverse one of the paths. Second, we use induction: As shown in Figure 3.3, if a dual loop encloses a single interior vertex, the accumulated rotation around it will be identity; Assume that the conclusion is true for a dual loop enclosing k vertices; For a dual loop enclosing $k+1$ vertices, we can always find one vertex adjacent to the dual loop such that removing it will leave the remaining region simply connected; Denote the common segment of the dual loop around the removed vertex and the original loop by $S_{i\to j}$, the other part of the original loop by $P_{j\to i}$, and the other part of the one-ring dual loop by $S_{j\to i}$; We have $R_{P_{j\to i}} R_{S_{j\to i}} = (R_{P_{j\to i}} R_{S_{i\to j}}^{-1})(R_{S_{i\to j}} R_{S_{j\to i}}) = Id$, as the product inside the first set of parentheses is equal to identity by the induction hypothesis.

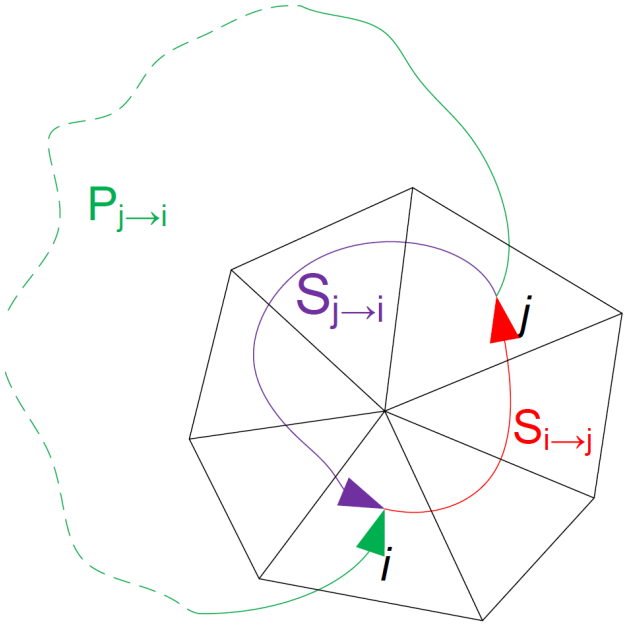


Figure 3.3: Two different paths to reach same triangle.

Thus we can compute all the f_i's following a dual spanning tree from a seed face, such that they are all consistent with the given data. Now we prove that we can also unambiguously reconstruct the vertex locations starting from a seed vertex. First, notice that with f_j consistent with R_{ij}, the edge vector (in Eq. 3.6) is the same when we use the edge equation from either face. Second, by

construction, $a^{\alpha}_{ij} + a^{\alpha}_{jk} + a^{\alpha}_{ki} = 0$, which means moving along each the boundary loop of each face creates no translation. We can immediately see that the coordinates $\mathbf{x}$ can be integrated consistently along any paths starting from any seed vertex, since translations are commutative.

□

Independence to the choice of local 2D frame: Note that we do not need an original immersion in order to locally check the integrability. Also, the conditions are not dependent on the choice of the 2D frames, since the rotation will affect both $R_z(-\theta_{T\to e})$ and $R_z(\theta_{T\to e})$, resulting in just a change of frame variables denoting the same tangent planes.

Links to the continuous theorem: This single condition actually contains 3 independent integrability conditions as 3D rotation has 3 DoFs. We can interpret the condition as the requirement for a normal to be propagated back to itself (2 DoFs), and an additional rotation in the tangent place (1 DoF). The former corresponds to the two Codazzi equations on the symmetry of the covariant derivative of $I\!I$; the latter corresponds to Gauss's equation stating that the Gauss curvature computed by $\theta_{T\to e}$ should be consistent with the area on the unit sphere formed by the N_i's along the dual path.

3.1.3.3 Global discrete fundamental theorem of surfaces

Given a closed mesh with genus g, if, for any one homology basis, (i) the above local integrability condition is satisfied on all but two arbitrarily chosen adjacent vertices, (ii) the accumulated rotation along each dual homology generator (nontrivial loop) is the identity, and (iii) the accumulated translation (following the rotated frames) along each primal homology generator is 0, we can create an immersion of the surface mesh in 3D, unique up to a rigid transformation.

Proof. For any genus-0 surface, removing one triangle is enough to make the local version of our

theorem applicable for the reconstruction of the coordinates of all the vertices. Adding back the triangle introduces 3 dihedral angles that are completely fixed by the local integrability condition around any one of the 3 vertices. For higher genus models, the only difference with the local version is that some loops include combinations of the $2g$ nontrivial loops in addition to the one-ring loops, so we need to make the rotations around them match as well. In addition, unlike in the simply connected case, the translations around these nontrivial loops do not automatically match, hence the additional constraints. □

Matching DoFs: As triangle inequalities do not eliminate DoFs, we can count the number of true DoFs as $2|E|-(3(|V|-2)+6\times 2g)$ (corresponding respectively to the length and dihedral values, the local integrability conditions on all but two vertices, and the additional topological constraints). Using Euler's formula $|V| - |E| + |F| = 2(1 - g)$ and the fact that triangulated manifolds satisfy $3|F| = 2|E|$, we see that there are $3|V| - 6$ DoFs, matching exactly the number of DoFs of an immersed mesh up to rigid transformations. Note that the choice of the homology basis does not matter, since, roughly speaking, any basis will span the space of non-trivial loops.

3.2 Surface Reconstruction from Discrete Fundamental Forms

3.2.1 Immersion Energy through the Lift to 6D

Ideally, the integrability condition should be used as a constraint in the geometry processing procedure producing the discrete fundamental forms. However, in practice, we may have additional point and direction constraints imposed by the user, or conflicting fundamental form data due to various reasons; one such source of incompatibility could be the discretization itself. In this section, we propose a simple energy functional, the minimization of which leads to a globally

optimal reconstruction in the sense that the local differential properties $F^{-1}dF$ deviates from the prescribed invariants in Ω as little as possible. This is equivalent to minimizing the L_2-norm of $dF - F\Omega$ provided that f is orthonormal. In the continuous setting, one could use

$$\int_M \|dF - F\Omega\|^2 = \int_M \|df - f\Omega_R\|^2 + \int_M \|d\mathbf{x} - f\Omega_x\|^2,$$

for such an energy, where $\|\cdot\|$ denotes the Frobenius norm of matrices. In the discrete setting, as we store different components of the lift at different locations, we can simply treat the rotation part and the translation part separately as above. To derive the discrete energy, we use the notation of discrete differential forms in Desbrun et al. [28]. We treat each entry of df and $f\Omega_R$ as dual 1-forms, and each entry in $d\mathbf{x}$ and $f\Omega_x$ as primal 1-forms. The discrete 1-forms are represented by their line integral along edges. Along a dual edge connecting face i and face j, the line integral of the exact form df is simply $f_j - f_i$, and the path-dependent line integral of the 1-form $f\Omega_R$ is approximated by $f_i(R_{ij} - Id)$. The primal 1-form $d\mathbf{x} - f\Omega_x$ can be integrated along the primal edge in each triangle. For any primal 1-form β, the L^2-norm is discretized to $\beta^T *_1 \beta$, where $*_1$ is the primal Hodge star, and for dual one-form γ, it is $\gamma^T (*_1)^{-1}\gamma$. Thus, we arrive at the following weighted least-squares energies

$$E_f(f) = \frac{1}{2} \sum_{e\in E, e=T_i\cap T_j} w_e^{-1} \|f_j - f_i R_{ij}\|^2,$$

and

$$E_{\mathbf{x}}(x, f) = \frac{1}{2} \sum_{e=v_m v_n \in E} \sum_{T\supset e} w_{e,T} \|\mathbf{x}_n - \mathbf{x}_m - f_T \begin{pmatrix} a^1_{mn,T} \\ a^2_{mn,T} \\ 0 \end{pmatrix}\|^2,$$

where the w_e's are the cotangent edge weights of the usual discrete Laplacian [91], and $w_{e,T}$ is the single cotangent term within the triangle T. We can thus construct the mesh by minimizing the following energy (quadratic with respect to (x, f))

$$E_M(x, f) = w\, E_f(f) + E_{\mathbf{X}}(x, f), \tag{3.9}$$

where w is relative weighting (set to $1,000$ in our tests). Varying w leads to a change in the "stiffness" of the mesh, with smaller w preserving the metric I better and larger w preserving dihedral angles better. However, the change in the result is subtle unless the weighting is changed in orders of magnitude.

The resulting linear system will be positive-semi-definite, and after fixing one vertex and one frame, the system becomes positive-definite. The cotangent weights used in E_f are thresholded to be above a small positive number (set to be 10^{-11} in our test) to improve the condition number. Without thresholding, the Cholesky factorization in the linear solver may fail. However, the threshold value does not lead to visually noticeable change on the results even if we increase it to 10^{-3}

We can also formally prove that the reconstruction does not depend on the choice of 2D local frame. Suppose that we perform a random 2D rotation with angle θ_i within each face i. Given the same fundamental forms, the new transition rotation will become $\hat{R}_{ij} = R_z(-\theta_i)R_{ij}R_z(\theta_j)$, and the new $\hat{a}$ satisfies

$$\begin{pmatrix} \hat{a}^1_{ij,T} \\ \hat{a}^2_{ij,T} \\ 0 \end{pmatrix} = R_z(-\theta_i) \begin{pmatrix} a^1_{ij,T} \\ a^2_{ij,T} \\ 0 \end{pmatrix}.$$

Hence, $\|f_j - f_i\hat{R}_{ij}\|^2 = \|f_j R_z(-\theta_j) - f_i R_z(-\theta_i)R_{ij}\|^2$, since the Frobenius norm satisfies $\|AR\|^2 =$

$tr(AR(AR)^T) = tr(AA^T) = \|A\|^2$ for any orthogonal matrix R. Thus if the rotation part of the minimizer of the new energy is $\tilde{f}_i$, $\tilde{f}_i R_z(-\theta_i)$ is the minimizer of the original energy, and the translation part $\mathbf{x}$ remains the same. Note that the above argument holds whether f is orthonormal or not.

3.2.2 Algorithm

The above convex energy function, together with possible constraints, can be optimized through a linear system resembling a Poisson equation. If we arrange elements of f in row-major order, we can express the sparse symmetric linear system using a block matrix; for example, the diagonal entry for the n-th row for $\mathbf{x}$ would be $w_{nn}Id$, and the block on the i-th row and j-th column for f would be

$$-w_{ij}\begin{pmatrix} R_{ij}^T & 0 & 0 \\ 0 & R_{ij}^T & 0 \\ 0 & 0 & R_{ij}^T \end{pmatrix}.$$

We omit further discussions here, as all the coefficients can be deduced from the energy functional. Through its similarity to the Poisson equation, it is easy to see that this block matrix is positive-definite once we fix at least one position and one rotation (or other equivalent combinations of constraints) as mentioned above. The main procedure for surface reconstruction is described in Algorithm 1.

Algorithm 1: Surface reconstruction from discrete first and second fundamental forms.

1. Compute R_{ij} and a from I and $I\!I$ or directly from immersion when it is available.
2. Include hard or soft constraints for positions and (optionally) for rotations through variable elimination or penalty terms.
3. Solve the sparse linear system to get the immersion $\mathbf{x}$.

We do not enforce (and in fact cannot enforce with a linear system) that the frames stay orthonormal. However, as noted in [70], since the R_{ij}'s are orthogonal, f remains close to being orthonormal, except for extreme constraints (in which case, one may (ortho)normalize the resulting f, and solve the system again with the normalized frames as soft constraints. In this optional step, the penalty weight on a frame for deviating from the orthonormal frame in the first solve is set to $1,000$ in our tests. A larger penalty can lead to better preservation of the first fundamental form, but may also produce small discontinuity near constrained vertices). Due to the fact that the frames are stored on triangles, we can also easily derive constraints for f if needed, obtaining the face normal $\mathbf{N}$ by the normalized cross product of two edges for triangles with all three vertices constrained.

3.3 Applications and Results

The basic local rigid-motion invariant formulation for surface equation can potentially benefit various applications involving geometry processing with triangle meshes. We developed a few proof-of-concept examples. We ran all the tests on an Intel Core2Dual@3.0GHz CPU with 4GB RAM.

Figure 3.4: Deformation examples.

3.3.1 Deformation

Our first example is mesh deformation or editing using position and (optional) rotation constraints. Using Region of Interest and prefactorization of the SPD matrix, we can achieve interactive rates for moderate size regions, similar to e.g.[60, 101, 70]. See Figure 3.4. From left to right: original and rearranged models of Octopus, bent Cactus, Armadillo, and Bunny, deformed using our linear reconstruction with position and optional rotation constraints. Octopus was deformed by relocating multiple handles (red) simultaneously, and the other models were results of one or more steps of deformation, with each step using one fixed handle (red) and one movable handle (yellow). Note that our method is robust even for the highly irregular triangle shapes in the Cactus model.

We also show some extreme test cases as in (Figure 10 of) the survey paper [14]. From left to right in Figure 3.5, we did tests on 1) bumpy plane before and after frame orthonormalization; 2) 120° bent cylinder before (top) and after (bottom) orthonormalization; 3) 135° twisted bar before and after orthonormalization; 4) 70° bent cactus. Note that the results with orthonormalized frames are produced by a second linear solve using normalized columns of the frames in the first solve as soft constraints to handle these extreme cases. In these extreme deformations, the second solve produces visible differences except for the cactus case. In the bumpy plane test before normalization, the smooth transition between frames would force the bumps to point upward as the constraints force the frames at both handle regions to point upward.

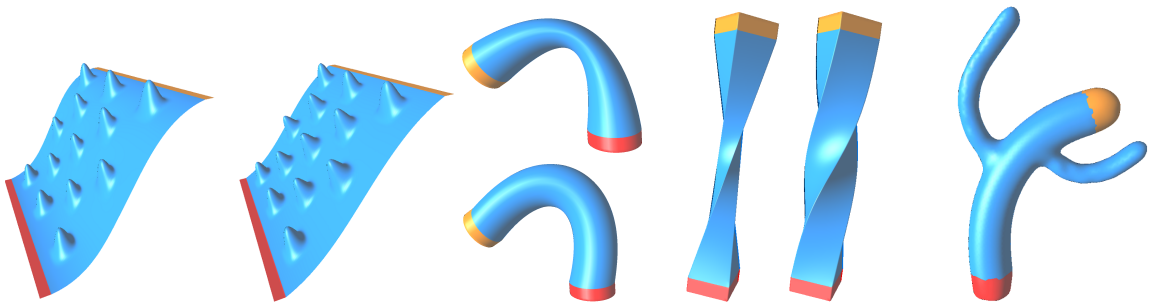
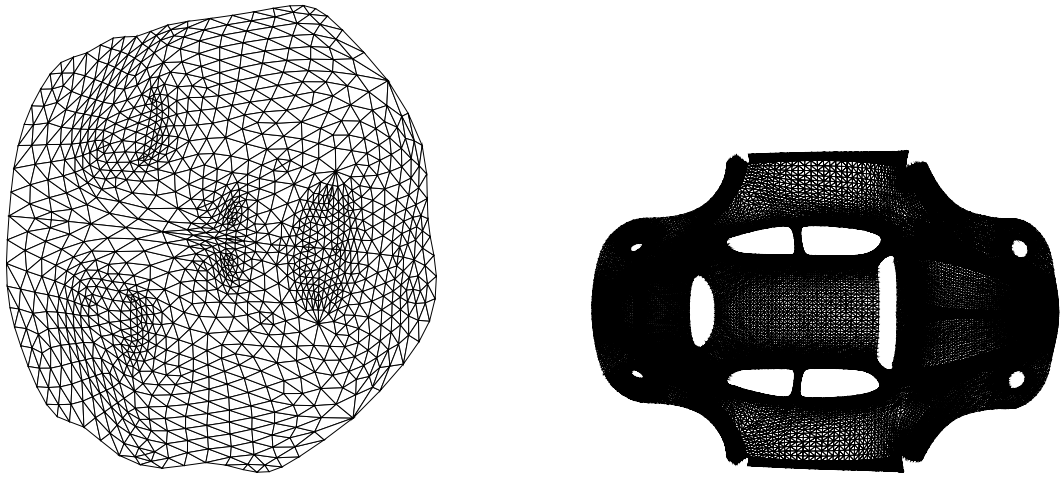


Figure 3.5: Applying our method to the extreme deformation test cases from the survey paper [14].

3.3.2 Quasi-isometric Parameterization

One interesting example when we manipulate the fundamental forms is to set $I\!I$ to 0, thus flattening the surface patch. In some sense, it is striving to preserve the first fundamental form, thus it can be called *quasi-isometric*. This method represents a new linear method to produce a parameterization with a "natural" boundary (see[27]). In this case, orthogonality can be weakly enforced by a quadratic penalty term. We sometimes perform a second linear solve as for the extreme constraints, since the abrupt change in $I\!I$ leads to large deviations from the compatibility conditions. However, as such, it cannot guarantee an embedding in 2D, but just an immersion. In such cases, the target function still forces the frames to produce a planar mesh, but the frames may be turned left-handed, i.e., flipped from the original orientation for certain triangles. However, parameterization of typical meshes rarely gives rise to flips, as illustrated in Figure 3.6.



(a) Nefertiti model. (b) Beetle model.

Figure 3.6: Parameterization produced by setting the dihedral angles to zeros.

3.3.3 Independence to Sampling Density

In Figure 3.7, a torus with varying vertex densities was deformed. As shown from the results, the overall shape remains the same despite the abrupt change in sampling density, and the symmetry of the shape is preserved.

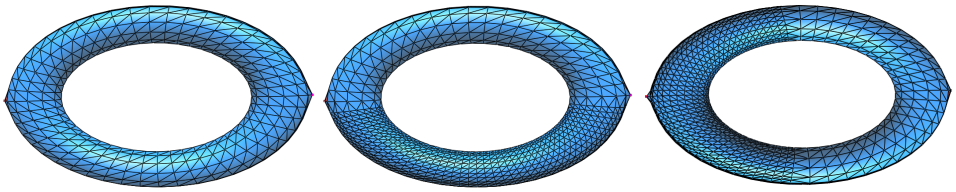


Figure 3.7: Tori with different sampling densities produce similar deformation and preserves symmetry.

3.3.4 Comparison to Existing Work

The goal of mesh deformation and other related geometry processing methods is often to produce reasonable, natural-looking surfaces that satisfy user-specified constraints at near interactive rates.

There are mainly two types of algorithms: surface-based, and ambient-space-based[ii] deformation methods. As our method fits squarely in the first category, we mainly go over related surface-based methods.[iii] Some among these methods focus on point sampled in a global coordinate system and employ a scalar (blending) field to get a smooth transition between constrained and unconstrained parts. For instance, Blandford et al. [7] use geodesic distances to define the blending function and add an implicit occluder function to the mesh editing environment, while Pauly et al. [89] use Euclidean distances in a hybrid geometry model by combining implicit and parametric surface representations. The latter produces more intuitive deformation, while increasing the cost computational time.

With such limited information, the point-based deformation methods are often not physical aware. For better physically plausible results, a number of authors propose to minimize various physically-motivated deformation energies. For example, Chao et al. [20] define an elastic energy based on the distance between the deformation and the rotation group.[iv] Wardetzky et al. [111] provide instead a quadratic curvature-based energy to achieve simpler computations.

Physically-motivated methods have a common drawback, the loss of geometric details for fine-scale surfaces. To preserve geometric details, other authors, Kobbelt et al. [60, 61], Botsch and Kobbelt [13], suggest to combine physically-motivated energy methods with multi-resolution modeling. Alternatively, with consideration of geometric detail as a local intrinsic property of the surface, several papers [101, 69, 114, 121, 115] propose to use so-called *differential coordinates* to encode these details instead of storing them in a global coordinate system. For instance,Yu et al. [114] and Zayer et al. [115] define differential coordinates based on the gradients of original sur-

[ii] also called free-form

[iii] For readers interested in second category, refer to the survey [82].

[iv] An additional thin-shell bending term can be added if necessary.

face coordinates, while Zhou et al. [121], Lipman et al. [69], Sorkine et al. [101] manipulate the Laplacian of the spatial coordinates. See Botsch and Sorkine [14] for a survey on these methods.

Unfortunately, differential-coordinates based methods are often non-linear and require relatively large computations. Among those, the method, proposed by Lipman et al. [70], is the only existing linear and rotation-invariant approach (not including iterative methods with linear steps, of course). It defines a local frame per vertex and encodes the local geometric information into the transition between adjacent local frames. The method produces deformations that preserve local details in the least-squares sense through a linear solve for the local frames and another for the coordinates. It handles rotational constraints efficiently, but is insensitive to translational constraints, since the solve for the frames is decoupled from the positional constraints [14].

There are a number of differences between our method and the only existing linear and rotation-invariant approach Lipman et al. [70]. First, our discrete fundamental forms representation is (around 50%) more compact, independent of the choice of vertex normal, and independent of choice of the edges in the definition of frames (as we have proven mathematically, and observed numerically). Second, we do not require *a prior* an embedding of the triangle mesh. Third, we couple the rotation and translation constraints in a natural fashion. As shown in Figure 3.8, our method produces similar results measured by edge length change, but produces less angle distortion, especially for the more complicated armadillo model. We added compatible rotation constraints for both methods, so that the linear rotation-invariant (LRI) coordinates [70] can produce results, although our method can in fact work with only translational constraints. For these large deformations, we found that orthonormalization of the frames improves the results for both methods. We used a weight of 100 for the afore-mentioned soft constraints approach for orthonormalization of the frames in our method.

Other surface deformation methods with point constraints are either not rotation-invariant or

non-linear. For a comparison to these methods, refer to (Figure 10) of Botsch and Sorkine [14] for models deformed under same type of constraints (bumpy plane, cylinder bending, bar twisting, and cactus bending) as in Figure 3.5.

In Figure 3.8, we visualize the distortion with pseudo-color and histograms. From left to right: angle distortion for results using LRI [70], angle distortion for results using our method, edge distortion for results using LRI, and edge distortion for results using our method. The pseudo color denotes the absolute value of changes in edge length and dihedral angles measured in radian. The sizes of the bounding boxes for the bar model (with 1666 vertices) and the armadillo model (with 16K vertices) are $1 \times 1 \times 6$ and $1 \times 1.2 \times 0.9$, respectively.

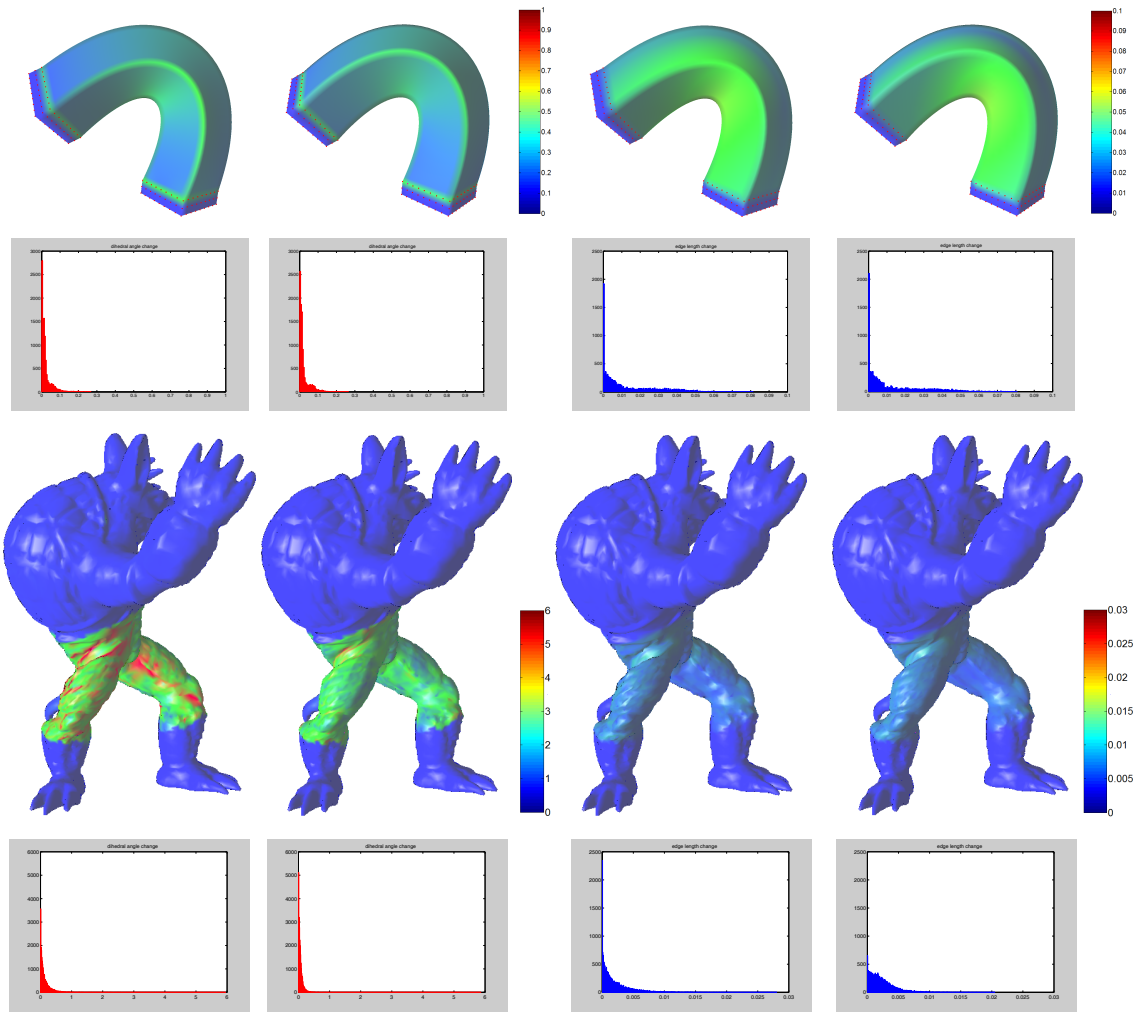


Figure 3.8: Comparison to Lipman et al. [70]'s method.

3.4 Conclusion

In this section, we have introduced compatibility conditions for edge lengths and dihedral angles for meshes with fixed topology, akin to Gauss's equation and the Codazzi equations. We have proposed a sparse linear system based on these lengths and angles, mimicking the continuous surface equation, the solution of which produces a mesh immersed in 3D with edge lengths and dihedral angles close to the specified ones. We have also demonstrated the benefits of such a rotation-invariant representation for mesh deformation applications.

One drawback of our approach (common to other work) is the lack of a guarantee on frame orthonormality—although for certain applications, it can provide additional flexibility necessary to satisfy the constraints. It is an interesting research direction to examine the possibility of using quaternions to represent rotations with a nonlinear but tractable optimization process.

Like the differential surface theory it is based on, our method cannot handle intersections without resorting to other numerical tools. However, this is also common to other methods in the same category, and can actually be remedied by incorporating existing collision handling techniques.

Chapter 4

Vector Field Map Representation for Surface Correspondence

In general, the problem of shape correspondence can be stated as: finding structure-preserving relations among a series of shapes, in particular, point-to-point mappings. The structures to preserve, be they global or local, depend on the application domains, with the simplest scenario being rigid matching for entire shapes. However, general correspondence of deformable shapes remains challenging even for similar complete shapes. We propose a vector field based method for representing the surface correspondence, leading to simplified expressions of constraints used in correspondence inference, in particular, the angle-preserving constraint.

4.1 Related Work

There are a multitude of existing correspondence algorithms in various application domains. Different error measures can be used to define correspondences with different properties. For instance, the traditional alignment applications involve rigid alignment, which keeps the Euclidean distances

between any pair of points. A typical iterative closest point (ICP) algorithm can be used to find the best rotation between two 3D shapes for partial matching [6]. Variants of ICP have also been proposed for isometric correspondences (e.g. Huang et al. [46], Bronstein et al. [18]). Under large deformations, sparse sets of correspondences can be found by methods such as deformation-driven shape correspondence [118] and voting after Möbius transformations [68]. A recent survey on the correspondence problem can be found in [109].

For many current mesh correspondence methods, the representation of the relation between shapes are point-based, which renders the search for such relations a costly procedure. However, for certain applications, it is unnecessary to build a point-to-point mapping in-between. No reasonable mapping may even exist for meshes with vastly different numbers of vertices. Methods such as [50, 68] suggest to build a mapping between a sparse subset of the vertices, which is called landmark correspondence. Theoretically, starting from the sparse set, it is possible to produce a unique dense correspondence between isometric shapes [85, 68], or through cross-parameterization even in a non-isometric case [1, 63]. Spherical parameterization can also be used to bridge the mapping between two meshes, and optimal conformal maps can also be computed [53, 116]. However, all of these approaches employing point-based presentations rely naturally on local correspondences, without direct access to global structures.

Recently, Ovsjanikov et al. [86] proposed a novel representation based on *functional maps* between two shapes. Instead of finding correspondence directly between surface points, they suggested to build a map between the *function spaces* associated with the surface pair. Their representation has two major benefits: first, flexibility in the choice of bases for the function space on each shape; second, linear function preservation constraints for inference. Thus, the representation is well-suited for correspondence inference, and shape analysis of collections of isometric shapes. Coupled harmonic basis proposed by Kovnatsky et al. [62] can further increase the sparsity of the

functional map, but it requires an initial correspondence before the joint basis can be computed. Applications of the representation have also been proposed in visualizing maps and measuring shape differences using maps [87, 94].

The functional map representation is particularly convenient when employing a specific basis for function spaces, closely tied to spectral analysis. As a popular technique used widely in computer vision, signal processing and other computational sciences, spectral analysis can also be adapted for irregularly sampled shapes in geometry processing. Ever since Taubin [105] first introduced the approach into the computer graphics community, it has been a powerful computational device in various geometry processing contexts, e.g. in [65, 46]. A commonly used basis for traditional spectral analysis of functions is composed of the eigenfunctions of the Laplace-Beltrami (LB) operator. In fact, the 1D Laplace-Beltrami eigenbasis (LB-basis) for the unit circle simply consists of the familiar sine and cosine functions with integer frequencies. The LB operator on triangle meshes is often discretized by the cotan formula used in denoising [26] and mesh editing [101], for instance. The 2D mesh version of LB-basis has already been used in correspondence problems [93, 52, 74], although they did not allow mixing across different frequencies as in [86]. Our vector field map representation also employs a vector field version Laplace-Beltrami operator basis (LB-basis). We also present a complete set of operators in triangle mesh LB-basis, including gradient, curl and divergence, which are not only necessary for our representation but also convenient in other areas of spectral analysis of vector fields.

Similar to functional map, our vector field map representation uses much less storage by restricting to only the low frequencies, and admits much more flexible correspondence specifications, such as geometric detail function matching and segment matching, than the point-based representations. However, unlike functional map, our method can handle conformal mapping in correspondence inference, extending the applicability of the method. As a gradient domain method, the

tangent vector field map can handle topological change naturally. The final map is constructed through a Poisson reconstruction with the landmarks as the boundary condition.

4.2 Functional Maps Between Shapes

We briefly review the definition of the functional map representation here, loosely following the notations used in [86]. Given a smooth manifold M, a smooth scalar function f on M is defined as a mapping $f : M \to \mathbb{R}$. All such smooth scalar functions form a functional space, denoted as $\mathcal{F}(M, \mathbb{R})$. For two smooth manifolds M and N related through a smooth bijective map $T : M \to N$, we can establish the correspondence between functions defined on them. More specifically, from a scalar function $f \in \mathcal{F}(M, \mathbb{R})$, we construct a scalar function $g = f \circ T^{-1}$ through composition. For simplicity, we use a symbol T_* to denote the map as $T_*(f) = f \circ T^{-1}$. With $\{\phi_{M,i}\}$ and $\{\phi_{N,i}\}$ as bases for functional spaces on M and N respectively, the linear operator T_* on function spaces can be represented by a matrix $C = [c_{ij}]$, such that $T_*(\phi_{M,j}) = \sum_i c_{ij}\phi_{N,i}$. The mapping between functions f and g can thus be expressed as

$$T_*(f) = T_*(\sum_j a_j \phi_{M,j}) = \sum_j a_j T_*(\phi_{M,j}) \tag{4.1}$$

$$= \sum_i \sum_j c_{ij} a_j \phi_{N,i} = \sum_i b_i \phi_{N,i} = g, \tag{4.2}$$

where $\mathbf{a} = [a_i]_i$ and $\mathbf{b} = [b_i]_i$ are the coefficient vectors of f and g in their corresponding functional basis. In matrix multiplication form, the above equation is simply $C\,\mathbf{a} = \mathbf{b}$.

One of the advantages of the functional map representation is that whenever a constraint can be translated into functional correspondence $T_*(f) = g$, be it matching of geometric descriptor functions, landmarks or segments, the constraint can be translated into a simple linear constraint

on C, i.e. $C\mathbf{a}–\mathbf{b}=0$. Operator commutativities can also be expressed as commutatvities of matrices. Furthermore, area-preservation can be depicted by the orthonormality of C. See [86] for proofs of those propositions.

4.3 Spectral Vector Field Analysis on Surface

Before describing the vector field map representation for correspondence, we first present the necessary tool, a novel spectral vector field analysis based on exterior calculus (introduced in Section 2.3.2.5). We associate the concepts in exterior calculus with their counterparts in usual vector field analysis. In fact, for a Riemannian surface (a surface with a metric), we have canonical mathematical one-to-one correspondences between both the fields and the operators in exterior calculus and those in vector field analysis. Denoting the area element by μ, we have listed the correspondences in the following lookup tables:

- fields defined on surfaces:

0-form ω^0	scalar function f
1-form ω^1	tangential vector field $\mathbf{v}$
2-form ω^2	scalar function with area element $f\mu$

Table 4.1: Vector fields defined on surfaces.

- operators:

differential $d^p : \omega^p \mapsto \omega^{p+1}$	
d^0	gradient
d^1	curl of a tangent vector field
(Hodge-)dual operator $\star^p : \omega^p \mapsto \omega^{2-p}$	
$\star^0$	multiplication by area element
$\star^1$	rotation by $\pi/2$ counterclockwise
$\star^2$	division by area element
wedge operator $\wedge : (\omega^p, \omega^q) \mapsto \omega^{p+q}$	
$\omega_a^0 \wedge \omega_b^0$	pointwise product
$\omega^0 \wedge \omega^1, \omega^1 \wedge \omega^0$	pointwise scalar product
$\omega^1 \wedge \omega^1$	pointwise cross product

Table 4.2: Differential operators defined on surfaces.

- L_2-inner product:

$\int_M \omega_a^p \wedge \star^p \omega_b^p$	scalar/vector L_2-inner product

Table 4.3: L_2-inner product defined on surfaces.

More precisely, 1-forms are covector fields, which are dual to vector fields, i.e. fields of linear mappings from tangent vector space to real number. Since the surfaces used here are all equipped with the metric induced by the 3D space, there is indeed a canonical isomorphism between vectors and covectors (through contraction with the metric or its inverse). Thus, we use the two terms interchangeably. Note that the mapping in our representation is the mapping induced for 1-forms,

which can then be treated as the mapping between tangent vector fields through the isomorphisms, as we describe in Section 4.4.1.

We use superscript to denote which type of forms we apply the operator to, and subscript to denote individual fields or surfaces. We drop the superscript when it is clear from the context. Some other operators can be created by the combinations of the basic operators:

$d\star^1$	divergence
$-\star d^0$	curl of a normal vector field
$\Delta^0 = \star d \star d^0$	$\nabla\cdot\nabla$ scalar LB operator
$\Delta^1 = \star d \star d^1 + d \star d \star^1$	$-\nabla\times\nabla\times+\nabla\nabla\cdot$ vector LB operator

Table 4.4: Other differential operators represented as combinations of basic operators.

In particular, Δ^0 is reduced to the commonly used cotan formula following the discretization of the basic operators in [29].

Through the generalized eigenvalue problem with symmetric matrices $d\star df = -\lambda \star f$, we obtain a series of eigenvalues λ_i in ascending order. These λ_i's correspond to squared spatial frequencies of the respective eigenfunctions. The 0-th frequency associated to the constant function is always 0. The set of eigenfunctions $\{\phi_i\}$ form the LB-basis for scalar functions. We drop the constant function ϕ_0 from the basis for clarity, as any continuous map between surfaces would induce a trivial mapping between constant functions. As a unit vector, the function has a constant value that is equal to the inverse of the square root of the total area. With the total surface area rescaled to 1, $\phi_0 = [1\ 1 \ldots 1]$.

While eigenfunctions can be obtained by solving a similar generalized eigenvalue problem for vector field LB-operators, we note that each $d\phi_i/\sqrt{\lambda_i}$ and $\star d\phi_i/\sqrt{\lambda_i}$ are both unit eigenfunctions of Δ^1 with eigenvalue λ_i. Here, d corresponds to gradient as mentioned above, and $\star d$ corresponds

to curl applied to the normal field with magnitude determined by the scalar field. According to the Helmholtz-Hodge decomposition, they span the entire space of 1-forms together with $2g$ independent harmonic 1-forms (eigenfunctions of eigenvalue 0), where g is the genus of the surface. Hence, they form the LB-basis for vector fields. For simplicity, we drop the harmonic 1-forms since they are not involved in the correspondence computation (detailed in Section 4.4.1). Note that differentials of piecewise-linear 0-forms are piecewise-constant (co)vector fields, so $\star^1$ can be applied within each triangle for the 1-form basis fields. If discrete 1-form representation (one integral along each edge) is needed, a simple average of the two integrals for the constant vector in the two triangles incident to the edge can be used, also known as the primal-to-primal dual in DEC. This would only introduce negligible error since the low frequency fields are extremely smooth.

Figure 4.1 shows a few example pairs of eigen-vector-fields used in the derived 1-form LB-basis. Similarly, LB-basis for 2-forms can be induced by the 0-form LB-basis. We simply multiply the area element to get $\star^0\phi_i$. We again drop the 0-th eigenfunction in the following discussion.

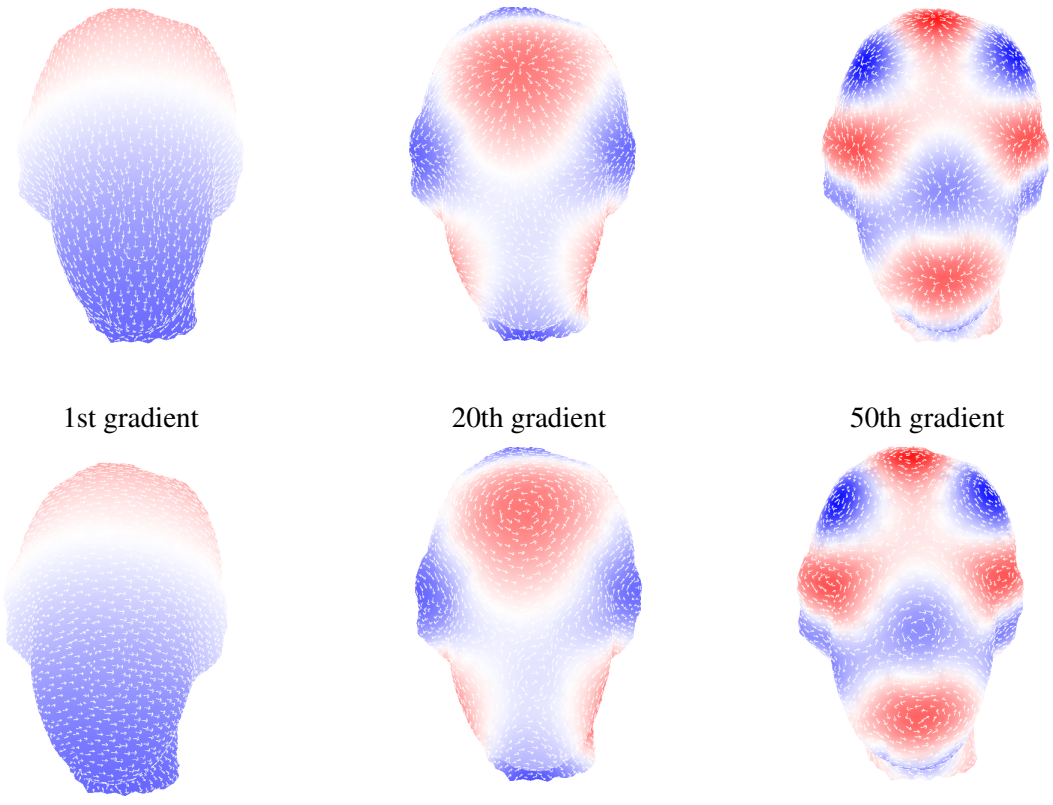


Figure 4.1: The vector field basis functions (shown as arrows) derived from the eigenbasis for functions (intensity shown by color).

Through discretization and keeping only the n leading low frequencies, most operators are made extremely simple in LB-bases, as in regular Fourier analyses. We use capital letters to denote operators in LB-bases, and assume that the LB-bases are arranged as follows:

0-form basis	$\Phi^0 = [\phi_1\ \phi_2\ \dots\ \phi_n]$
1-form basis	$\Phi^1 = [\frac{d\phi_1}{\sqrt{\lambda_1}} \dots \frac{d\phi_n}{\sqrt{\lambda_n}}\ \frac{\star d\phi_1}{\sqrt{\lambda_1}} \dots \frac{\star d\phi_n}{\sqrt{\lambda_n}}]$
2-form basis	$\Phi^2 = \star^0\Phi^0$

Table 4.5: Arrangements of LB-bases.

Any field projected onto the low frequencies can be expressed as $\omega^p = \Phi^p \mathbf{a}^p$, where $\mathbf{a}^p$ is the coefficient vector for the p-form. In order to express the spectral domain operators effectively, we first arrange the frequencies into a diagonal matrix $D = \text{diag}[\sqrt{\lambda_i}]$. Now taking derivatives and taking integrals will simply be multiplying or dividing by the frequencies in LB-bases, respectively. The vector field operators can be denoted by linear matrices applied to coefficient vectors. We list the basic operators below, and some other operators can be expressed as their combinations as in the spatial domain.

gradient	$2n \times n$ matrix GRAD $= \begin{bmatrix} D \\ 0 \end{bmatrix}$
curl on vector field	$n \times 2n$ matrix CURL $= [0\ D]$
$\star^1$ (rotation by $\pi/2$)	$2n \times 2n$ matrix $R = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix}$

Table 4.6: Differential operators represented in the spectral domain.

Note that we ignore the constant function, otherwise, there would have been an additional leftmost column with zeros for GRAD. The operator R resembles a 2D rotation by $\pi/2$, except that I is the $n \times n$ identity matrix. Laplace-Beltrami operators are simply D^2 (second derivative) by construction. Integrating a gradient field, for instance, is achieved by $[D^{-1}\ 0]$ (integration constant ignored). In addition, $\star^0$ and $\star^2$ are both just identity matrices. L_2-inner product is also simply vector inner product for coefficient vectors as if in Euclidean spaces.

4.4 Tangential Vector Field Map

We present the theoretical description and implementation details on the vector field map representation in this section. Constraints for inference of a proper tangential field mapping induced by a point-to-point correspondence are then introduced.

4.4.1 Properties of the basic representation

As mentioned above, not all linear maps between functional spaces are induced by a point-to-point correspondence. Although it can be regularized through the commutativity between the scalar field map and the Laplace-Beltrami operator together with a postprocessing to orthogonalize the map, an implicit restriction is required for the correspondence to be isometric. These regularization terms are essential for correspondence inference applications, so the isometric requirement is limiting the applicability of the map representation. In order to handle shape correspondences with large variation and enable vector field transfer while retaining the compactness of the functional map representation, we introduce the tangential vector field map.

4.4.1.1 Induced maps (pullbacks)

As in the case for mapping between scalar functions, it is possible to induce a mapping between vector fields from the bijective map $T : M \rightarrow N$, defined by the Jacobian matrix (first derivative DT) of the map T, which maps an infinitesimal motion on M to the corresponding infinitesimal motion on N. Mapping a vector field on M to a vector field on N is also known as the *pushforward*. The transpose of the Jacobian can be used as a mapping from 1-forms (covector fields) on N to those on M, known as the *pullback*. We choose the pullback induced by T^{-1} as our basic representation, which maps 1-forms on M to 1-forms on N. When using our representation to map

a true vector field, the procedure is equivalent to turning the vector field to a covector field through the metric on M, mapping it to N through the pullback ${T^{-1}}^*$ induced by T^{-1}, and turning the covector field back into a vector field through the metric on N.

In the following, we use C^p to denote the matrix for mapping p-forms from M to N using LB-bases, and use subscript to denote different surfaces. As 0-forms are scalar functions, C^0 is just the $n \times n$-matrix in the original functional map representation. Note again that we discard the trivial mapping between the constant function ϕ_0, since constant functions are mapped to constant functions equal to the same constant, under the assumption that we only deal with single-piece surfaces.

One important property of this induced map is its commutativity with the differential operator d (exterior derivative, regarded as the dual of gradient), $dC^0 = C^1 d$. A mapping between 2-forms can also be induced from T, which commutes with d (regarded as the curl) as well, $dC^1 = C^2 d$.

4.4.1.2 Conformality and orthogonality

One crucial property of the vector field map representation is that conformality can be expressed as the orthonormality of C^1. This can be seen from the preservation of the L_2-norm of a 1-form

$$\int_M \omega^1 \wedge \star_M \omega^1 = \int_N ({T^{-1}}^* \omega^1) \wedge \star_N ({T^{-1}}^* \omega^1).$$

Through the correspondence T, M and N can be regarded as the same surface with different metrics, but an angle-preserving conformal T will not changing the 90° rotation $\star^1$, and $\wedge$ is metric independent, so the two integrals are identical. The discrete version of the above formula is

$$\mathbf{a}^T \mathbf{a} = \mathbf{a}^T {C^1}^T C^1 \mathbf{a},$$

for any discrete 1-form. Thus C^1 must be orthonormal.

For instance, for gradient fields transferred by C^1, the orthonormality can be understood as a result of conformal invariance of the Dirichlet energy of any function $E_D(f) = \int_M |\nabla f|^2$: where the area is increased, the corresponding gradient is reduced, since the distance required to produce the same change in f is increased.

4.4.1.3 Block-diagonal form of C^1

The covector field map C^1 is a $2n \times 2n$-matrix, which can be written in the following 2×2 block matrix form,

$$C^1 = \begin{pmatrix} \bar{C} & C' \\ C'' & \tilde{C} \end{pmatrix}. \tag{4.3}$$

As mentioned earlier, we omit the $2g$ harmonic components for correspondence inference, because T is completely determined by the scalar function map C^0 through the mapping of the indicator function of each point, and C^0 is determined only by the mapping of the gradient fields. The commutativity of C^p and d, i.e. the fact that gradients are mapped to gradients, simplifies the expression of C^1: given arbitrary gradient field $df = (\mathbf{a}, 0)^T$,

$$C^1 df = (\bar{C}\mathbf{a}, C''\mathbf{a})^T = dC^0 f = (\mathbf{b}, 0)^T,$$

which implies that $C'' = 0$ since $\mathbf{a}$ is arbitrary. Similarly $(C^1)^{-1}$ is also with a trivial bottom left block, but $(C^1)^{-1} = {C^1}^T$ as we restrict our discussion to the near conformal case, so $C' = 0$ as well. Thus, $\bar{C}$ is the mapping between gradient fields, and $\tilde{C}$ is the mapping between curl fields.

Under conformal mapping, rotation by 90° is preserved, $RC^1 = C^1R$, leading to

$$\begin{pmatrix} 0 & -\tilde{C} \\ \bar{C} & 0 \end{pmatrix} = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}\begin{pmatrix} \bar{C} & 0 \\ 0 & \tilde{C} \end{pmatrix}$$
$$= \begin{pmatrix} \bar{C} & 0 \\ 0 & \tilde{C} \end{pmatrix}\begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\bar{C} \\ \tilde{C} & 0 \end{pmatrix},$$

which is equivalent to $\bar{C} = \tilde{C}$. Thus, we only need to keep $\bar{C}$ as the main variable, implying the same efficiency for our representation C^1 as for functional map C^0.

4.4.1.4 Relation with C^0 and C^2

The commutativity with d, $C^1 \,\mathrm{GRAD}_M = \mathrm{GRAD}_N C^0$ is equivalent to $C^0 = D_N^{-1}\bar{C}D_M$, i.e. C^0 is the same as taking the gradient, mapping the gradient field through C^1, and integrating the gradient. Likewise, $C^2 = D_N\tilde{C}D_M^{-1}$, i.e., C^2 is integration, C^1-mapping of a curl field, and differentiation. In other words, the vector field mapping includes the information for both scalar field mapping and the 2-form mapping.

4.4.1.5 Isometry

In the special case of isometry, there are additional constraints on C^0 that can be used in correspondence inference. For instance, the orthogonality of C^0 due to

$$C^{0T}C^0 = D_M\bar{C}^T D_N^{-2}\bar{C}D_M = D_M\bar{C}^T\bar{C}D_N^{-2}D_M = I,$$

since $\bar{C}$ commutes with the Laplacian D_N^2 and the two spectra are identical $D_M^2 = D_N^2$. It can also be seen as a result of the additional area preservation constraint, as the L_2-norm of any function is

preserved under that constraint, $(C^0\mathbf{a})^T C^0\mathbf{a} = \mathbf{a}^T\mathbf{a}$. This serves as the basis for the C^0-based ICP algorithm in improving the results (as explained in Sec. 4.4.3).

4.4.2 Constraints

As a generalization of C^0-based method, we can use any constraints described in Ovsjanikov et al. [86]. In addition, we list some additional constraints for correspondence inference.

4.4.2.1 Vector field constraints

Given a pair of vector fields that are supposed to be mapped to each other, we can create a $C^1\mathbf{a}-\mathbf{b}$ type constraint, where $\mathbf{a}$ and $\mathbf{b}$ are coefficients for the vector fields on M and N respectively. One such example is the gradient fields of scalar functions, since the constant component does not actually produce useful information.

It is easy to see that all C^0-based constraints can still be used through a variable substitution. For example, we can keep any C^0 type function pair constraints, and regard them as the gradient field constraints filtered by a low pass filter D_N^{-1}. Feature direction alignment or user-specified direction alignment can be naturally incorporated (e.g. Figure 4.10).

4.4.2.2 Area preservation

We also note that area preservation can be expressed as a soft constraint $w(C^0 - C^2) = 0$, as they are mappings of scalar fields with or without multiplication by the area element. In all our tests, we set the weighting $w = 100\sum|\mathbf{a}|^2$ to balance the regularization with all $C\mathbf{a}$–$\mathbf{b}$-type constraints. Another way to see that area preservation is equivalent to the above equation is through the fact that the Hodge dual $\star^0$ (multiplication by the area element) is commutative with the mapping $C^2\star^0_M = \star^0_N C^0$, and $\star^0$ is simply the identity map in LB-bases. The area constraint combined

with the hard constraint on conformality can deduce the commutativity of C^0 with the Laplace-Beltrami operator, but the relative weighting would be different in the least-squares sense.

4.4.2.3 Landmark constraints

It is acceptable to use Gaussian kernel function pairs to enforce landmark correspondences as they are invariant under isometry. However, Gaussian kernels are not invariant under conformal mapping. Thus we design a simple construction for function pairs that would indeed correspond to each other under conformal mapping. We pick a subset of the landmarks, set one of them to 1 and the others to 0, and solve for a harmonic function with these as boundary constraints. Since harmonic functions are invariant under conformal mapping, they form natural landmark functions compatible with the conformal class of mappings.

4.4.2.4 Segment constraints

As pointed out by Ovsjanikov et al. [86], segmentation correspondences are often more robust than landmark correspondences in automatic detection. For segment constraints, we can directly use the characteristic function pairs. Alternatively, we can follow the same procedure as the landmark constraints, and turn it from the characteristic function pairs to a smoother representation. More precisely, we set the points on one segment to 1, set the points on some other segments to 0, and leave the rest to be determined by solving a Laplace equation. With the harmonic functions defined on some of the segments, we can have a smoother constraint for segment association.

4.4.2.5 Constraint splitting

For near isometric cases, many descriptor functions are available. When there are fewer but higher confidence corresponding function or vector field pairs, constraint splitting can increase the effi-

ciency of using the information for each pair. In theory, with only two pairs of matching generic functions, one can determine the entire mapping, as each level set of a function f should be mapped to the corresponding level set of the matching function g. On the other hand, $C\mathbf{a} - \mathbf{b}$ can only provide n constraints. We propose to use compound function pairs $s \circ f$ and $s \circ g$, where $s : \mathbb{R} \to \mathbb{R}$ is an arbitrary function. We found it stable to use a box function (square pulse) as s to select unions of level set in intervals, or use a Gaussian kernel as a blurred version of the box function.

4.4.2.6 Symmetrization

For numerical stability, we can use the fact that ${C^1}^T = (C^1)^{-1}$ to duplicate the vector field constraints, i.e. to add ${C^1}^T \mathbf{b} - \mathbf{a} = 0$. As the orthonormality constraint is not a linear constraint for C^1, we cannot include it in the least squares system. So the inclusion of the dual constraint improves the initial guess for C^1. The same procedure can also be used for C^0 in the isometry case.

4.4.3 ICP in the Gradient Embedding Space

The elegant method of matching the mesh embedded in the space of indicator function coefficient vectors by Ovsjanikov et al. [86] turns the deformable matching problem into a rigid registration in n-D space. First, treating the coefficients of an indicator function $\mathbf{a}_{v_i}$ of each vertex v_M on the source mesh as the coordinates in the n-D space, and similarly $\mathbf{b}_{v_i}$ for target vertex v_N, both meshes are embedded in the n-D space. Then, a C^0-based ICP method can be directly applied: find the best point-to-point correspondence under the current rotation C^0; find the best C^0 such that ${C^0}^T C^0 = I$ and C^0 aligns the n-D points to their counterparts best; repeat the two steps.

The matrix C^0 is only orthonormal when the map is area preserving, which is true in the isometric case. In dealing with the class of conformal mapping instead of isometry, we replace this

requirement by the orthonormality of C^1. We list the detailed steps in C^1-based ICP:

1. Construct function preservation constraints as $D_N^{-1}\bar{C}D_M\mathbf{a} - \mathbf{b} = 0$.

2. Construct vector field preservation constraints and include area preservation constraint as a regularization term.

3. Solve the linear system to obtain the initial C^1.

4. Build the point-to-point correspondence $v_M \to v_N$ by finding target vertex v_N with the indicator function $\mathbf{b}_{v_N}$, which matches best with the mapped image of the indicator function $\mathbf{a}_{v_M}$ associated with source vertex v_M.

5. Find the best rotation C^1 minimizing the L_2-distances between n-D vectors $D_M\mathbf{a}_{v_M}$ and the corresponding $D_N\mathbf{b}_{v_N}$. (e.g. Kabsch [54])

6. Iterate the above two steps, and produce the final point correspondence if needed.

An efficient way to perform the search in Step 4 is by building a kd-tree populated by all the $\mathbf{b}_{v_N}$'s first. The indicator function used in the functional map representation is chosen to be the Dirac function, which integrates to 1 over the surface. For each vertex v, setting the function value to be the inverse of its lumped area (a third of the one-ring area) $1/\star^0(v)$ and 0 everywhere else would produce such a function. Projecting onto the LB-basis, each row of the basis matrix Φ^0 directly gives the correct coefficient vector $\mathbf{a}_v$ in each row, since $(\Phi^0)^T\star^0$ is the projection operator. Performing additional smoothing or regularization can make the indicator functions relate to each other in a more continuous way. For example, an optional Gaussian smoothing can be easily performed by multiplying the i-th coefficient by $e^{-h\lambda_i}$ for a small diffusion time h (e.g., $h = 2/\lambda_{max}$ is used in all our tests, where λ_{max} is the largest eigenvalue for the eigenfunctions in the LB-basis, assuming that all surface areas are rescaled to 1).

4.5 Results

4.5.1 Conformal correspondences

We compare the results using our representation (C^1-based) mainly with the results using the only other compact map representation based on spectral analysis (Ovsjanikov et al. [86], C^0-based). In the example shown in Figure 4.2, we demonstrate that C^1 normalization does lead to correct conformal maps, for correspondence pairs with ground truth conformal mapping computed based on, e.g., the spin transform Crane et al. [22]. The discontinuous artifacts on the result using the functional map representation show the incompatibility of an orthonormal C^0 and conformal mapping. In the C^0 case, the orthonormalization of C^0 still provides regularization to the results even though insufficient, as the point-to-point (P2P) correspondences were much worse before orthonormalization. In this test, we simply used three landmarks, due to the lack of useful corresponding geometric descriptors in this extreme case.

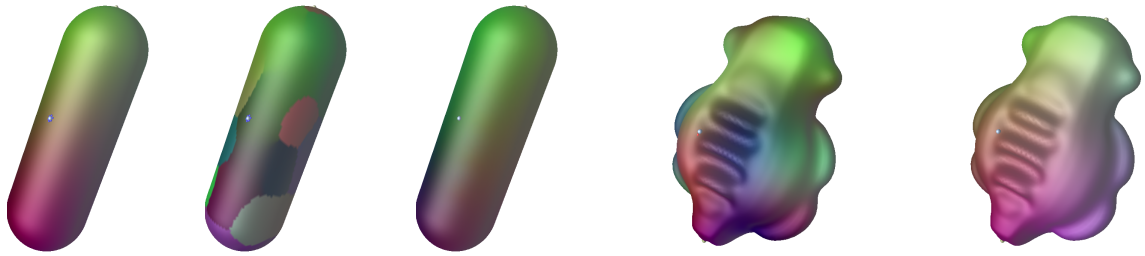
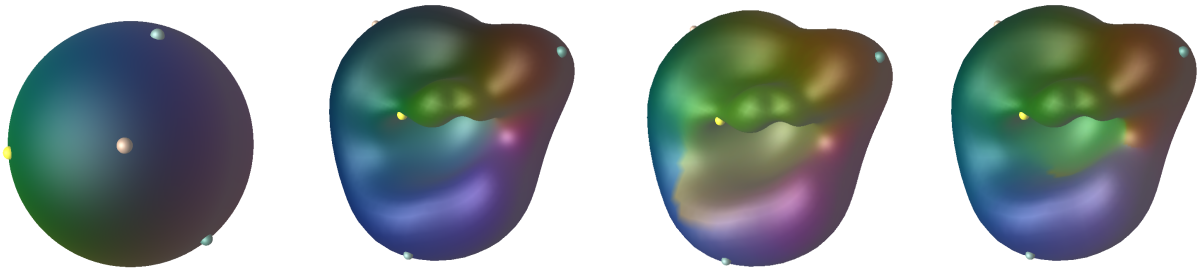


Figure 4.2: Comparison of transferring coordinate functions and point-to-point (P2P) correspondence between C^0 and C^1 representations, on a capsule (top) conformally mapped to another shape (bottom). Three pairs of landmarks are shown as dots. For P2P correspondence, we draw on each source mesh vertex the (x, y, z) of the corresponding vertex on the target as RGB color components (top). For function transfer comparison, we transfer the coordinate functions of the source to the target (bottom).

Figure 4.3 presents another example for a conformal pair of shapes.



Source shape Ground truth P2P Through C^0 Through C^1

Figure 4.3: Comparison of P2P mapping through C^0 and C^1.

We ran our tests on a dataset consisting of all 8 pairs of non-copyrighted models shown in Crane et al. [22], each with 5 manually selected landmarks. We measure the correspondence rate by the percentage of points mapped to a point within a given geodesic distance away from the ground truth point. The error plot in Figure 4.4 shows the advantage of C^1-based method in the conformal case. In both cases, ICP improves the results, but the initial estimate in C^1 map is already better than the final C^0 map. It also shows that when the connectivity of mesh is changed through a remeshing to a uniformly sampled mesh with approximately the vertex count, the spectral methods are resilient to such changes.

We also evaluated our method on the bunny-to-sphere sequence dataset provided by Crane et al. [23], which is a sequence of shapes generated from a conformal flow from the Bunny model to a sphere. The entire sequence contains 218 frames in total. We set up the dataset with 5 landmarks specified by users. We performed a test of mapping from frame-0 to all other frames. Figure 4.5 shows the result of this experiment. In the above error plot of Figure 4.5, the shapes are more isometric to each other, so the C^1 representation produces slightly better results than C^0, but when we only choose the pairs at least 150 frames apart, which are less isometric to each other, the margin is larger.

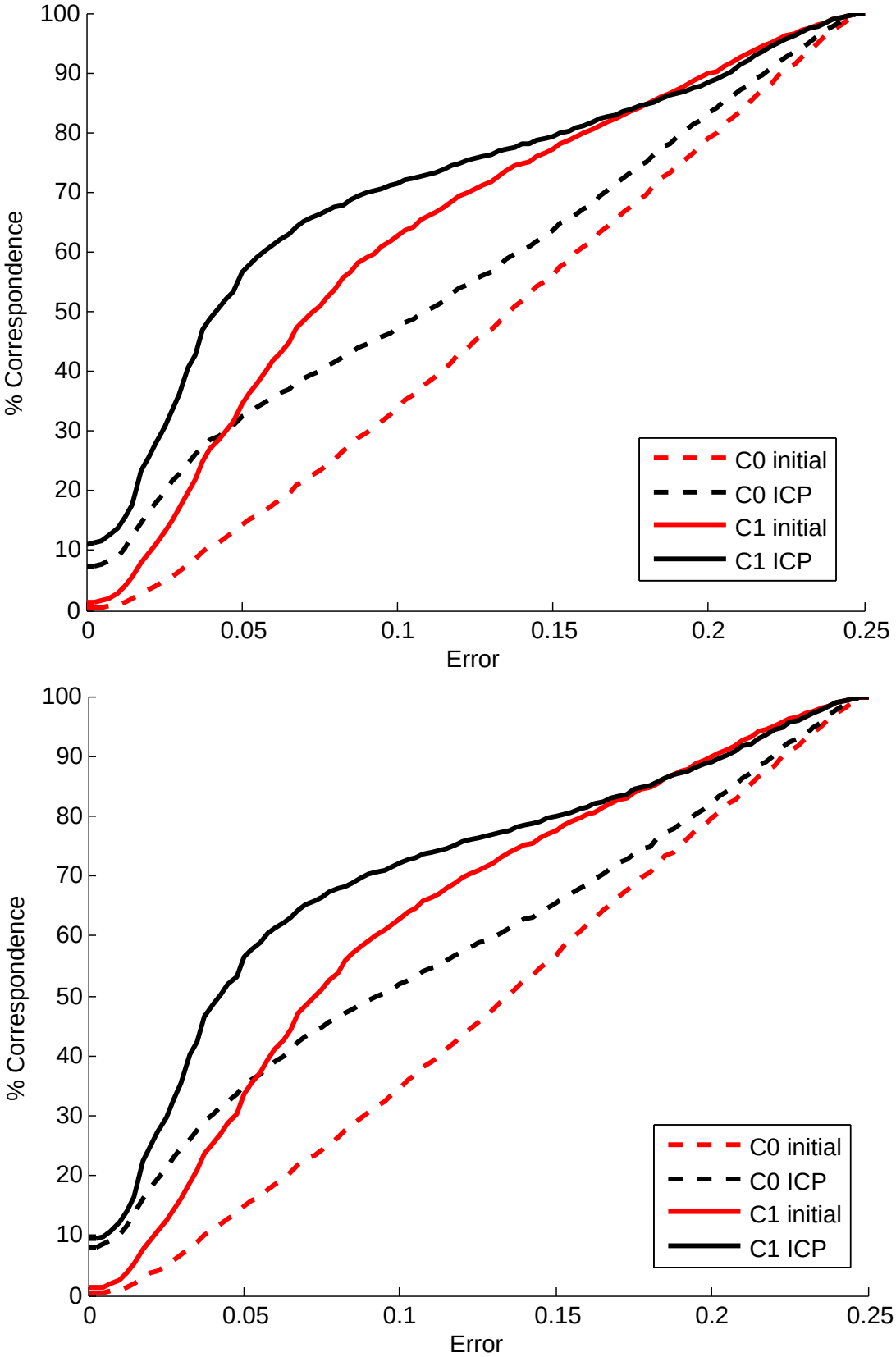
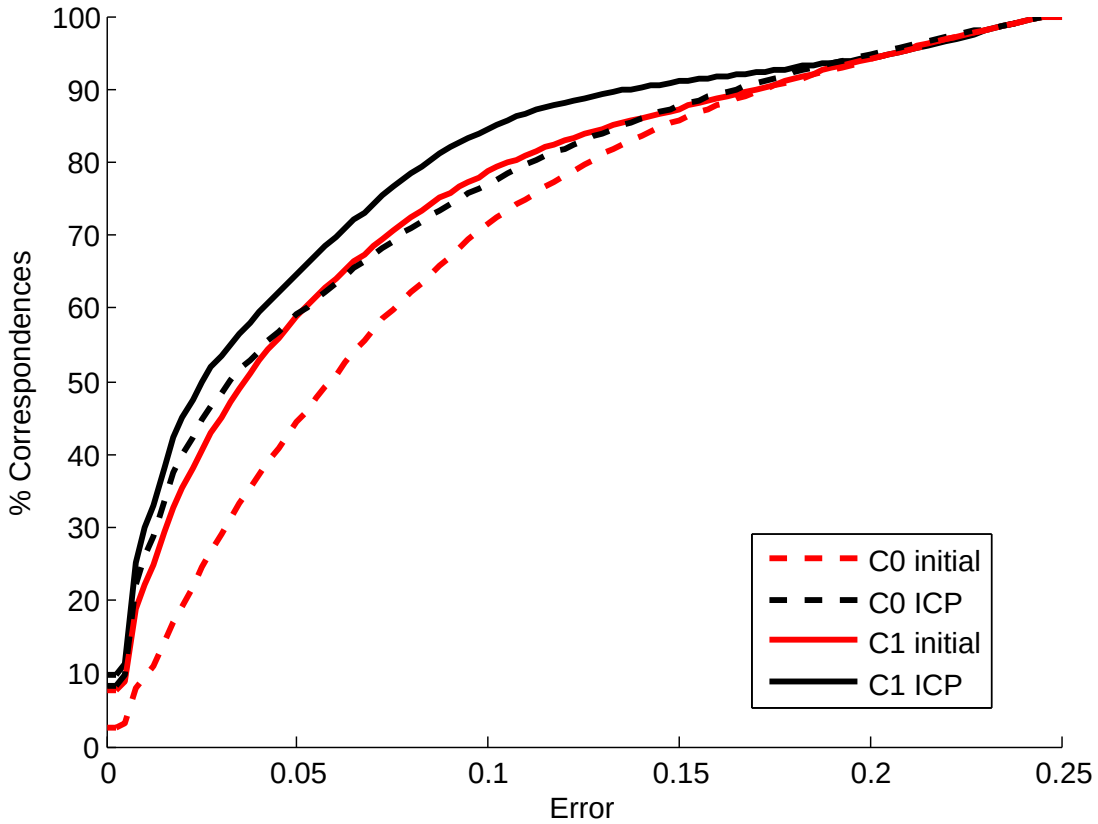


Figure 4.4: Error plot for conformal pairs (top) and conformal pairs with connectivity change (bottom). This test clearly indicates that C^1 is better suited for conformal deformation.



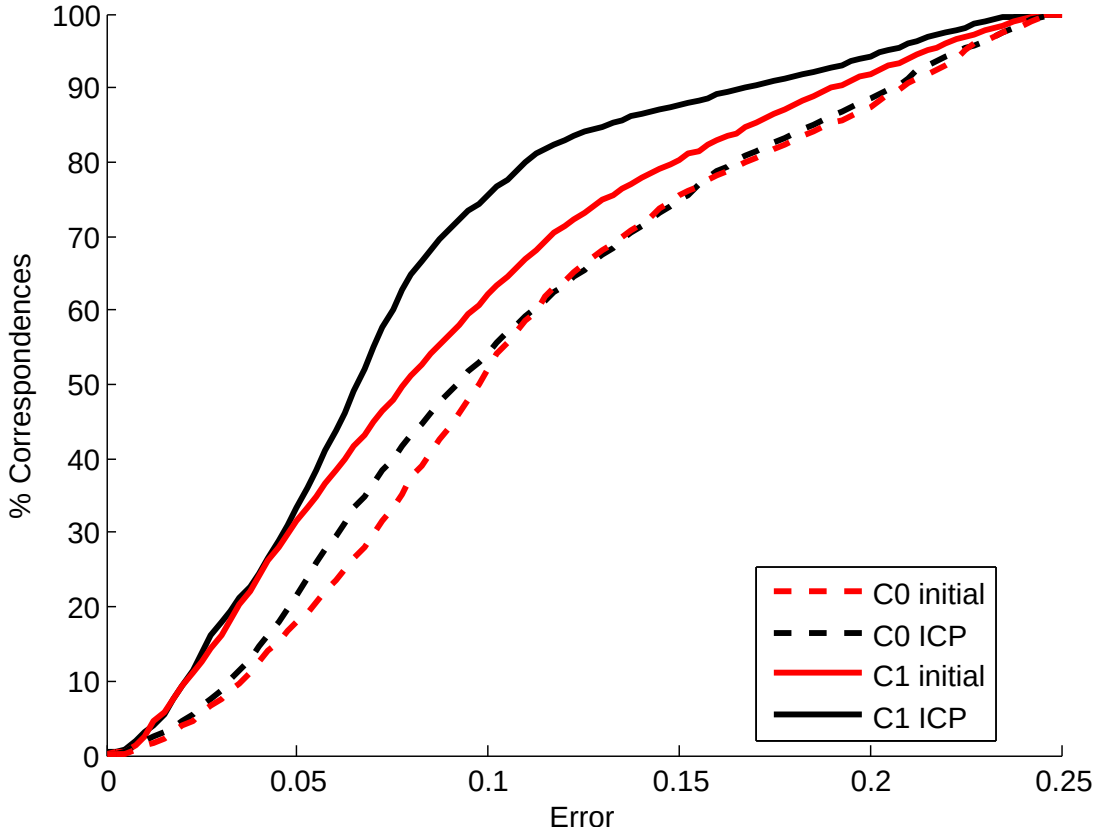


Figure 4.5: Tests on the Bunny-to-sphere sequence dataset with different frame distances between pairs. Top: average case; Bottom: pairs with distance of 150 frames.

4.5.2 Isometric benchmarks

We have also evaluated our vector field map method on two benchmark datasets, SCAPE Anguelov et al. [4] and TOSCA Bronstein et al. [17], which are the only large public domain isometric benchmarks that provide the ground truth point-to-point mapping information, to the best of our knowledge. For either dataset, we tested the method on around 400 pairs under near-isometric deformation. In this evaluation, the constraints and parameters for both C^0 and C^1 methods are identical, based on the landmarks specified in the datasets: 36 landmarks for SCAPE; 21 on animals, 36 on humans and 46 on centaurs for the eight groups in TOSCA. With eigenbasis size of 50, we initialize C^0 and C^1 with the landmark constraints and the commutativity regularization term, and then refine the C^0 and C^1 by their corresponding ICP for 5 iterations. Figures 4.6 show that our C^1 method performs properly under the same constraints induced from C^0 method, at the same cost with the C^0 representation, with a small amount of improvement in the quality of correspondence.

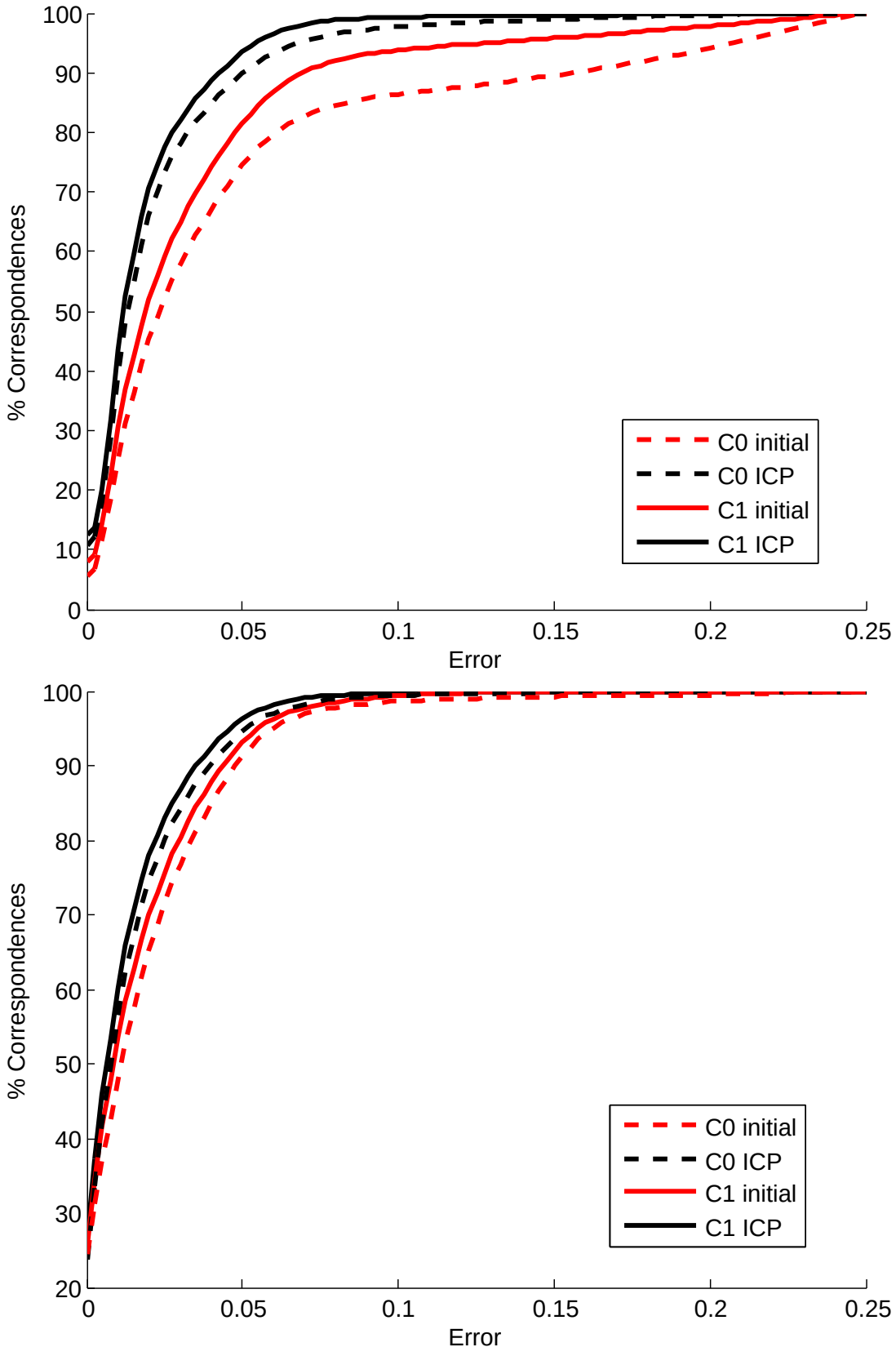


Figure 4.6: Comparison of our method (C^1) with the original functional map (C^0) method. The top error plot is for SCAPE dataset and the bottom error plot is for TOSCA dataset.

4.5.3 Segment correspondences

As with the functional map representation, segment correspondences can be used. With the vector field map, we can even use it for the non-isometric cases. Figures 4.7 presents point-to-point correspondence inference under such segment constraints (9 pairs) on the models from the TOSCA dataset Bronstein et al. [17]. Note that C^0 using unweighted Laplacian suggested in Ovsjanikov et al. [86] (the one without inverse of area element $\star_0$) produces much worse results, even for identical shapes with different connectivity, so we always use the weighted Laplacian. Near conformal mapping again shows fewer artifacts, compared to the more restrictive isometric mapping. The error plot for the cat and dog pair is shown in top of Figure 4.9. In this case, since there is no actual ground truth matching, we generate an alignment using the deformation technique of Sumner and Popović [104] using landmarks. More non-isometric pairs can be found in Figure 4.8.

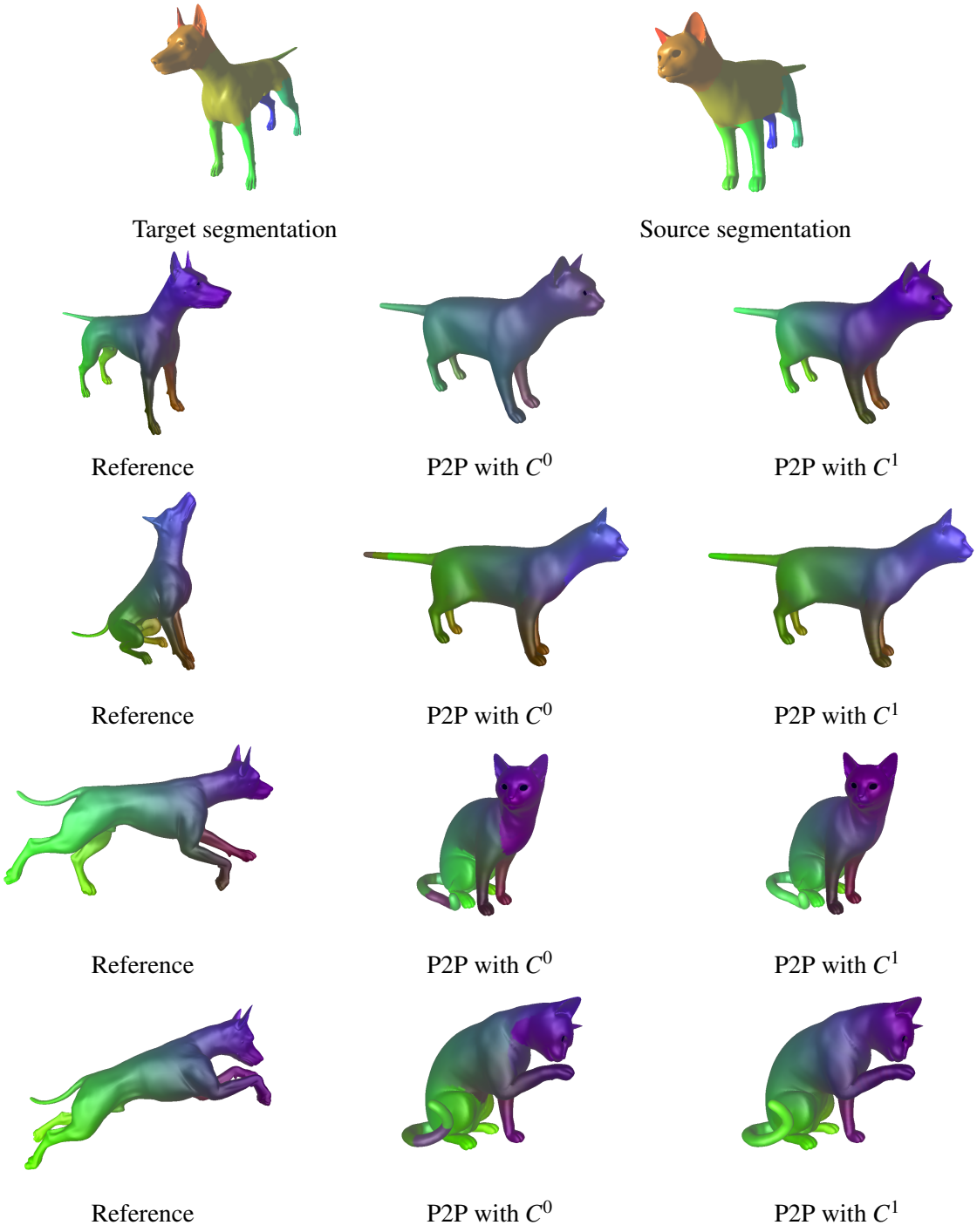


Figure 4.7: Comparison of P2P mapping with only segment constraints. The discontinuity on C^0 induced map exhibits the incompatibility with non-isometric deformation.

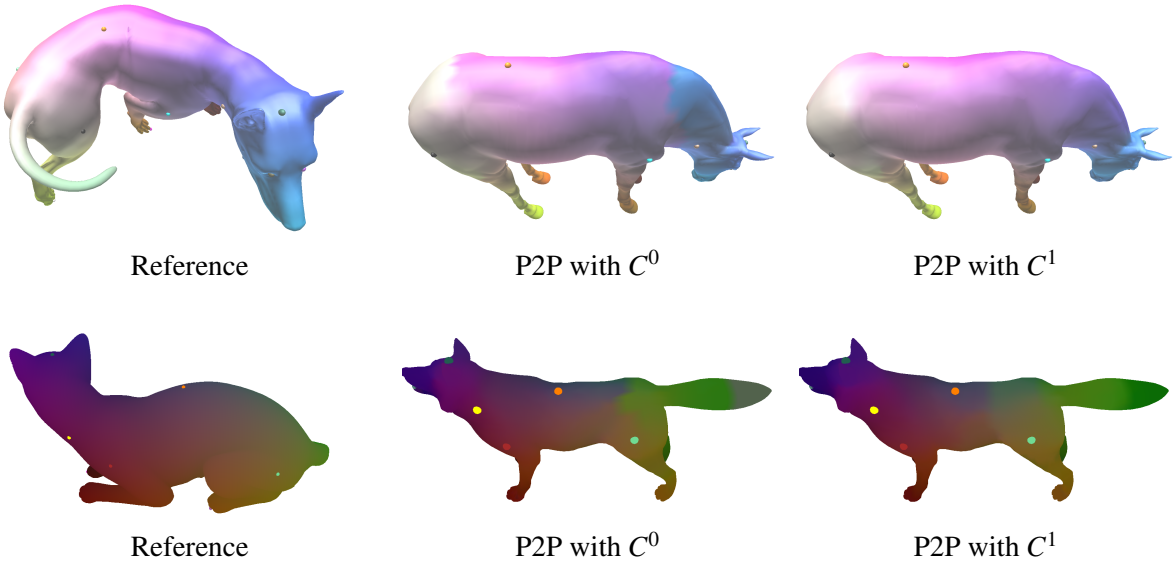
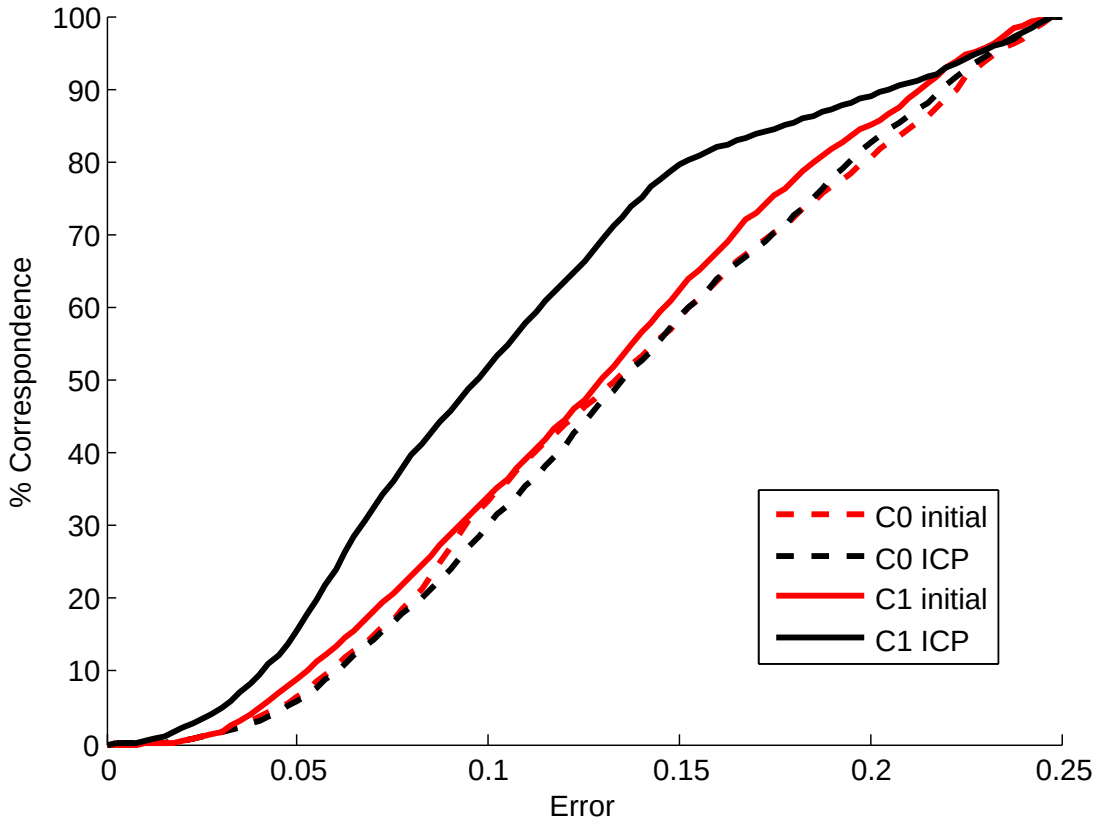


Figure 4.8: Comparison of more TOSCA non-isometric models: the dog to the horse (left); the cat to the wolf (right).

Functional map produced one of the best results for automatic correspondence inference through segment correspondences when no landmarks are available Ovsjanikov et al. [86]. We performed another test based on the same Heat Kernel Signatures (HKS) descriptor functions Ovsjanikov et al. [85]. With segment-based HKS constraints and commutativity regularization, we evaluated C^1 and C^0 methods under the same condition on SCAPE dataset. The result is shown in the bottom error plot of Figure 4.9. Once again, we show slight improvement even for this isometric dataset.



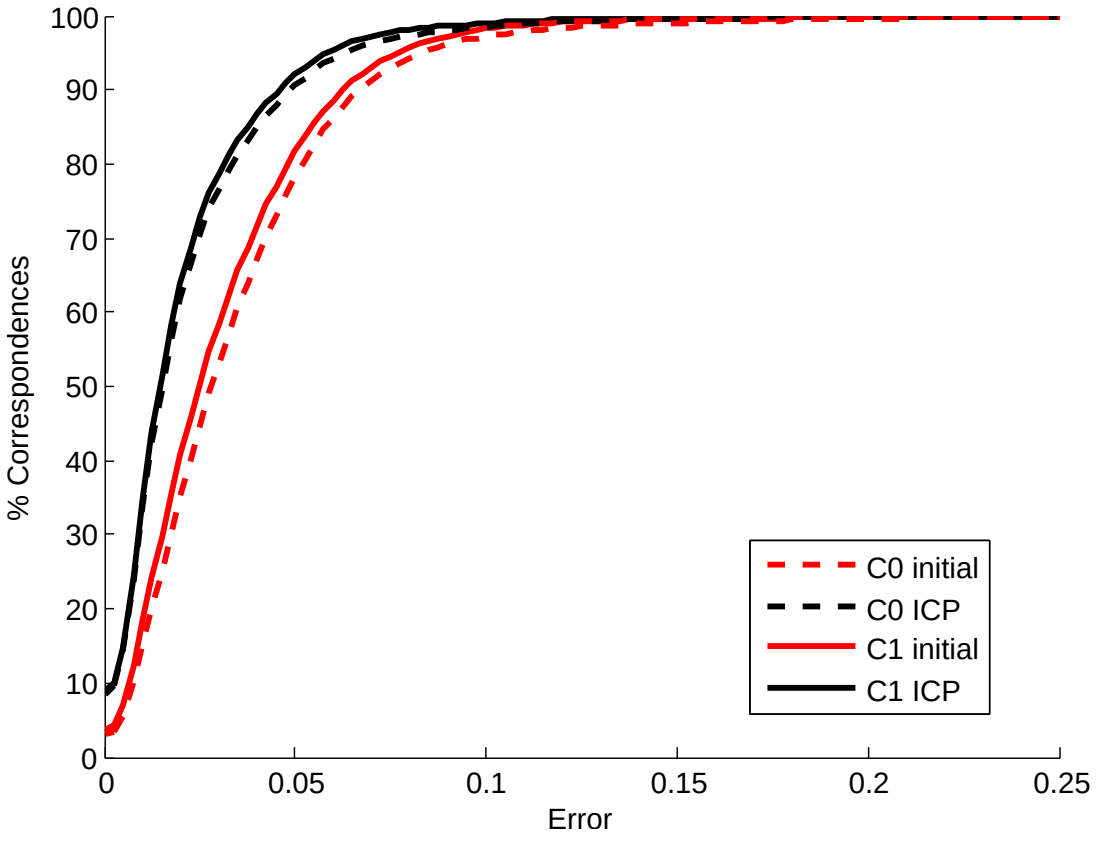
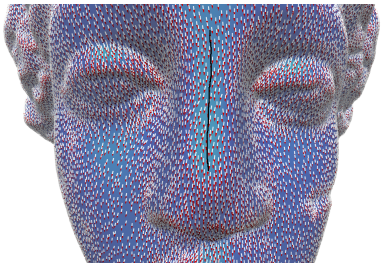


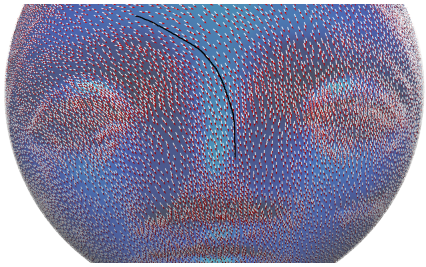
Figure 4.9: Segment constraints for non-isometric pairs (top) and for automatic correspondence inference (bottom).

4.5.4 Vector field guided alignment

Sometimes, it is unnecessary that a certain landmark pair should correspond to each other or a feature curve should correspond to another when designing shape correspondences, but an overall direction field should correspond to another direction field on the target mesh. In Figure 4.10, we provide a proof-of-concept example. We first created a correspondence without considering a direction alignment. Then, the user designed two vector fields using intuitive tools by sketching strokes, and generated vector fields through the method in Fisher et al. [36]. With this additional constraint, the desired alignment can be achieved.



vector field on source mesh



vector field on target mesh

without the vector field constraint.

with the vector field constraint.

Figure 4.10: Here we demonstrate the effect of our new type of constraint applied on vector fields directly. The map from the Venus head model to the sphere was first established based on 4 landmark pairs. The user can further specify the correspondence of certain direction fields. For easy manipulation, the source mesh venus head was shown on the sphere through the first correspondence to help the editing process. We then add an additional constraint for C^1 and produce the expected modified result.

4.5.5 Function transfer across different topology

We demonstrate in Figures 4.11 function transfer for shapes with different topology. Transfer functions through C^0 induced by C^1 is sufficient for shapes with the same topology, but not in this case, since the LB-basis structure changes significantly with a topological change. Even low frequency eigenfunctions can be very different. We can take advantage of the gradient domain technique: setting the precise scalar function values at the corresponding landmarks as the boundary condition, the process can be implemented through Poisson reconstruction of the transferred gradient of functions, i.e., solving a Poisson equation with the right hand side set to the divergence of the transferred gradient vector field. With the scalar values set directly at the landmark locations using the known values on the corresponding landmark locations on the source mesh, we show that the functions are transferred smoothly, since we only transfer the low frequency gradient fields. Directly using the Laplacian of the transferred scalar function using the function method for a reconstruction with the landmark used as boundary condition does not lead to similar results, probably due to the drastic change in area often associated with topological changes. The same set of manually selected landmarks is used for C^0 and C^1 in this test, 8 for the bunny and the torus, and 11 for the dog and the kitten.

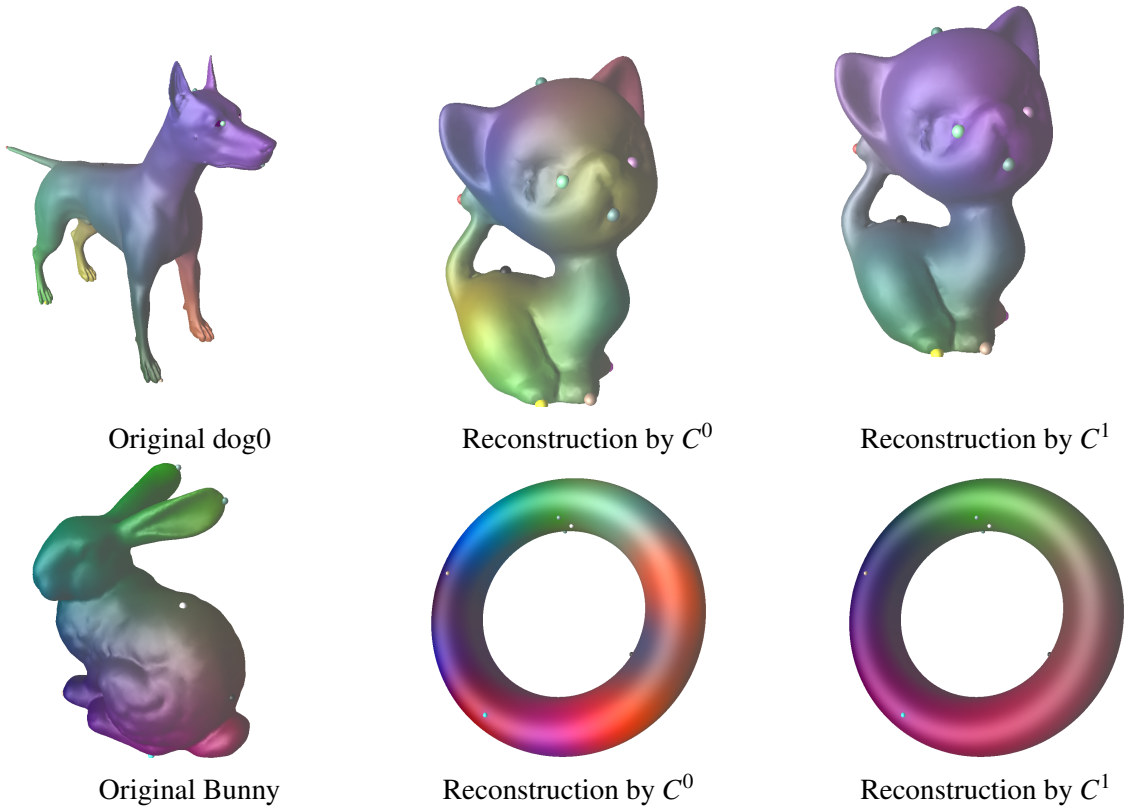


Figure 4.11: Comparison of Poisson reconstruction for function transfer between shapes with different topology.

4.5.6 Implementation details

All our tests were run on a Windows 7 system with Intel Core i7@2.8GHz and 12GB RAM). The efficiency of C^1 representation matches exactly that of C^0 representation, including the construction of the eigenbases, since the eigenbases used for vector fields are derived from the one for scalar fields. Conversion among C^0, C^1, and C^2 introduces negligible overhead. The most costly step by far is the kd-tree search used in conversion from C to P2P correspondences. We use kd-tree implementation on Matlab Central by Michael [76], which seems not as optimized compared to the timing reported in Ovsjanikov et al. [86], and typically runs for about 30 seconds for a P2P conversion for a mesh with around 25K vertices. We use the Kabsch algorithm [54] for finding the best rotation in high dimensional space, with a simple modification to allow reflection, as the

eigenfunction pairs may happen to be in different signs. The only weighting term in solving for the initial C^1 or C^0 is the one for the regularization term w. The results remain similar unless w is different by orders of magnitude, so we stick to the simple setting described in Sec 4.4.2. We also observe not much further improvement in results when more than 5 steps of ICP is performed, so in all our tests, we opt to use 5 iterations. We have also consistently used 50 LB eigenfunctions, since it is both efficient and without much loss in accuracy compared to representations that we tested with a higher number of basis functions.

4.5.7 Limitations

The point-to-point map obtained through kd-tree search or indicator function transfer is in fact a discrete vertex-to-vertex map. Such correspondences do not map a vertex to its best matching location, which normally would be a point within a certain triangle of the other mesh. Without an additional relaxation process, this type of correspondence cannot be directly used for operations such as texture transfer. However, in the case of genus-0 surfaces or patches, we expect that using a sphere as an intermediate shape can allow cross parameterization between any pair with post-processing through smoothing on the sphere or disk. Another issue with the functional map or the vector field map representations is that thin long features (high-frequency structures) can be mapped to collapsed curve-like structures on the target mesh when a small number of frequencies are used, which is perhaps the price to pay for representing the mapping with such a low dimension matrix. A decomposition of the entire shape into several simpler components, and performing component-wise function or vector field maps may alleviate the problem. We dropped the harmonic vector fields in the correspondence inference, as the the one-to-one correspondence can be determined by C^0 while C^0 only depends on the mapping among gradient vector fields. However, for objects with different topology, no one-to-one correspondence exists, so it might be a useful

control tool to exploit the correspondences among harmonic vector fields and other vector fields.

4.6 Conclusion

We have demonstrated that our vector field map representation facilitates correspondence inference for the important case of near-conformality (angle-preserving maps). Most important constraints on correspondence can be expressed as linear constraints, and conformality is just the orthogonality of the proposed map representation, which leads to a simple algorithm including a least-squares method followed by an ICP. We also demonstrate the possibility of extending the method to change of topology through Poisson reconstruction of the transferred gradients. The spectral analysis on surface meshes developed for the representation is also an efficient tool in handling vector fields in general.

Chapter 5

Parametrization-based Automatic Hexahedral Meshing

Recent development in quadrangulation has stirred new research effort in parametrization-based on volumetric meshing methods guided by the cross-frame fields, fields of equivalence classes of local frames under the chiral cubic symmetry group.[i] Such methods first create a parameterization of the volume for 3D charts intersecting at common interfaces, followed by extracting the vertices of the hex mesh from the integral points in the parameter domain. The edges of the hexahedra in the mesh would then follow the gradient lines of the parameterization. For computational purposes, the boundary of the resulting mesh must conform to the original boundary, create patches sharing the same integer parameter value over smooth regions, and introduce sharp edges and corners near the original features of the mesh. Thus, feature alignment and angle distortion reduction are both linked to a high-quality frame field with one of the axes aligned to the boundary normal on all surface points. Combining the global volumetric parameterization with compatible

[i] The set of 24 rotations that keep a cube centered at origin invariant without changing the right-handed frame to a left-handed one.

rotational and translational transitions as specified in the CubeCover method with automatic generation of guidance fields, one would imagine that an automatic hexahedral meshing tool would be straightforward to construct. However, unless manually constructed or adjusted, automatic generated frame fields do not usually satisfy these compatibility conditions for non-degenerate clean volumetric parameterization.

In this chapter, we represent the singularities in the global parameterization guided by a frame field as a graph, and present a systematic treatment of singularity that would lead to degenerate cell. Finally, based on the modified singularity graph, we present an efficient hexahedral mesh generation through a topological simplification of the volume.

5.1 Related Work

As mesh generation has been an active research topic for a long period of time, a number of methods have been proposed to generate high quality triangle or tetrahedral meshes automatically [72, 78]. Most of them rely on Delaunay tessellation (dual of Voronoi diagrams), which ensures proper connectivity and element quality. However, quadrilateral or hexahedral remeshing is substantially more difficult because there is no such dual structure for non-simplicial elements. In 2D manifold quadrangulation, some recent works [32, 46, 119] use Morse-Smale Complexes (MSC) [33] to guarantee a pure quadrangulation based on a scalar function with periodic property. Unfortunately, 3D MSC does not necessarily lead to hexahedral structures. As a consequence, many hexahedral remeshing methods, such as sweeping [99] and paving [102] rely heavily on manual input to define the proper topological structure by decomposing the volume into simple parts. Such challenges make grid-based methods [103] or hex-dominant mesh-based methods [112, 64] often the practical choice for computation performed on automatically generated domains in spite of

their inferior results as compared to those obtained with hexahedral meshes.

Recent progress shows that global parameterization has achieved great success in surface quadrangulation [55, 11] with the help of a smooth frame field defined on a manifold surface with compatible transition functions between charts. From such a non-degenerate parameterization, an all-quadrilateral mesh can be extracted by tracing the iso-lines. To extend such a scheme to hexahedral remeshing, [81] has proposed a global parameterization method for hexahedral remeshing, and [48] gives a method to automatically construct a desired frame field. These two works are most relevant to our method. Roughly speaking, we attempt to apply a parameterization method in CubeCover to the automatically generated frame field. However, the task is extremely challenging as the original frame field optimization does not take into account the conditions on admissible singularity types (both inside the volume and on the boundary). In addition, without a coarse meta-mesh manually specified or generated from a manually specified frame field, a topologically sound partitioning of the volume for a global parameterization is not so straightforward to generate as one might expect. Gregson et al. [42] propose a method for all hexahedral remeshing with the topological restriction of no internal singularity, thus leaving few degrees of freedom to optimize the shape of hexahedra. As noted in [81], it is often necessary to move the right angle transition line in the parameter domain on smooth regions of the surface into singular edges inside the volume to have better element quality, as the former turns the dihedral angle between a hexahedron's two faces into nearly 180°, while the latter only turns the sum of three such dihedral angles around the internal edge to 360°.

Singularity plays a critical role in detecting and remedying this issue. Shepherd [96] lists many topological restrictions on primal elements of a hexahedral mesh, such as nodes, edges and faces, and uses these restrictions in a frame-field-independent remeshing algorithm. Some methods have been proposed to analyze the global topological structure of a quadrilateral mesh or a 2D symmetry

vector field by singularity graphs [107, 88]. In 2D quadrangulation, noisy singularity points lead to problematic parameterization results. Thus, some remedies have been proposed for denoising or adjusting the singularities in the frame field [55]. For hexahedral remeshing, Nieser et al. [81] provide a definition on internal singularity types and prove some of its properties. However, there is no existing work on automatic surface singularity and internal singularity adjustment for frame fields used in hexahedral meshing, except a concurrent work on hexahedral meshing by Li et al. [66]. However, their method does not address the potential issues involving surface singularities, and may require manual adjustment of the frame field to produce non-degenerate parameterization.

5.2 Overview

Algorithm 2: Hexahedralization process overview.

Data: tetrahedral mesh Ω with one frame F per tet computed by, e.g., Huang et al. [48]
Result: hexahedral mesh with edges guided by F
Compute the singularity graph (Section 5.3);
while *there is a zigzag (Section 5.4.2)* **do**
    Remove zigzag by "straightening"
while *a compound edge (Section 5.4.3) adjacent to two admissible edges exists* **do**
    Split the compound edge:
    turn it into an admissible type;
    create a separate singularity path through vertices newly introduced in a local refinement;
Adjust the frame field (Section 5.4.4);
Reduce the number of integer variables
(used in boundary and transitions, Section 5.5.1);
while *an untreated integer variable (Section 5.5.2) exists* **do**
    minimize the parameterization energy;
    round the integer variable closest to an integer;
    turn the variable into a constant;
Minimize the parameterization energy;
Construct hexahedral meshes with integer grid points;
Postprocessing if necessary (Section 5.6);

Our work focuses on generating all-hexahedral meshes from input cross-frame fields generated

automatically by the method proposed by Huang et al. [48]. Given a tetrahedral mesh Ω, each tetrahedron is associated with an orthonormal frame $F = (U, V, W)$, with U, V, and W as the basis vectors. The frame is considered as a representative of a cross-frame (an equivalent class of 24 orthonormal frames with axes chosen from $\{U, -U, V, -V, W, -W\}$). To generate an all-hexahedral mesh with edges following such an input, we compute a parameterization with a method similar to the one proposed in CubeCover[81]. We loosely follow their notations below.

We denote the parameterization $f : \Omega \to \mathbb{R}^3$, which can be expressed in each chart (tetrahedron) as a linear function with three components $(u, v, w)^T$. In tetrahedron t, we denote its expression by f_t.

If we use integral lines of the gradients of the parameterization for edges, they must connect to each other across the tetrahedron boundaries. Thus, in order to obtain a parameterization whose integer grid points can be used in hexahedralization, the transition from f_s to f_t, for two adjacent tetrahedra s and t, must satisfy

$$f_t = \Pi_{st} f_s + g_{st}, \tag{5.1}$$

where $g_{st} \in \mathbb{Z}^3$ and $\Pi_{st} \in \mathbb{O}$, the set of one of the 24 matrices for the cross frame containing the standard identity frame, a.k.a. the chiral octahedral symmetry group. Comparing with the transition from t to s, we have

$$\Pi_{st} = \Pi_{ts}^{-1}, \quad g_{st} = -\Pi_{st} g_{ts}. \tag{5.2}$$

We call any face with a non-identity rotational transition or non-zero translational transition a jump-face.

For a surface triangle a associated with tetrahedron t, to avoid cutting the corresponding hexahedron by the boundary surface, one of parameter must be an integer, with its gradient aligned to

the surface normal. Thus, there is a transition Π_{ta} that makes the parameterization coordinates of a surface point $p \in a$ satisfy

$$(\Pi_{ta} f_t(p)) \cdot (1,0,0)^T = g_{ta}, \tag{5.3}$$

where $\Pi_{ta} \in \mathbb{O}$ aligns the U-axis of the frame to the normal of a, and $g_{ta} \in \mathbb{Z}$.

The translational part (the gap) g_{st} can be resolved during the parameterization process if the hex edge size can be adjusted, but the rotational part Π_{st} must be properly prescribed to avoid fold-overs and degeneracy in parameterization while following the given frame field.

For the parameterization with its gradients following the frame field to be smooth, a natural requirement for the rotational transitions is that they make the angles between the corresponding axes in F_t and $F_s\Pi_{st}$ respectively as small as possible. If one of these angles becomes greater than $\pi/2$, it can lead to large angle distortion in the parameterization. In addition, more constraints are required to avoid the defects caused by singularities as shown in the next section.

5.3 Rotational Transition and Singularity

A straightforward method to evaluate the rotational transition is to find the best transition matrices that minimize the following "alignment error":

$$\begin{gathered} \|F_t - F_s\Pi_{st}\|, \\ \|(F_t\Pi_{ta})(1,0,0)^T - n_a\| \end{gathered} \tag{5.4}$$

where n_a stands for the normal of the surface triangle a in t, and the matrix norm is the Frobenius norm, which is used throughout the rest discussion. For a tetrahedron t containing more than one boundary faces, e.g., a and b, $\Pi_{ta} \neq \Pi_{tb}$ or t should be split into two. When the rotation

Π_{ta} minimizing the above error is non-unique, we choose the one fixing one of the axes. Such a setup is necessary for the rotational transition of the gradient fields of the parameterization to be smooth while remaining consistent with the rotational transition of the guidance frame fields [81]. However, it does not guarantee degeneracy-free parameterizations in the presence of certain types of singularities as discussed below.

5.3.1 Internal Singularities

If the valence (number of adjacent hexahedral cell) of an internal edge in the final hexahedral mesh is not 4, it is an internal singularity. Such singularities can be detected by checking the rotational transitions. As formulated in [81], for an oriented tetrahedral mesh edge e, surrounded by a small counter-clockwise oriented dual edge loop passing through tetrahedra $(t_0, t_1, \cdots, t_k, t_0)$, we can define the type of the edge with respect to starting tetrahedron t_0 as

$$\text{type}(e, t_0) = \Pi_{t_0 t_1} \Pi_{t_1 t_2} \cdots \Pi_{t_k t_0}. \tag{5.5}$$

If $\text{type}(e, t_0) \neq I$, we call it an *internal singularity edge*.

An internal singularity can lead to a non-valence-4 segment in the final hexahedral mesh, only if $\text{type}(e, t_0)$ is a rotation around one of the axes in the frame. Other types of singularities are said to be *compound*, and forced to be mapped to a point in any parameterization producing a hexahedral mesh consistent with the transitions, leading to degeneracy. It can be shown by contradiction. Suppose that the image of e in the parameterization is not a point. If we choose a sufficiently fine hexahedral mesh, the image is either parallel to hexahedral faces, or intersecting with a hexahedral face. For a compound singularity edge e, U, V, or W is not an eigenvector of $\text{type}(e, t_0)$, so its image cannot be parallel to one of the gradient lines of the parameters. On the other hand, the

hexahedral face that it intersects with contains edges along U, V, or W, which cannot form a loop consistent with the accumulated rotational transition type(e, t_0), leading to a contradiction.

The additional requirement on the inner singularity vertex (Theorem 2.2 in [81]) through counting valences would be satisfied automatically when the rotational transitions form the minimizer of the frame field transitions. Although valence 3 and 7 (or 4 and 8) cannot be distinguished by edge type, valence 7 (or 8) edges do not appear in automatically generated frame fields in practice.

5.3.2 Surface Singularities

In Section 5.4, we will show that internal singularities may lead to degeneracy in parameterization because of the constraints that they impose. For surface edges, it is also possible that they introduce constraints that cause degeneracy. Such constraints force two components of the parameterization coordinates along the edge to be both constant integers, i.e. the edge must be along some edges of the hexahedral mesh. We call such surface edges *surface singularities*, which are formulated mathematically below, with a definition consistent with the internal singularities.

Similar to the internal case, for an oriented surface edge e, we also create a small counter-clockwise oriented loop around it, which passes through $(a, t_0, \cdots, t_k, b, x, a)$ (e.g., Figure 5.1), where the faces sharing e, a and b, are the triangles adjacent to tetrahedra t_0 and t_k, respectively, and x stands for a point outside of the tetrahedral mesh. We then define the type of the edge as

$$\text{type}(e, a) = \Pi_{ba}\Pi_{t_k b}\Pi_{t_{k-1} t_k} \cdots \Pi_{t_0 t_1}\Pi_{a t_0}, \tag{5.6}$$

where Π_{ba} is a rotation around axis U aligning the V, W-axes on triangles a and b when U-axis on

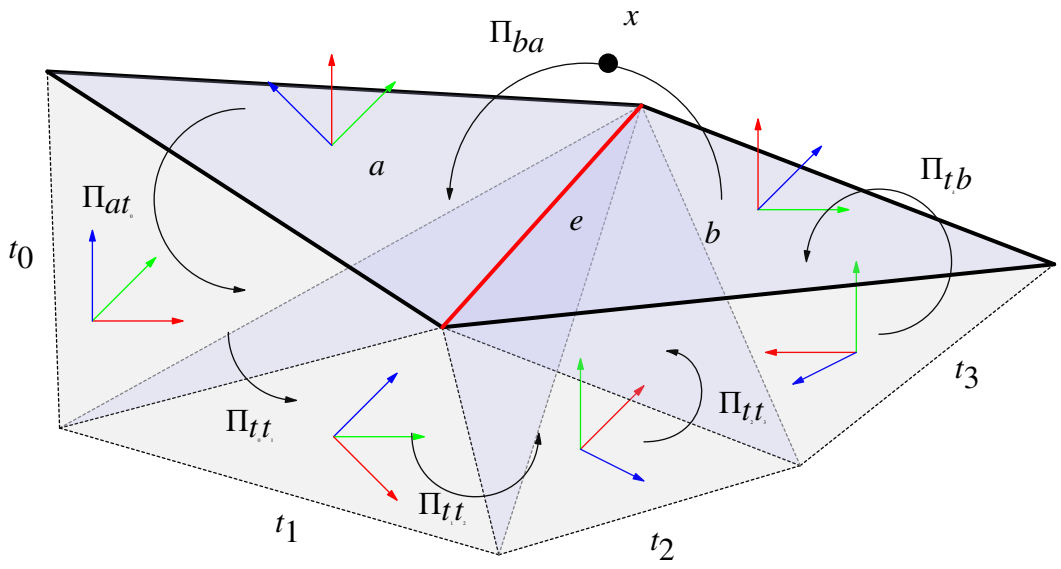


Figure 5.1: Surface singularity example.

each triangle is aligned to the normal. More precisely, it minimizes

$$\|Re(n_a, n_b)F_{t_0}\Pi_{t_0a}\Pi_{ab} - F_{t_1}\Pi_{t_1b}\|,$$

where $R_e(n_a, n_b)$ is the rotation around e that aligns n_a (normal of a) to n_b (normal of b), which flattens the hinge formed by the two boundary triangles. If type(e, a) is a rotation around an axis by $\pm\pi/2$, it creates a sharp edge on the surface in the parameter domain, and denotes an admissible singularity. In other words, the valence of such an edge in the hexahedral mesh would be different from 2. If it is not a rotation around one of the frame axes, we have a *compound boundary singularity*.

The transition between surface faces Π_{ab} describes a rotation on V, W-axes in the parameter domain. The product of Π_{ab}'s along a loop around a surface vertex can be used to detect singularity vertices on the surface, which form sharp corners in the parameter domain with a discrete Gaussian curvature (angle defect) of multiples of $\pi/2$. Only $\pm\pi/2$ would lead to reasonable sharp corners. If

the angle defect becomes π, the parameterization would wrap two surface squares around a surface vertex, leading to degeneracy. However, such a degeneracy never occurred in all our experiments based on automatically generated guidance fields.

5.4 Inadmissible Singularities

The whole *singularity graph G* is composed of the internal and surface singularity edges, and the vertices incident to these edges. The following two issues in such a graph will lead to degeneracy in the parameterization, and are thus necessary to eradicate:

- Compound singularity (a rotation not around any axis in the frame): It maps to a single point in the parameter domain, thus inducing degeneracy. In addition, a surface rotation around a frame axis by π is also treated as a compound singularity, because it indicates zero or 4 neighboring hexahedra, leading to great distortion except for the case with a valence-4 boundary edge whose adjacent boundary faces have nearly opposite normals.

- Zigzag (two same type consecutive singularity edges in one tetrahedron): Denote the edges by e_0 and e_1, and the tetrahedron by t, $\text{type}(e_0, t) = \text{type}(e_1, t)$. Such edges map to a straight line in the parameter domain, thus making the image of the tetrahedron degenerate.

These issues can be viewed as resulting from improper discretization of a smooth frame field with only admissible singularity types. A compound singularity can be viewed as several close-by singularity lines merged onto one tetrahedral edge. A zigzag can be explained as misalignment of the edges of tetrahedral mesh with the singularity line in the smooth frame field. Unless a manually constructed or adjusted frame field is used, these issues almost always exist and must be addressed.

Based on the above observation, we propose a two-step method to schematically adjust the initial rotational transition set derived from Equation 5.4 to address the above issues. In the first step, we remove all the zigzag issues by straightening the singularities. In the second step, we split the compound singularities into admissible ones.

Additional care may be necessary to avoid extreme angle distortion associated with high hexahedral edge valences. However, for internal singularity edges, the valences are in most cases less than 6 when the singularity types are derived from the automatically generated guidance frame field. For boundary singularity edges, one may want to make sure that the dihedral angles do not deviate far from the dihedral angles specified by the singularity type in the parameter domain. Similarly, the boundary singularity vertex should have compatible Gaussian curvature in the parameter domain and on the mesh to avoid extreme angle distortion at the sharp corners.

5.4.1 Atomic Operation to Adjust the Singularities

For an internal face shared by tetrahedra s and t, it is an atomic operation of modifying the rotational transition Π_{st} into $\Pi_{st}\Pi$ by a matching matrix Π, since the face only affects the types of its three edges. The type of one of these three edge type(e, s) that is counterclockwise oriented around the direction of $s \rightarrow t$ becomes type$(e, s)\Pi$ after the modification. Thus, the types of all these edges change by a same rotation. It can be likewise applied to surface singularities. We use this simultaneous change as the basic operation to remedy the issues in the singularity graph.

5.4.2 Zigzag Removal

As shown in the Figure 5.2, the triangle between tetrahedra s and t contains a zigzag $\overline{pxq}$ with type X. Changing the rotational transition Π_{st} into $\Pi_{st}X^{-1}$ turns the edges $\overline{px}, \overline{xq}$ into regular

edges. The type of the third edge $\overline{pq}$ in this triangle with original type Y will become XY. Such an operation can be viewed as straightening the singularities $\overline{px}, \overline{xq}$ into $\overline{pq}$, and superposing their type onto the original type of $\overline{pq}$. If the original type of edge $\overline{pq}$ is not I, its type may become compound singularity after removing the zigzag, which will be split into admissible ones in the second step of our method.

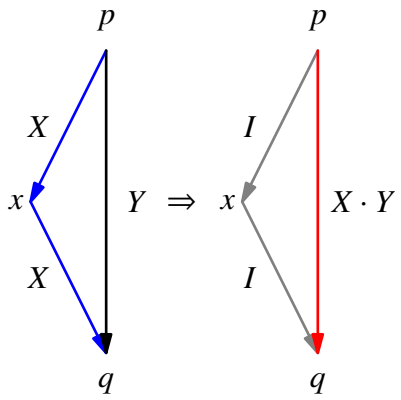


Figure 5.2: Removing a zigzag $\overline{pxq}$. The gray edge is regular, and the black edge is possibly singular.

Removing a zigzag may introduce new zigzags, but the procedure always terminates when we repeatedly find a zigzag to remove until all zigzags are eliminated. The number of zigzag removal operations to be performed cannot exceed the total number of singularity edges in the graph, since that number decreases by either one or two in each operation.

5.4.3 Compound Singularity Split

After removing all the zigzags in the graph, the remaining inadmissible singularities are compound singularities. To remove a compound singularity, we split it into multiple admissible singularities inside its one-ring neighborhood without affecting other singularities or introducing zigzags (Figure 5.3). During this procedure, tetrahedra may be split into smaller ones through local refinement.

In the refinement steps, we keep the rotational transition on split faces, and set the rotation transition to identity for the newly introduced faces. Thus, the split edges share the same type as the original ones, and the newly introduced edges are all regular edges.

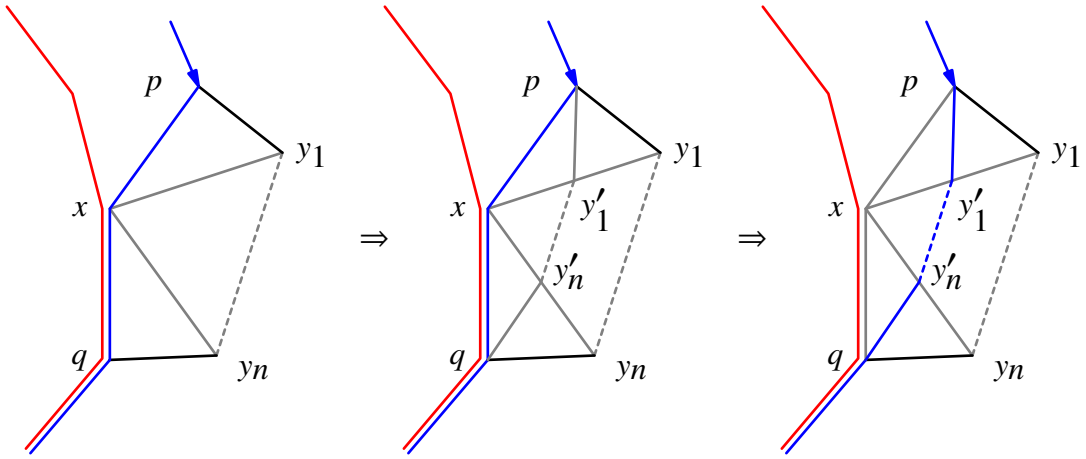


Figure 5.3: Splitting a compound singularity edge $\overline{xq}$. The gray edges are regular, and the black edges can be singular.

We first find an open end node x of a compound singularity polyline, i.e. a node connected to only one compound singularity edge. In its one-ring neighborhood Ω_x, we pick one node p on one adjacent admissible singularity, and another node q on the compound singularity. Then, we subdivide each triangle on the boundary of Ω_x into four triangles and its associated tetrahedron into four tetrahedra, by inserting one new node at the middle of each boundary edge. As the newly introduced nodes y_i's are connected, and each original node is adjacent to some of them, we can always find a path of the form $\overline{py_1y_2\cdots y_nq}$. In particular, we can pick the shortest one. These edges and x form a fan of triangles (in the same orientation), with each edge $\overline{xy_i}$ being regular.

To avoid re-introducing zigzag edges when removing the complex singularity, we further subdivide the triangle fan $x, \overline{py_1y_2\cdots y_nq}$ as follows. Each edge $\overline{xy_i}$, along with each of its incident tetrahedra in the one-ring, is split into two by inserting a new node y'_i at the middle. Then we get a new path $\overline{py'_1y'_2\cdots y'_nq}$ completely inside the original one-ring neighborhood, with each of its

edges initialized to the identity type. We modify the rotational transition on all the triangles in the fan composed of x and $\overline{py'_1y'_2\cdots y'_nq}$ by multiplying the inverse matrix of the edge type for $\overline{px}$, turning $\overline{px}$ into a non-singular edge. Then the polyline $\overline{py'_1y'_2\cdots y'_nq}$ contains all admissible singularity edges with the same type of $\overline{px}$. The vertex x is now incident to only two singularity edges, with one of them the original incident admissible singularity. According to following Proposition 5.4.1, the other edge $\overline{xq}$ must also be an admissible singularity.

Proposition 5.4.1. *If a vertex is incident to only two singularity edges, the types of the two edges must match in a certain local alignment.*

Proof. If vertex x is incident to only two singularities $\overline{px}, \overline{xq}$, we prove that the two share the same rotational transition in a local alignment.

First, we construct the local alignment by cutting the one-ring neighborhood $N(x)$ of x open to expose the two singularities onto the boundary. Similar to the situation in Figure 5.3, such a cut always exists. As shown in the inset, we denote the split nodes as $y_i, y'_i, i = 1, \cdots, n$. Then, the frames in every tet of

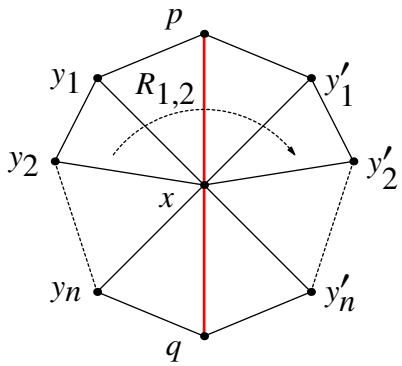


Figure 5.4: Illustration for the Proposition 5.4.1.

$N(x)$ can be aligned so that each internal face has identity as its rotational transition type. The only none-trivial rotation transitions are between the triangle pairs $\overline{xy_iy_{i+1}}$ and $\overline{xy'_iy'_{i+1}}$, where $y_0 = p, y_{n+1} = q$. We denote the transition from $\overline{xy_iy_{i+1}}$ to $\overline{xy_iy'_{i+1}}$ as $R_{i,i+1}$.

For each i, the edge $\overline{xy_i}$ before the cut is regular. Thus, for the small circle through the tetrahedra around the edge $\overline{xy_i}$ in the one-ring before the cut, the accumulated rotational tranisition $R_{i-1,i}R^{-1}_{i,i+1}$ must be I, i.e. $R_{i-1,i} = R_{i,i+1}$. Thus, the edges $\overline{px}, \overline{xq}$ must have the same singularities type $R_{0,1} = R_{n,n+1}$.

□

The above operation can be viewed as splitting and shifting the singular segment $\overline{pxq}$ to the new paths. It fails when there exist loops of compound singularity edges, which contain no end node. However, it never occurred in all frame fields automatically generated during our experiments. The monotonic decrease in the number of compound singularity edges within each iteration guarantees the termination of the procedure.

5.4.4 Frame Field Adjustment

The above singularity adjustment removes the compound and zigzag inadmissible edges as defined in the previous section, but it may increase the alignment error, leading to possible difficulties in the subsequent parameterization step. By minimizing the alignment error with respect to F, we can update the frame field according to the new rotational transitions. For robustness, we adopt a scheme that incorporates the non-linear constraints gradually. With the rotational transitions fixed to the adjusted matrices, we first obtain an initial guess by turning the problem into a linear system, minimizing Eq. 5.4, without the constraints $F \in SO(3)$, i.e. treating each F as an arbitrary 3 by 3 matrix. We then incorporate the orthonormality constraints on F as soft constraints, by including a non-linear penalty term $w\|F^TF - I\|^2$ (w is set to 100 in all our results), and solve the optimization problem again with Gauss-Newton method starting from the initial guess. The solution is converted to the closest ZYZ Euler angle representation, and serves as the initial value

to minimize the alignment error in such a representation. We then transform the solution back to the 3 by 3 matrix representation as the updated frame field satisfying the hard constraints $F \in SO(3)$. In most cases, the largest angle between the corresponding axes of two adjacent tetrahedra is about 30° when measured with the initial rotational transition, but increases to about 120° after adjusting the rotational transition to resolve compound singularity. After the above adjustment, it finally drops to about 75°.

In some cases, the resulting singularities may be close to each other, resulting in high distortion in the parameterization. However, a simple post-processing is discussed in Section 5.6 to reduce the distortion of the parameterization in the hexahedral elements between these singularities, producing high quality hexahedral meshes.

5.5 Parameterization

We follow essentially the same idea in the CubeCover method for parameterization, i.e. finding the minimizer of the following energy

$$\int_V \|df - F\|^2 \, d\, Vol,$$

which means that the gradient of each component of the parameterization should follow the corresponding axis of the given frame field F as much as possible. The main difference is that we are able to greatly reduce the number of integer variables through a simple preprocessing procedure, making the subsequent steps more efficient and robust.

5.5.1 Topological Simplification

We merge all the charts to get a global parameterization domain. To improve efficiency, instead of just creating a minimal spanning tree (MST) to remove a minimum set of transition variables across faces between tetrahedra adjacent in the tree as in [81], we additionally create a 3D domain with trivial internal topology to fill the volume, removing as many transition variables associated with faces as possible. Note that the original (possibly nontrivial) topology is encoded by the self-intersection of the new domain boundary. This can be achieved by slowly growing the domain to eventually occupy the whole volume, while maintaining a ball-like topology for the interior of the domain throughout the process. In Figure 5.5, we show a simple example of the topological simplification process for torus. Figure 5.5 (a) shows the arbitrary tetrahedron chosen as the starting cell. Figure 5.5 (b) and (c) are the first several steps of expanding the domain by merging neighbouring tetrahedra. Figure 5.5 (d) and (e) are the intermediate stages of the merging process. In Figure 5.5 (e), we render the faces (in green) only for illustrative purposes. They are not detected until the merging process is completed. Figure 5.5 (f) is the final remained faces that cannot be removed in order to maintain the ball-like topology.

After using MST to globally align the frames while keeping the singularity structure intact, we start from a seed tetrahedron, gradually remove faces with trivial transitions to merge tetrahedra, and expand the domain until all tetrahedra are merged to it. Any non-rotational-jump-face is a candidate for removal. If both adjacent tetrahedra of such a face are already merged to the domain, it can still be removed when it contains a non-singularity edge adjacent only to this unremoved face. Since all the faces corresponding to the MST edge can be removed, the entire volume can always to be merged into a single domain. However, an arbitrary order of traversing such face candidates can leave more faces unremoved than necessary, leading to more integer variables. We

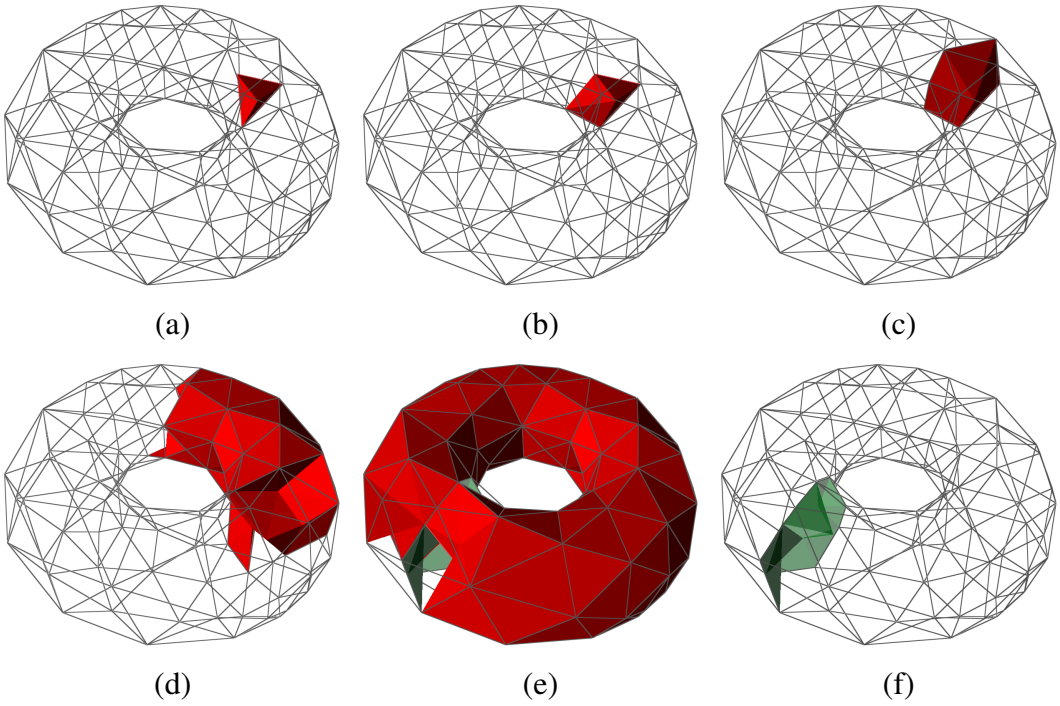


Figure 5.5: An example of topological simplification.

follow a simple rule when expanding the domain, under which all candidate faces around an edge on the boundary of the growing domain are simultaneously removed. This ordering will leave few rotational transition face patches, which, together with the at most P translational transition patches, cut the genus-P volume into a ball-like parameterization domain.

5.5.2 Parameterization as a Mixed Integer Programming Problem

We can easily cluster the remaining faces into patches with same transition types. Each internal patch uses only three integer jumps, and each boundary patch has one integer parameter. It is equivalent in theory to associate variables with each individual face, and then use Gaussian elimination to reduce the number of variables, or simply leave these variables in the final linear system with an increased number of equations. However, the Gaussian elimination process is significantly slower with a much larger number of constraints. Leaving the constraints as equations in the linear

system would turn them into soft constraints when we solve the over-constrained system using least-squares method. With the patches clearly identified, we can actually leave a small number of integer constraints only on the internal singularity lines and the boundary patches, and use three floating point values per internal patch. This approach can greatly reduce both the integer degrees of freedom of the system and the number of constraints, and thus expedite the process of the Gaussian elimination when we enforce the hard constraints.

The final system normally contains less than few tens of integer variables for a given input frame field. We successively snap one integer variable from the floating point value obtained in the previous solver in a greedy fashion. The linear system is assembled through first removing redundant hard constraints, and then eliminating them from the variables by substitution. This can be done on the gradient operator. After that, the reduced gradient operator and its transpose can be assembled as the reduced Laplacian operator for the optimization step. The construction process may take long if the model contains a large number of vertices, but the subsequent linear systems turned out to take little time for each solve when we call the sparse linear solver of Matlab.

With a singularity graph containing only admissible singularities, the parameterization may still contain degenerate or flipped tetrahedra initially, due to geometric distortion and numerics. In our experiments, less than five iterations of local stiffening are sufficient to eliminate the flipped tetrahedra and most of the degenerate tetrahedra. In the few cases when there are remaining degenerate tetrahedra, they most often do not actually lead to any issue in the mesh generation process described below. The only actual problematic degeneracies are all associated with internal singularities very close to the surface, which can be treated by a postprocessing procedure as detailed in the next section.

We use a simple approach to produce the final mesh. First, we generate the integer points $(u, v, w) \in \mathbb{Z}^3$ and half-integer points $(u + 1/2, v + 1/2, w + 1/2)$ within each tetrahedron, and

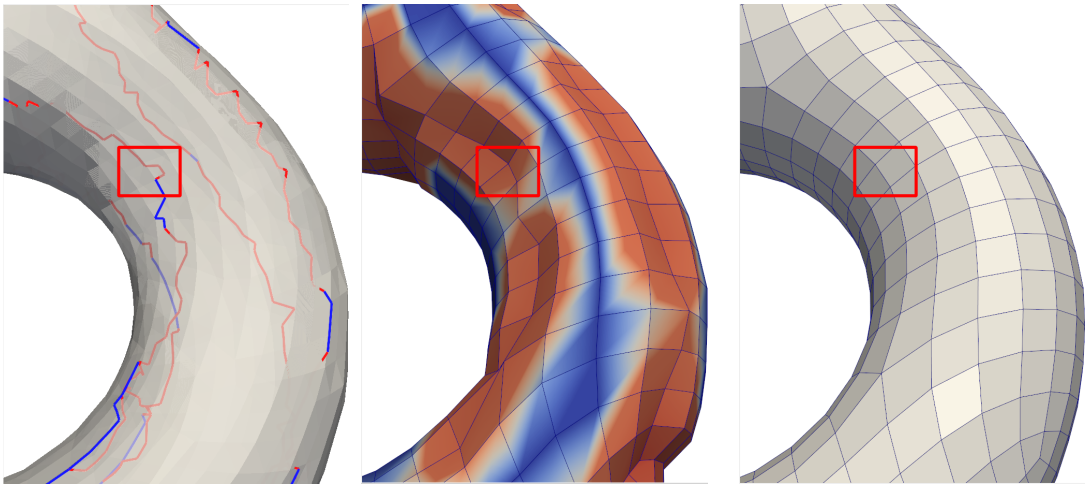
snap nearby points of the same type together through a spatial indexing structure to avoid potential duplicates near the faces. Then, starting from each half-integer point which serves as the center of a hexahedron, we follow the U,V,W directions to find the 8 corresponding corner vertices from the set of integer points and produce the connectivity information for the hexahedral mesh. During this process, we keep track of how far it needs to go in the parameterization domain, and the change of directions to handle any jump-faces encountered along the path.

5.6 Results and Discussion

The singularity correction method mentioned in Section 5.4 is able to resolve all the zigzag and compound singularity issues in our experiments. However, it remains an open problem to give a *sufficient condition* on singularity graphs compatible to non-degenerate hexahedral meshing.

In our experiments, some singularities close to the surface (near-misses) lead to degeneracy as well. As shown in Figure 5.6, some singularities sink right below the boundary surface of the tetrahedra mesh, and lead to degeneracies in regions between them and the boundary. As a result, the boundary of output hexahedral mesh at these singularities edges is not aligned with the input boundary.

Similar to zigzag and compound singularity, discretization of a singularity in a smooth cross-frame field which is close to, but not exactly on the surface (possibly due to the numerics of the frame field generation process) may lead to such singularities. This issue is inevitable when few assumptions can be made on the automatically generated cross-frame field. The difficulty of solving it lies in that such a degeneracy cannot be simply detected from the singularity graph. Fortunately, such sunk parts are all narrow and shallow (often less than two layer of tetrahedra) as they resulted from near-misses, and more importantly, the output hexahedral mesh still shares



(a) (b) (c)

Figure 5.6: The red singularity edges in (a) lead to degeneracy and the sunk part (in red) in (b), but the defects can be easily fixed by surface snapping and post-smoothing (c).

the same topology to the input tetrahedral mesh. Thus, we simply first snap the sunk nodes in the hexahedral mesh onto the tetrahedral mesh boundary by projecting them along the normal, and then improve the quality of the hexahedral mesh by a local relaxation (few iterations of Laplacian-based local smoothing).

We tested the proposed method on several models. The statistics on the models are given in Table 5.1 and Table 5.2. The timing is measured on a PC with an I7 940 CPU at 2.8 GHz and 12 GB RAM. To measure the dihedral angles of an edge in a hexahedron, the normal of each adjacent quadrilateral face is evaluated by averaging the four triangle normals in two different tessellations of the quadrilateral face. As shown in the table below, nearly all the elements are with decent shape quality measures.

Model	Tet	Comp	Zz-inner	Zz-surf	T(s)
Torus	30k	0	100	15	0.04
Nut	65k	0	0	5	0.10
Sphere	80k	0	26	10	0.05
Pretzel	300k	5	204	35	6.27
Sculpture	105k	0	8	1	0.11
Fandisk	301k	0	148	16	0.22
Elk	123k	12	318	58	5.04
CAD1	68k	0	36	2	0.07
CAD2	130k	0	56	1	0.12
CAD3	530k	0	60	132	0.54

Table 5.1: Statistics of the results. The number of input tetrahedra, inadmissible singularity edge (Compound, Zigzag-inner and Zigzag-surface), and time spent (in seconds) for fixing them are in columns Tet, Comp, Zz-inner, Zz-surf and T, respectively.

Model	T	Hex	Angle	Length	Scaled_jac	Haus.
Torus	0.5m	12768	90.0/85.7	1.53/1.08	0.981/0.036	0.448
Nut	1m	6540	90.0/88.8	1.28/1.04	0.985/0.0229	0.286
Sphere	1.5m	7776	90.1/83.4	1.50/1.05	0.980/0.0391	0.745
Pretzel	1.17m	3024	90.0/89.4	1.62/1.13	0.921/0.104	0.605
Sculpture	1.3m	10003	89.9/88.4	1.42/1.07	0.984/0.0192	0.295
Fandisk	5m	24001	90.0/82.1	1.70/1.02	0.959/0.0551	0.341
Elk	1.1m	14144	90.0/79.4	1.58/1.01	0.921/0.101	0.574
CAD1	0.5m	21530	89.9/82.4	1.37/1.09	0.977/0.0406	0.709
CAD2	2m	13788	90.0/87.3	1.24/1.01	0.987/0.0264	3.55
CAD3	81.5m	17784	90.0/88.7	1.18/1.01	0.897/0.0257	1.09

Table 5.2: Statistics of the results (average/minimal). The time used in parametrization is listed in the second column. The following columns list mean/minimal of dihedral angles, edge lengths and scaled Jacobian of the parameterization (measurements proposed by Shepherd and Tuttle [97]) , and the Hausdorff distance/ $10^{-3}\times$ bounding box diagonal for measuring the quality of the output hexahedral mesh.

For models shown in Figure 5.7, our method naturally captures their symmetries. Even for high genus models, our method is able to automatically find a proper topological structure.

In the following figures, to demonstrate the parameterization results, we show the adjusted singularity graphs, the resulting hexahedral meshes, and their cutaway views from left to right. In Figure 5.8, without any manual input, the models Pretzel, Elk and Fandisk are remeshed into a polycube-like topology, as indicated by their frame fields. In Figure 5.9, several internal singularities are automatically introduced on the models CAD1, Sculpture and CAD2, producing excellent element quality.

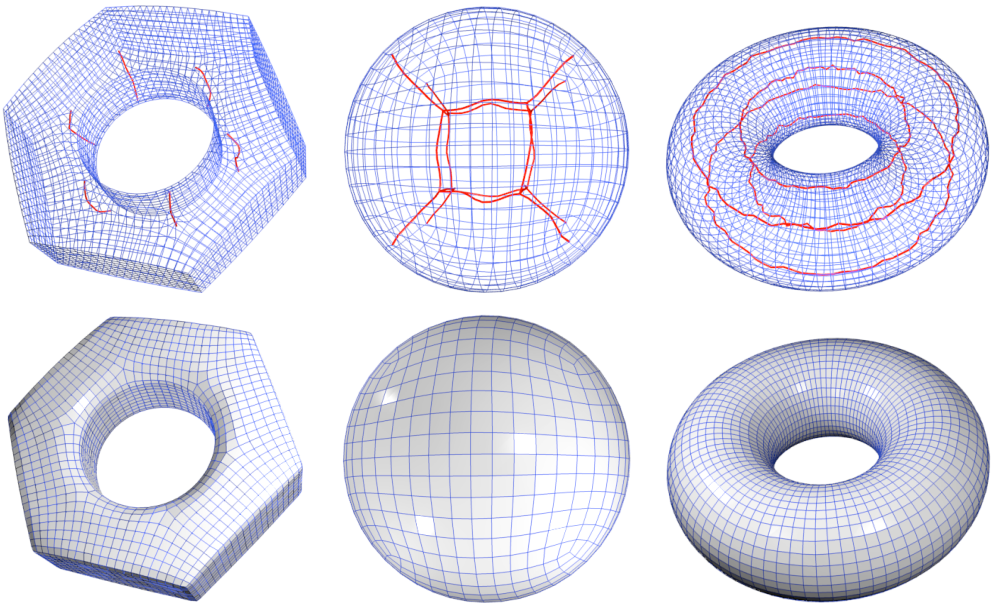


Figure 5.7: Topological structures of the singularities.

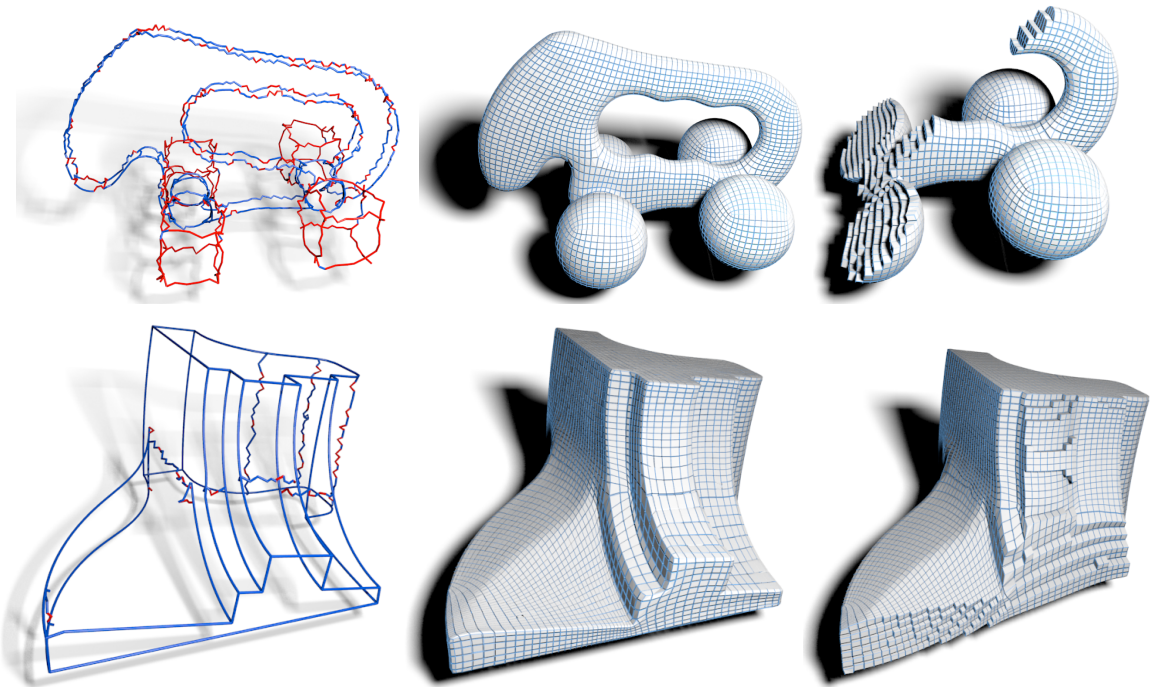


Figure 5.8: Results with polycube-like structure. The red lines are near-misses, exposed on the boundary after parameterization.

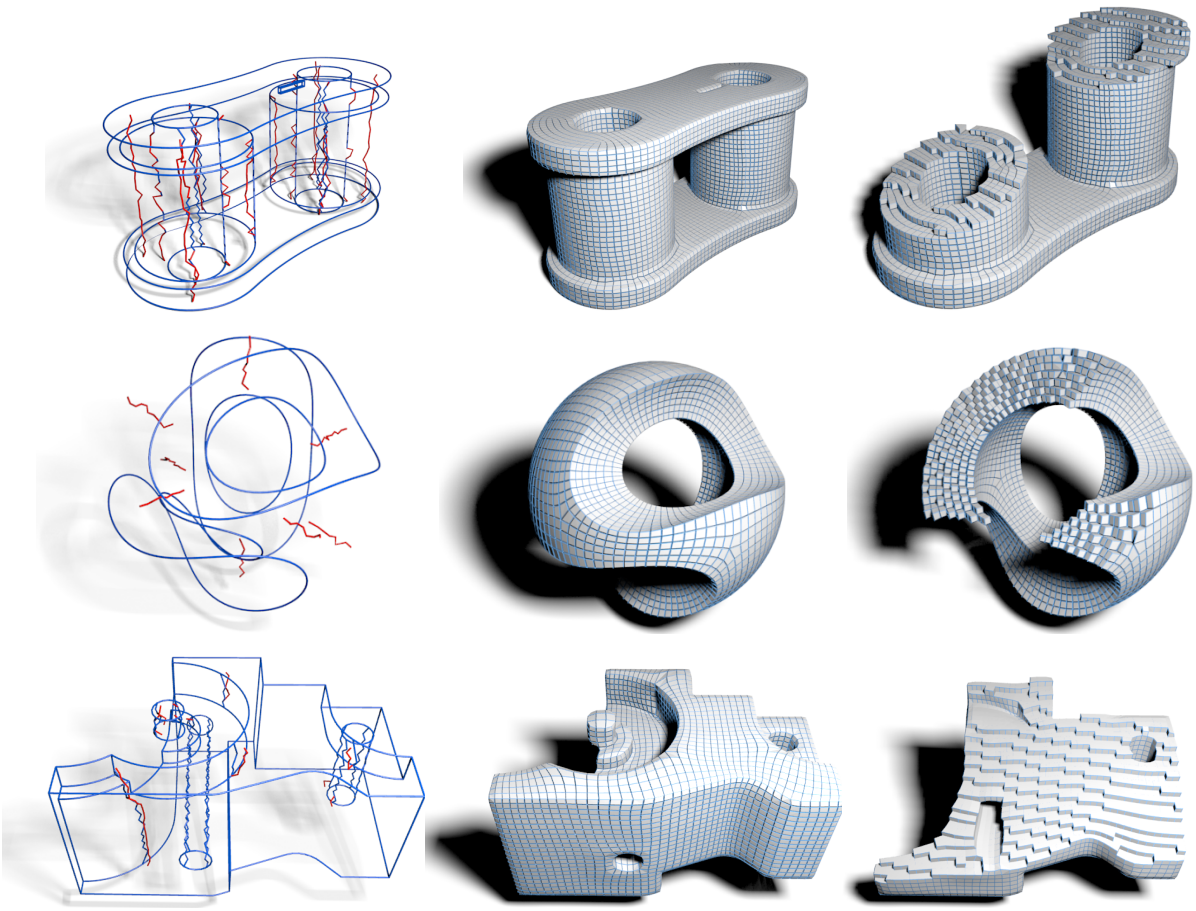


Figure 5.9: Results on CAD1, sculpture and CAD2 contain internal singularities.

In Figure 5.10, we show the result of our method applied on the complex model CAD3. Manually constructing a meta-mesh for such models would have been extremely time-consuming. The models have some small features, which even cause problems in some surface quadrangulation techniques. By choosing a relatively small element size, our method is able to generate an all hexahedral mesh while preserving important features at the same time. The initial singularity graph extracted from the input frame field has roughly depicted the final connectivity structure. However, the original singularities are on or near the surface, such near-misses can lead to distorted and even degenerate parameterization. After automatic adjustment, all singularities are snapped to the surface for the model shown, resulting in a polycube topology.

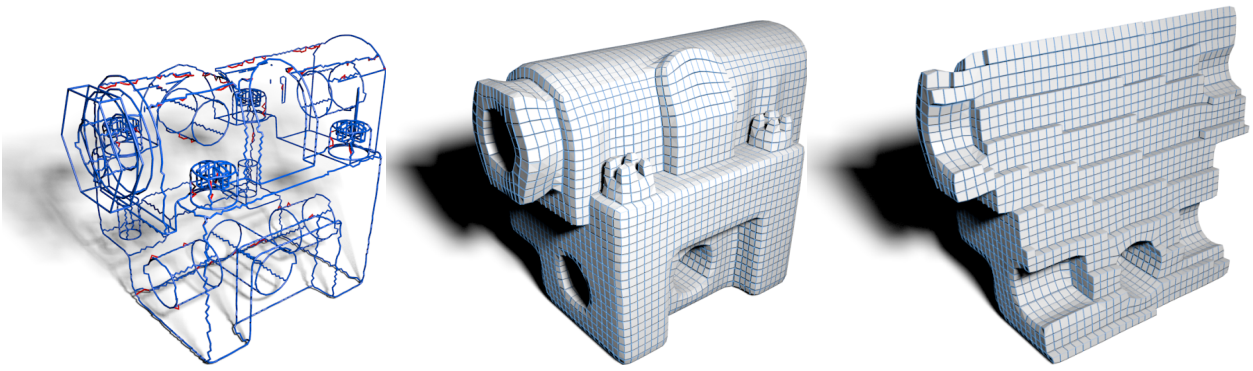


Figure 5.10: Our method can automatically remesh the complex model CAD3 while preserving its important features without time-consuming manual meta-mesh construction.

5.6.1 Comparisons

Automatic generation of high quality hexahedral meshes is still an open problem. Many methods have been proposed under various assumptions. Gregson et al. [42] assumes no internal singularities inside of the object. Nieser et al. [81] requires a valid singularity graph in the input. Lévy and Liu [64] allows non-hexahedral elements in the resulting mesh. The most closely-related work Li et al. [66] also takes a frame field in the input, and tries to reduce the inadmissible singularities. Although our method shares some of the same limitations, such as lack of guarantee in eliminating all inadmissible singularities (e.g. in presence of looped compound singularities), our method has the following advantages:

- In addition to all the inadmissible cases mentioned in Li et al. [66], our method can detect and fix surface inadmissible singularities. As shown in Figure 5.11, our method is more robust by taking such inadmissible singularities into account.

- Our method guarantees convergence for both zigzag and compound edge removal steps. The compound edge collapsing strategy Li et al. [66] may introduce new zigzags, and is thus without convergence guarantee.

- Near miss cases are often inevitable in automatically generated frame fields. We are the first to recognize and address such issues in these frame fields.

- There are known specific local degenerate cases that our method can handle but the method in Li et al. [66] cannot. For instance, a face with three singularity edges in types R_u, R_v, and R_v cannot be properly handled either in "matching adjustment" phase, "improper singular edge collapse" phase, or their ad hoc split method. Our pipeline can trivially handle such cases by first removing zigzag, and then splitting the compound singularity.

- Unlike our subdivision strategy, the collapse strategy used in Li et al. [66] leads to possible numerical issues in parameterization because of the numerous poorly shaped tetrahedral elements introduced in fixing long compound singularities (Figure 5.12).

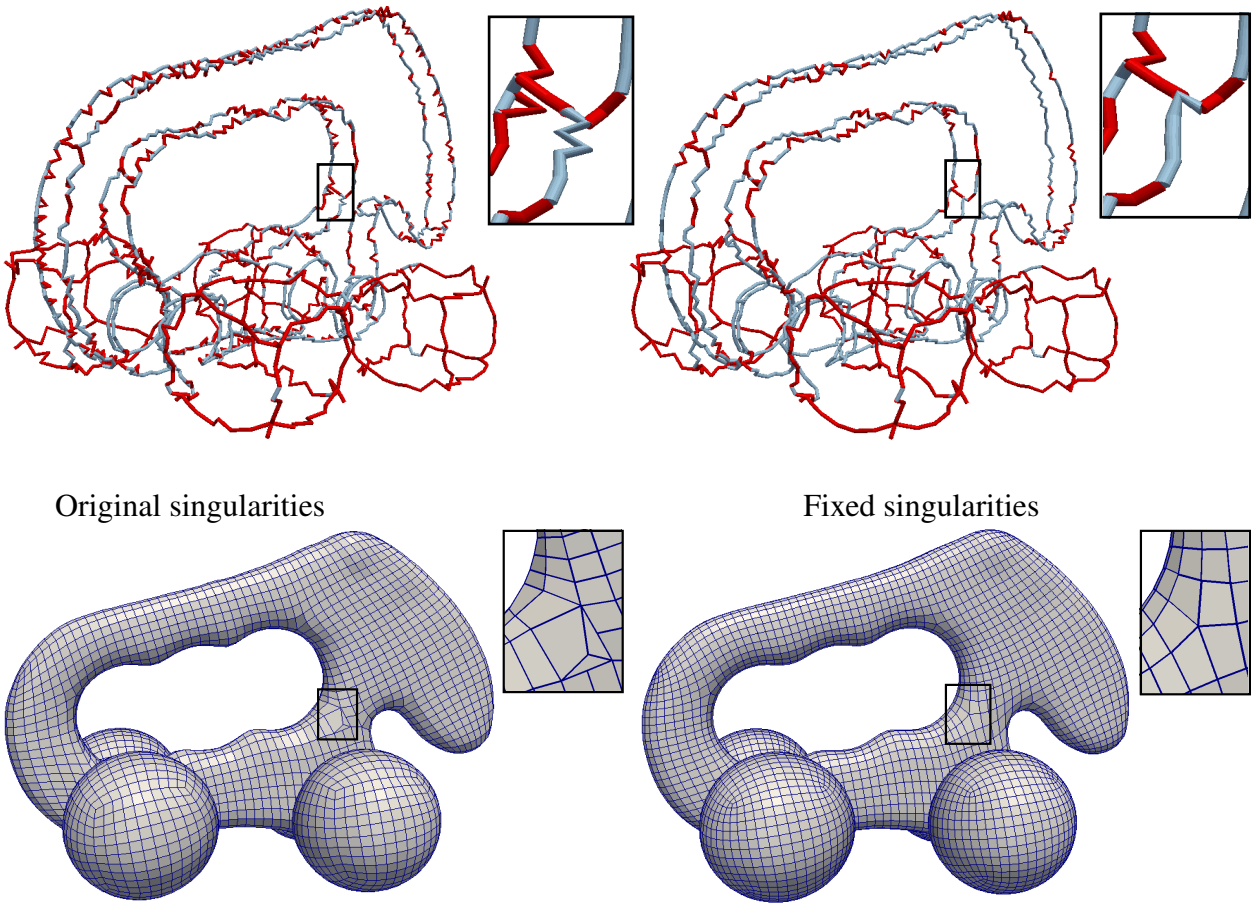


Figure 5.11: Compared with the method in Li et al. [66], which lacks the ability of detecting and fixing inadmissible surface singularities, our method is more robust in generating high quality hexahedral meshes in the presence of such singularities.

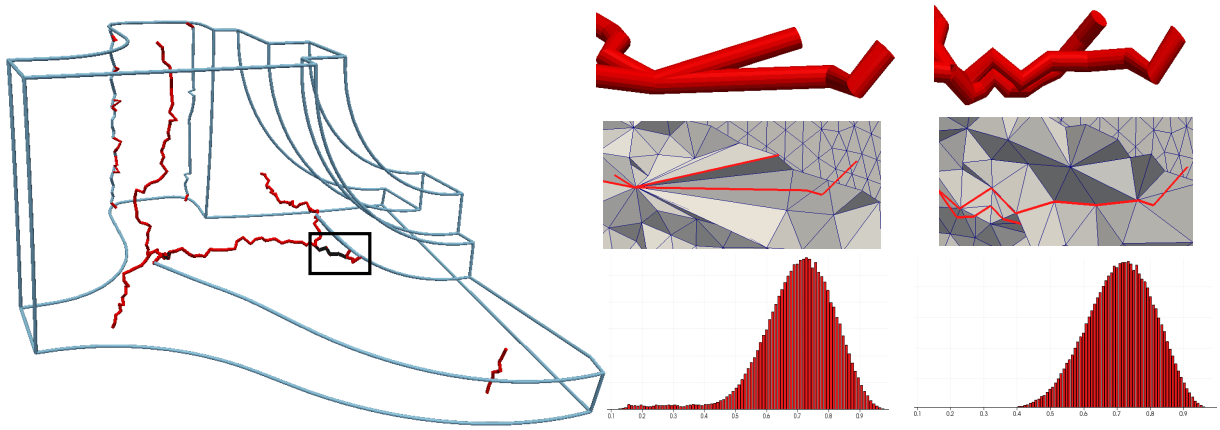


Figure 5.12: Unlike our method, the collapse strategy Li et al. [66] may introduce numerous poorly shaped tetrahedral elements. The histograms show the scaled Jacobian of the tetrahedral meshes.

5.7 Conclusion

In this section, we represent the relations of the singularities in a frame field guided parameterization as a graph, and mathematically formulate the common problematic singularity cases. Then, we provide a framework to automatically detect and treat them in a consistent way within the graph, and finally generate an all-hexahedral mesh by solving a mixed integer programming problem after reducing the number of integer variables. Our main contributions include:

- We give a definition of surface singularities, and necessary conditions for admissible singularity graph of both internal and surface singularities for hexahedral meshing.
- We provide a procedure for treatment of common defects in the singularity graph, which would lead to degenerate parameterization if left alone.
- We demonstrate a practical solution for hexahedral mesh generation guided by automatically generated guidance cross-frame input fields.

Chapter 6

3D Volume Data Structures for Compact Storage

Volumetric meshes, e.g., those generated using our method in the previous chapter, are now ubiquitous in solid modeling, physics-based simulation, computational science, and even rendering of translucent materials. However, the ever-increasing size and complexity of meshes impose undue stress on both memory access times and usage, especially since mesh size typically grows as a cubic function of the resolution. A data structure with small memory footprint that can efficiently handle queries of incidence and adjacency would thus benefit a wide range of applications in graphics and scientific computing in general.

Based on a topological representation called combinatorial maps, we provide a simple but generic method for defining volume cell types, extending the array-based mesh data structure proposed by Alumbaugh and Jiao [3], through *completing the data structure with a list of edges and improving incidence queries within each volume cell*. As shown in Table 1.2, our data structure increases memory usage by only a small fraction compared with Alumbaugh and Jiao [3]'s method.

6.1 Related Work

A common assumption for volumetric meshing is that 3D volumes are of 3D manifolds without degenerate structures such as "shark-fins" or "hanging rods". Under this assumption, a so-called element connectivity table, which maps volume cells to their corner vertices, provides us complete adjacent information about vertices, edges, faces, and volume cells. Although this can be adequate for a certain geometry processing algorithms like those presented in [106, 79], computational applications still prefer to constant-time incidence queries (i.e., access lower dimension geometry primitives from higher dimension ones, or vice versa). For instance, some applications may only require certain incidence queries, e.g., cell-face relations in ray-tracing [77], cell-vertex (cells around vertex) relations in Delaunay tetrahedralization. On the other hand, many FEM-based applications may need all incidence queries, including face-edge relations [28]. Those queries cannot be easily achieved if we only have the element connectivity table. This desired property is referred to as *comprehensiveness* in [3].

To solve this problem, several data structures are proposed. Guibas and Stolfi [43] initially propose a face-edge data structure. Brisson [16] invents a more abstract "cell-tuple" data structure, using the idea of boundary representation for a mesh. This innovative idea is based on the fact that it is theoretically possible to "order" all the $(k-1)$-cells and k-cells around a $(k-2)$-cell on the boundary of a $(k+1)$-cell. In fact, nearly three decades earlier, Edmonds [34] already had introduced a notion called Combinatorial Map for polyhedral surfaces and orientable quasi-manifolds[i]. Lienhardt [67] proposes to extend the combinatorial maps to generalized d-maps to represent non-orientable manifolds. Beall and Shephard [5] develop a topology-based mesh data structure which stores the adjacent information in the boundary representation; however, their direct implemen-

[i] Also called singular manifold in some cases.

tation of adjacent relationships requires a relatively large amount of additional memory storage. Although generalized d-maps can also be compressed [92], the adjacency information will only be restored through decompression.

Recently, a number of compact data structures have been proposed. For example, Blandford et al. [7] introduce a method to provide comprehensive connectivity information using only about 7.5 bytes per tetrahedron. However, this approach is only applicable to simplicial meshes. Alumbaugh and Jiao [3] present a compact array-based data structure for 3D orientable manifold cell complexes. With the concept of anchored half-faces, incident cells can be computed in constant time. However, while an incident edge can be represented by two vertex indices, *it is not possible to find an identifier for the edge, using only the proposed connectivity representation*. Celes et al. [19] independently developed a similar data structure which only explicitly stores the incidence information of nodes and mesh elements, with pre-defined element types. Edges and faces are still implicitly represented, but bit flags are used to ensure uniqueness. Their method also improves the speed of adjacency queries with "reverse indices".

6.2 Combinatorial Maps

In order to introduce the notion of combinatorial maps, we loosely follow the notation used in [24] and call k-dimensional cells *k-cells*. Hence, vertices are 0-cells, edges are 1-cells, faces are 2-cells, and volume cells (such as tetrahedra, prims, etc) are 3-cells. Two cells of different dimensions are said to be incident if one is a subset of the other. Two k-cells of the same dimension are adjacent if they share a common $(k-1)$-cell.

A combinatorial map describes the incidence and adjacency relations among cells of the mesh using a basic element called *dart*, and a group of relations between darts. For an orientable 3D

manifold, a 3D dart corresponds to a cell tuple (v, e, f, c), where v is a starting vertex of an edge e that lies in a face f of 3-cell c. For 2D orientable surfaces, a 2D dart would be the same as the usual half-edge.

An abstract way to define a whole 3D combinatorial map M is to use

1. D is a finite set of darts;
2. for $i = 1, 2, 3, \beta_i : D \to D$ is a mapping;
3. β_1 is a permutation;
4. β_2, β_3, and $\beta_1 \circ \beta_3$ are involutions.[ii]

In this abstract sense, we can define k-cells by *orbits* $\langle S \rangle(d)$, i.e., the set of darts that can be reached by arbitrary combination of maps $m \in S$:

- the 3-cell containing d is $\langle \{\beta_1, \beta_2\} \rangle(d)$;
- the 2-cell containing d is $\langle \{\beta_1, \beta_3\} \rangle(d)$;
- the 1-cell containing d is $\langle \{\beta_2, \beta_3\} \rangle(d)$,
- the 0-cell containing d is $\langle \{\beta_1 \circ \beta_2, \beta_1 \circ \beta_3\} \rangle(d)$.

The following pictures show an example of how combinatorial maps defined on a volume. Readers are invited to verify the above definitions.

[ii] For a map $f : X \to X$, if $\forall x : f \circ f(x) = x$, we call this map an involution. Hence, condition 4 means that $\forall d \in D, \beta_2 \circ \beta_2(d) = d, \beta_3 \circ \beta_3(d) = d$, and $(\beta_1 \circ \beta_3) \circ (\beta_1 \circ \beta_3)(d) = d$.

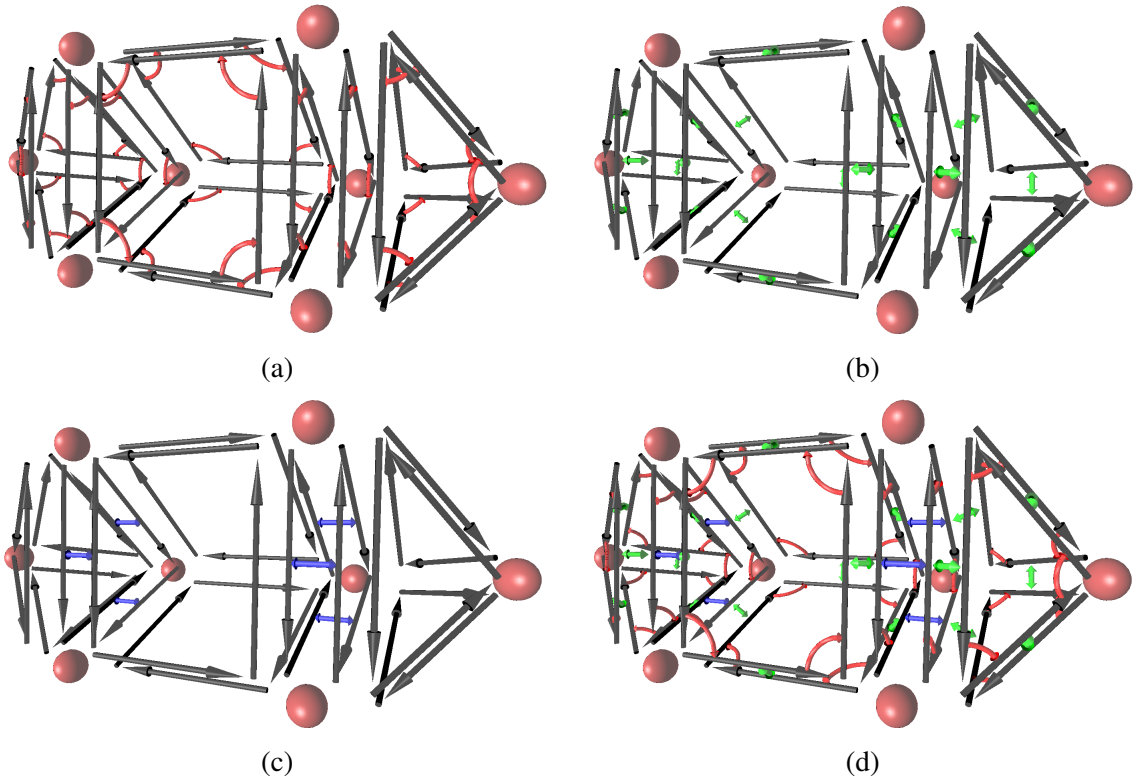


Figure 6.1: Example: Combinatorial map for a volume: (a) β_1; (b) β_2; (c) β_c; (d) the complete combinatorial maps.

Intuitively speaking, β_i maps a dart to another dart with a different i-cell and a different vertex. If we identify the darts with (v, e, f, c) in the regular cell complex description, $\beta_1((v, e, f, c)) = (v', e', f, c)$, $\beta_2((v, e, f, c)) = (v', e, f', c)$, and $\beta_3((v, e, f, c)) = (v', e, f, c')$. Note that β_1 and β_2 are the 3D analogues of a half-edge's next() and opposite() operations, respectively.

6.3 Compact Volume Data Structure Using Compact Maps

In this section, we propose our compact data structure based on the above combinatorial maps to achieve two goals: 1) a data structure compatible with arbitrary valid cell elements; 2) efficiency of all commonly-used incidence queries.

6.3.1 Overview

6.3.1.1 File format

For a 2D polygonal mesh, the complete connectivity information can be encoded by a face list, with each entry corresponding to the list of vertices in the polygon face. However, for a polyhedral mesh, the same list of vertices can correspond to different polyhedra. For instance, an octahedron and a prism both have six vertices. As there are only a handful of k-cell types in most k-dimensional meshes used in practice, we opt to describe all the k-cell types in the header part of the file, and to describe each polyhedron by an ordered vertex list and its k-cell type.

6.3.1.2 Comprehensive data structure

All low dimensional ($\leq k-1$) relations ($\beta_1, \ldots, \beta_{k-1}$) map darts within the same k-cell. Given the type of a k-cell, we may assign each dart in that cell a local id, and the maps among the darts can be precomputed when the k-cell type. One can easily assemble a global ID for each dart by (C, d), where C is the global ID of the k-cell, and d is the local dart ID. Additional auxiliary local incidence mapping to increase efficiency can also be created for each k-cell type at a constant memory cost (independent of the mesh size).

Combinatorial map β_k maps a dart in one k-cell C_1 to another dart in an adjacent k-cell C_2. Noticing the relation among β's, we only store β_k for one dart in the common $(k-1)$-cell in C_1. Thus, the size of β_k can be reduced to one dart per pair of k-cell and $(k-1)$-cell.

The relation between k-cells and darts is implicitly given in the way we express a global ID for each dart (C, d). The mapping from darts to vertices (0-cells) is stored in the vertex lists for k-cells, also called the element connectivity in array-based methods such as Alumbaugh and Jiao [3], denoted $Cv2V$ below. The map from each vertex to one of its darts is stored in a table, denoted

by $V2D$ below .

The above information enables constant time incidence/adjacency inquiries among vertices, k-cells, and "half"-$(k-1)$-cells, akin to Alumbaugh and Jiao [3] except some subtle differences. However, no unique IDs are actually given to 1-cells, 2-cells, through (k–1)-cells, hence no constant time incidence inquiries involving these cells can be achieved, without additional memory cost.

We propose to build a minimal set of additional connectivity tables to provide these incidence relations crucial to real world applications. We describe them as optional, since often only some of the tables in this set are needed, although at least one of them is, in many cases, indispensable. Here we restrict our discussion to 3D. To create a unique edge identifier we use a table called $E2D$, which maps a global edge ID to one of its darts. The map from darts back to edges can be implemented through a table $V2E$ mapping a vertex to the edge starting from it with the smallest ID, as elaborated below. Similarly, but less frequently required, we assign unique face IDs through the table $F2D$, and the backward mapping by $V2F$.

6.3.2 Details for 3D

To illustrate the detailed actual data structure, we use as a running example the description of the simple meshes shown in Figure 6.2, as found in a mesh file—skipping the list of vertex coordinates since our focus is on connectivity information. As in the compact array-based half-face data structure (HFDS) [3], we leverage the fact that there are only a few types of cells typically used in engineering or graphics applications. However, unlike in HFDS, we will not limit ourselves to 3-cells used in the CFD General Notation System (tetrahedron, pyramid, prism, and hexahedron): any 3-cell type for which faces are locally defined can be specified in the header of a mesh file.

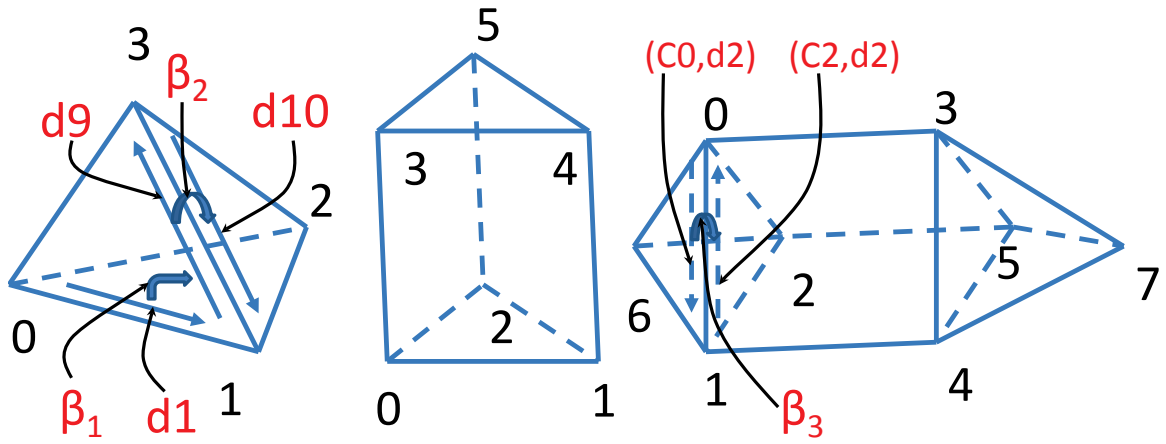


Figure 6.2: Running example.

6.3.2.1 Local information within each 3-cell

Each 3-cell is treated locally as a 2-manifold cell complex, which can be represented by a local half-edge structure, i.e., a 2D combinatorial map. For a given type of 3-cell with n_v vertices, n_e edges, n_f faces:

- locally denote each vertex by v_i, with $i \in \{0, \ldots, n_v - 1\}$;
- locally label each face as $f_m = (v_i, v_j, v_k, \ldots)$, with $m \in \{0, \ldots, n_f - 1\}$.
- (optionally) locally label each of the $2n_e$ darts as $e_k = (v_i, v_j)$, with $k \in \{1, \ldots, 2n_e\}$;

Darts are indexed starting from 1, as 0 is reserved for boundaries.

The mesh file for Figure 6.2 would thus contain the following information:

faces	0:(0,2,1)	1:(0,1,3)	2:(1,2,3)	3:(2,0,3)
darts	1:(0,1)	2:(1,0)	3:(0,2)	4:(2,0)
	5:(0,3)	6:(3,0)	7:(1,2)	8:(2,1)
	9:(1,3)	10:(3,1)	11:(2,3)	12:(3,2)

Table 6.1: Cell type 0 (tetrahedron).

faces	0:(0,2,1)	1:(0,1,4,3)	2:(1,2,5,4)
	3:(2,0,3,5)	4:(3,4,5)	
darts	1:(0,1)	2:(1,0)	3:(0,2)
	4:(2,0)	5:(0,3)	6:(3,0)
	7:(1,2)	8:(2,1)	9:(1,4)
	10:(4,1)	11:(2,5)	12:(5,2)
	13:(3,4)	14:(4,3)	15:(3,5)
	16:(5,3)	17:(4,5)	18:(5,4)

Table 6.2: Cell type 1 (prism).

type 0	C0:(1,0,2,6)	C1:(3,4,5,7)
type 1	C2:(0,1,2,3,4,5)	

Table 6.3: Cells table for mesh shown in Figure 6.2 .

In all the tables we list, the information before ":" is for illustration purposes only, and is thus not stored in memory or files. For each 3-cell type, defining only the faces would be necessary and sufficient, since we can build the darts based on faces and give them labels. We then build a lookup table for β_1 and β_2 of all darts, with $2n_e$ entries and $2n_e$ possible values in the range for

each entry. In our running example, the β_1 and β_2 tables for 3-cell type 0 are

d	1	2	3	4	5	6
β_1	9	3	8	5	12	1
β_2	2	1	4	3	6	5
d	7	8	9	10	11	12
β_1	11	2	6	7	10	4
β_2	8	7	10	9	12	11

Table 6.4: β_1 and β_2 tables for the running example shown in Figure 6.2.

Here the rows labeled β_1 and β_2 contain the images of the darts of the same column in the rows labeled with d, e.g. $\beta_1(1) = 9$ and $\beta_2(1) = 2$.

Assuming a small number of 3-cell types compared to mesh size, these type specifications only use a negligible amount of memory. In fact, storing all the *local* incidence and adjacency information directly for improved speed only requires an additional constant memory cost. We denote local incidence mappings as follows:

- $d2f(d)$ maps a dart d to its local face ID;
- $f2d(f, i)$ is the i-th dart of the local face f;
- $d2v(d)$ maps a dart d to its starting vertex.

We use lower (resp., upper) cases in the name of a map to denote whether the index is local (resp., global).

6.3.2.2 Global information

We load the connectivity table that contains, for each 3-cell, the global indices of its vertices. We denote this table by $Cv2V(C, v)$ since it maps the v-th vertex of 3-cell C to its global index V. Note that this corresponds to the usual way of storing the bare minimum connectivity information of 2D polygonal meshes in files. Similarly, one can organize the file by listing vertex lists of every 3-cell in their order of enumeration; that is, the file first lists the descriptions of 3-cell types, followed by vertex lists of all 3-cells of the first type, the second type, etc. Once we have the 3-cell connectivity, a dart can be globally indexed by an ordered pair $D = (C, d)$, where C is the global 3-cell ID, and d is the local dart index. Note that instead of using a local face index with a starting vertex (called anchored half face) as in HFDS, we use local indices of darts; for the common case of tetrahedron meshes, this means we can cope with meshes twice as large for the same amount of memory.

To complete incidence and adjacency information in the combinatorial map, we need to construct β_3. We save space by noticing that $\beta_3 = \beta_1 \circ \beta_3 \circ \beta_1$, which means that $\beta_3(D)$ can be inferred if $\beta_3(\beta_1(D))$ is known. Thus, we only store β_3 for the *first* dart in each half face $H = (C, f)$, and denote this additional table by $H2D(C, f)$. If the application requires the use of boundary darts, their β_3 can be stored in a separate list $B2D(B)$, mapping the first dart of each boundary face B to its corresponding dart in the 3-cell adjacent to it. We also need to map from a vertex to one of its darts $V2D(v)$; but the map from a dart to its starting vertex is trivially found by $D2V(C, d) = Cv2V(C, d2v(d))$.

β_3	C0	f0d3:(2,8)	f1d1:(0,0)	f2d7:(1,0)	f3d4:(2,0)
	C1	f0d3:(2,16)	f1d1:(3,0)	f2d7:(4,0)	f3d4:(5,0)
	C2	f0d3:(0,8)	f1d1:(6,0)	f2d7:(7,0)	f3d4:(8,0)
		f4d13:(1,2)			
B2D	BF0:(0,1)	BF1:(0,7)	BF2:(0,4)	BF3:(1,1)	BF4:(1,7)
	BF5:(1,4)	BF6:(2,1)	BF7:(2,7)	BF8:(2,4)	
V2D	V0:(0,7)	V1:(0,1)	V2:(0,4)	V3:(1,1)	V4:(1,7)
	V5:(1,4)	V6:(0,10)	V7:(1,6)		

Table 6.5: The tables for the 3-cell example.

6.3.2.3 Boundary

The map β_3 usually returns an internal dart (C, d) with $d > 0$. However, if the opposite is a boundary dart, it will return $(B, 0)$, where B is the boundary half-face ID. We carefully choose $V2D$ so that whether a vertex V is on boundary can be determined by examining $\beta_3(V2D(V))$. Darts belonging to boundary half-face do not need to explicitly maintained in most cases .

6.3.2.4 Edge and face incidence information

If we need to use a unique edge identifier, a table for $E2D(E)$ is maintained to map an edge to one of its darts. We sort the edges in the $E2D$ table by lexicographic order of their vertices (V_{start}, V_{end}) assuming that it always points from the vertex with a smaller index to the one with a larger index. A backward mapping $D2E$ can be implemented by a table $V2E(V)$, mapping vertex V to the first edge starting from it. We can avoid sorting the edges by using a linked list at the cost of storing another n_1 integers. The map $V2E$ would then be made to map a vertex to a linked list of edges

starting from it.

If only half faces need identifiers, (C, f) can be used instead. Otherwise, a table $F2D(F)$ is required. Similar to the edge case, we can sort the faces by their first three vertices, assuming that the vertices are in ascending order within each face F. Then the backward mapping $D2F$ can be implemented by $V2F(V)$, mapping vertex V to the first face that has V as its smallest-indexed vertex.

For our running example, the (optional) edge tables are:

E2D	E0(V0,V1):(0,2)	E1(V0,V2):(2,3)	E2(V0,V3):(2,5)
	E3(V0,V6):(0,9)	E4(V1,V2):(0,3)	E5(V1,V4):(2,9)
	E6(V1,V6):(0,5)	E7(V2,V5):(2,11)	E8(V2,V6):(0,11)
	E9(V3,V4):(2,13)	E10(V3,V5):(1,3)	E11(V3,V7):(1,5)
	E12(V4,V5):(2,17)	E13(V4,V7):(1,9)	E14(V5,V7):(1,11)
V2E	V0:0	V1:4	V2:7 V3:9
	V4:12	V5:14	V6: V7:

Table 6.6: Optional edge tables for the running example.

6.3.2.5 Example table construction

The construction of most tables is straightforward since the mesh connectivity information is complete. We only give an example of how to build $E2D$ in Algorithm 1. Note that the procedure ensures that a quick counter-clockwise traversal of the edge's one-ring is possible even when it is on the boundary, and an easy boundary test through $\beta_3 \circ \beta_2(E)$.

Algorithm 3: Build $E2D$ table

```
1:  Initialize flag table visited(), E ← 0
2:  for all non-boundary dart D do
3:    if visited(D) then
4:      continue
5:    end if
6:    D_0 ← D
7:    while true do
8:      D' ← β_3 ∘ β_2(D) {rotate clockwise}
9:      if Boundary(D') or D' = D_0 then
10:       break
11:     end if
12:     D ← D'
13:   end while
14:   E2D(E) ← D, E ← E + 1, D_0 ← D
15:   repeat
16:     visited(D) ← true, vistied(β_2(D)) ← true
17:     D ← β_2 ∘ β_3(D) {rotate counter-clockwise}
18:   until Boundary(D) or D = D_0
19: end for
```

6.3.2.6 Spatial complexity

Tetrahedron meshes are the easiest test objects to establish comparisons between various data structures: for such meshes, we can approximate all k-cell counts n_k as a function of the number of tetrahedra n_3 and boundary faces n_b—other mesh types must be analyzed using the count of darts, and its estimated relation with k-cell numbers. Following Beall and Shephard [5], we assume the average valence of a vertex is around 4π divided by the solid angle for a vertex of an equilateral tet 0.5513, i.e., 22.8. Additionally, we assume that the average solid angles at boundary nodes are about half of the average angle. Based on these assumptions, the fact that each tetrahedron has 4 vertices and 4 faces, and Euler's formula, we have

$$22.8n_0 \approx 4n_3,\ 4n_3 + n_b = 2n_2,\ n_0 - n_1 + n_2 - n_3 \approx 0.$$

The k-cell counts are therefore

$$n_0 \approx 0.175n_3,\ n_1 \approx 1.175n_3 + 0.5n_b,\ n_2 = 2n_3 + 0.5n_b.$$

In the following analysis, we assume that the lowest four or more bits are sufficient to encode the local dart index or the local half face index; thus we need only one integer for (C, d) or (C, f). Alternatively, for tetrahedron meshes with fixed connectivity, we can use an integer D such that it represents $C = D/12$ and $d = D\%12$. When 3-cells are sorted by type, this method can be easily extended to cope with hybrid meshes and to include boundary darts. The memory size required for the various connectivity tables are listed below:

Table	V2XYZ	Cv2V	H2D	V2D	B2D
Space	$3n_0$	$4n_3$	$4n_3$	n_0	n_b
Table(optional)		E2D	V2E	F2D	V2F
Space		n_1	n_0	n_2	n_0

Table 6.7: Memory cost for the various connectivity tables.

By tallying up these numbers, we find that $8n_3 + n_0 + n_b \approx 8.175\, n_3 + n_b$ integers are required for the basic tables, in par with the basic eight pointers per tetrahedron (pointing to adjacent tetrahedra and corner vertices) plus one pointer per node (to one incident tetrahedron) used to encode connectivity in Pyramid and CGAL, and close to Blandford et al. [7]'s tetrahedron mesh structure prior to difference code compression. Data structures capable of handling generic polytope meshes require more memory space when used for simplicial meshes, e.g., Dobkin and Laszlo's structure [30] would require around $18n_3$ pointers, while radial-edge, cell-tuple, and G-map representations, as well as CGAL's combinatorial map, would use even more memory. If unique edge

identifiers are needed, we require $n_0 + n_1 \approx 1.35\, n_3$ additional integers, which is more compact than the pure tet mesh encoding of [7] before difference coding.

HFDS [3] uses the same amount of basic space ($8.175\, n_3 + n_b$). However, their encoding of a local dart (anchored face) identifier (C, f, v) uses a separate local index f for a face within the tetrahedron and a local index v of a vertex within the face. Thus, it would be less memory efficient when dealing with generic 3-cells, for example, 3-cells that have 5-edge faces or more. In addition, even in the common case of tetrahedron meshes, HFDS requires 5 bits for local indices ($f = 0$ is reserved for boundary), while we only need 4 bits, enabling us to handle meshes with 256M 3-cells with a 32-bit integer representation, instead of their 128M limit.

Furthermore, and *key to runtime efficiency*, we provide a simple way to give edges and faces unique identifiers. As we elaborate upon next, this enables constant time incidence queries, and allows appending attributes to edges and faces, which are important in simulation and other computational tasks. The HFDS data structure does not actually provide any means to get unique adjacent edge IDs in constant time.

6.4 Incidence/Adjacency Queries

As our data structure can be seen as an internal representation of a combinatorial map, it can directly leverage any implementation of combinatorial maps to get incidence and adjacency information in constant time. In addition, with integer IDs, additional attributes associated to vertices, darts, half faces, cells, edges, and faces, can be directly allocated as an array with the appropriate size, making it highly efficient and flexible for static meshes. We will first give a few examples of commonly-used neighborhood constructions such as one-rings in Algorithms 4 and 5.

Assuming constant maximum valence, both algorithms run in constant time. To map a dart to

Algorithm 4: One-ring darts and cells around vertex V_0

Ensure: {darts} and {cells} contain darts and cells in the one-ring.

$C \leftarrow (V2D(V_0) \rightarrow C)$
Queue $Q.push(C)$, save C in {cells}
while Q not empty **do**
  $C \leftarrow Q.pop()$
  $\{D_i\} \leftarrow$ all darts starting at V_0 in C
  save $\{D_i\}$ in {darts}
  for all D in $\{D_i\}$ **do**
    $C \leftarrow (\beta_3(D) \rightarrow C)$
    if C not in {cells} **then**
      $Q.push(C)$, save C in {cells}
    end if
  end for
end while

Algorithm 5: One-ring (internal) HF around Edge E_0

Ensure: Array {HF} is the CCW ordered one-ring.

$D_0 \leftarrow E2D(E_0)$, $D \leftarrow D_0$
repeat
  $C \leftarrow (D \rightarrow C)$, $d \leftarrow (D \rightarrow d)$
  save $(C, d2f(d))$ and $(C, d2f(\beta_2(d)))$ in {HF}
  $D \leftarrow \beta_2 \circ \beta_3(D)$ {rotate counter-clockwise}
until $Boundary(D)$ or $D = D_0$

a unique edge ID, we find the end vertices (V_{start}, V_{end}) with $V_{start} < V_{end}$. We then perform a linear search in $E2D$ starting from $V2E(V_{start})$, this again would terminate in constant time.

In most cases, faces do not need a unique ID, as attributes are often associated to half faces; but if needed, our $F2D$ and $V2F$ tables can be used to provide a unique face ID.

All other incidence information can be similarly assembled from the mappings between cells and darts and the mappings among darts.

6.5 Conclusion

In this section, we present a concise local connectivity representation of generic 3-cell (volume cell) types, an efficient way to store all the combinatorial maps in volume meshes, and a straightforward way of associating attributes to k-cells ($k \in \{0, 1, 2, 3\}$). With this compact volumetric data structure, we demonstrate that the time complexity of accessing to adjacency information (e.g., *face-edge*) is constant.

One limitation of the compact combinatorial map data structure described here is its apparent inability to deal gracefully with dynamically changing connectivity, in particular with possible changes of 3-cell types. (On the other hand, if 3D cells are kept intact as in the case of cutting or merging meshes along faces, the mesh can be easily modified accordingly.) However, we believe that our data structure can be readily altered to efficiently handle connectivity changes as well: one could use pointers instead of integers for the IDs of 3-cells and vertices—and the last few bits of the pointer can actually be used to encode local dart index as in the integer case.

Chapter 7

Conclusion and Future Work

Based on the theory of discrete differential geometry, we present novel representations of mesh connectivity, geometric shape, singularity structure, and shape correspondence, and demonstrate the effectiveness of our representations in applications such as meshing and shape correspondence. We summarize our main contributions as follows:

- In Chapter 3, based on the first and second fundamental forms represented simply by edge lengths and dihedral angles, we present local and global versions of the discrete fundamental theorem of surfaces. The preservation of the fundamental theorem of surfaces leads to a simple procedure to construct an embedding through a sparse linear solver given an arbitrary set of edge lengths and dihedral angles. As the main application, a practical mesh deformation algorithm is proposed to handle point constraints and orientation constraints concurrently in a rigid-motion-invariant fashion.

- In Chapter 4, we extend the functional map representation for correspondence maps to the vector field map representation. The generalized vector field map representation is able to handle the constraints for correspondence inference in the important conformal case. We

also demonstrate the possibility of extending the method to handle function transfer between shapes with different topology. For the specific bases used in the generalization, we develop a spectral vector field analysis tool for handling various vector fields on surfaces.

- In Chapter 5, we present a global volumetric parameterization-based tool to automatically generate all-hexahedral meshes based on 3D frame fields. We first introduce the definition of surface singularities in the singularity structure represented as a graph. Although a manually constructed or adjusted frame field may lead to a singularity graph compatible to a non-degenerate parameterization, an automatically generated one usually will not. To eliminate degeneracy in parameterization caused by conflicting rotational transitions often presented in an automatically generated frame field, the definitions and some analyses on commonly seen inadmissible internal and surface singularity types are provided in this dissertation. We also devise a framework to adjust these problematic singularities by applying a sequence of local operations with guaranteed convergence, by rerouting and splitting the singularity edges within the graph.

- In Chapter 6, an efficient internal representation of combinatorial maps is presented to describe the connectivity of volumetric meshes. All necessary components in combinatorial maps can be implemented in compact form. Compared to previous work, our data structure can handle arbitrary 3-cell types, and it provides adjacency and boundary inquiries in constant time. Appending attributes to *cells of any dimension* is also straightforward.

Our novel representations based on the theory of discrete differential geometry points to several directions for future research:

- In our local surface shape representation, we proposed an ad-hoc method for conversions between matrix functions and discrete fundamental forms. For further improvement on ac-

curacy, it is necessary to investigate the continuous limits of the matrix functions when we refine the mesh. Another topic worth exploring is the design of a modified version of our energy functional to serve as a potential energy for thin shell simulations—we could associate translational velocities at vertices with a lumped mass matrix, and rotational velocities at faces with moments of inertia of each triangle. We also wish to study proper handling of potential energy with inhomogeneous and anisotropic material property using our basic representation. Based purely on geometric information, we can also try to weight different dihedral angles differently to enforce feature preservation, or weight regions differently to modify their rigidity according to user specifications.

- In theory, our vector field map representation assumes conformal mapping, which implies that the gradient field mapping ($\bar{C}$) and curl field mapping ($\tilde{C}$) are identical. Thus, to extend the representation to non-conformal cases, one must explore the representation with $\bar{C} \neq \tilde{C}$. In practice, experimenting with possible applications in automatic feature alignment through our novel constraint for vector field pair preservation may lead to improved correspondence inference. Applying our vector field map representation to areas such as analysis of large collections of non-isometric similar shapes [80, 58, 94] or improvement to point-to-point maps [57] are also interesting directions for future endeavor.

- One major limitation of our automatic hexahedral meshing method is the lack of a theoretical proof to guarantee that the set of all the conflicting geometric and topological structures can be detected and fixed by using the definition of inadmissible singularity. It is interesting yet challenging to develop a sufficient condition to guarantee a complete solution for automatic all-hexahedral remeshing. Current methods may generate hexahedral meshes with degenerate elements that cannot be removed without introducing new degeneracy. Indeed, we find

that even a singularity graph containing no zigzag or compound singularities can still lead to degeneracy. We conjecture that it may be related to near misses or certain global inadmissible structure of the singularity graph, which may force certain singularity lines to be mapped to a single point in the parameter domain. Thus, we still study the global admissible condition for the singularity graph.

- For compact volumetric mesh data structure, a possible research direction worth exploring is the design of admissible local connectivity changes (such as edge removal or 2-3 flip) that maintain the validity of our combinatorial map data structure. Compression of neighboring information (β_3) using difference coding after sorting the cells along space-filling curves could also lead to further reduction of memory usage. Additionally, the extension to dimension $n > 3$ could be done by encoding the local connectivity ($\beta_1, \ldots, \beta_{n-1}$) of n-cell types, and storing only β_n. Another possible future direction to explore would be the application of the data structure in tasks involving volumetric data, such as 3D tensor field design and solid texturing [120, 117].

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Alexa. Recent advances in mesh morphing. *Computer graphics forum*, 21(2):173–198, 2002.

[2] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3):485–493, jul 2003.

[3] T. J. Alumbaugh and X. Jiao. Compact Array-Based Mesh Data Structures. In B. W. Hanks, editor, *Engineering*, IMR '05, pages 485–503. Springer Berlin Heidelberg, 2005.

[4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, jul 2005.

[5] M. W. Beall and M. S. Shephard. A general topology-based mesh data structure. *International Journal for Numerical Methods in Engineering*, 40(9):1573–1596, 1997.

[6] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, feb 1992.

[7] D. K. Blandford, G. E. Blelloch, D. E. Cardoze, and C. Kadow. Compact Representations of Simplicial Meshes in Two and Three Dimensions. *International Journal of Computational Geometry and Applications*, 15(1):3–24, 2005.

[8] A. Bobenko, P. Schröder, J. Sullivan, and G. Ziegle. *Discrete Differential Geometry*. Springer, 2008.

[9] A. I. Bobenko and P. Schröder. Discrete Willmore flow. *ACM SIGGRAPH 2005 Courses on - SIGGRAPH '05*, page 5, 2005.

[10] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, oct 1984.

[11] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph. (SIGGRAPH)*, 28, 3:77:1–77:10, July 2009.

[12] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-Integer Quadrangulation. *ACM Trans. Graph.*, page to appear, 2009.

[13] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Symposium on Geometry Processing 2004*, SGP '04, pages 185–192, New York, NY, USA, 2004. ACM.

[14] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213 –230, 2008.

[15] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. uno Levy. *Polygon Mesh Processing*. AK Peters, 2010.

[16] E. Brisson. Representing geometric structures in d dimensions: topology and order. In *SCG 89 Proceedings of the fifth annual symposium on Computational geometry*, volume 9, pages 218–227. ACM Press, 1989.

[17] A. Bronstein, M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[18] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Science*, pages 1168–1172, 2006.

[19] W. Celes, G. H. Paulino, and R. Espinha. A compact adjacency-based topological data structure for finite element mesh representation. *International Journal for Numerical Methods in Engineering*, 64(11):1529–1556, nov 2005.

[20] I. Chao, U. Pinkall, P. Sanan, and P. Schröder. A simple geometric model for elastic deformations. *ACM Trans. Graph. (SIGGRAPH)*, 29:38:1–38:6, July 2010.

[21] K. Crane, M. Desbrun, and P. Schröder. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum (SGP)*, 29(5):1525–1533, 2010.

[22] K. Crane, U. Pinkall, and P. Schröder. Spin transformations of discrete surfaces. In *ACM Transactions on Graphics (SIGGRAPH)*, volume 30, page 104. ACM, 2011.

[23] K. Crane, U. Pinkall, and P. Schröder. Robust fairing via conformal curvature flow. *ACM Trans. Graph.*, 2013.

[24] G. Damiand. Combinatorial maps. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.0 edition, 2012. www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html.

[25] J. Daniels, C. T. Silva, J. Shepherd, and E. Cohen. Quadrilateral mesh simplification. *ACM Trans. Graph.*, 27(5):148:1–148:9, dec 2008.

[26] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Graphics Interface*, volume 11, pages 145–152. Citeseer, 2000.

[27] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum*, 21, 2002.

[28] M. Desbrun, E. Kanso, and Y. Tong. Discrete Differential Forms for Computational Modeling. In G. Bobenko, A. Schröder, P. Sullivan, J. Ziegler, editor, *Discrete Differential Geometry*, pages 287–324. Springer, 2008.

[29] M. Desbrun, E. Kanso, and Y. Tong. Discrete differential forms for computational modeling. *Discrete differential geometry*, pages 287–324, 2008.

[30] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of the third annual Symposium on Computational Geometry*, SCG '87, pages 86–99, New York, NY, USA, 1987. ACM.

[31] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des.*, 22(5):392–423, 2005.

[32] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph. (SIGGRAPH)*, 25(3):1057–1066, 2006.

[33] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry (SoCG)*, 30(1):87–107, July 2001.

[34] J. R. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices Amer Math Soc*, 7:646, 1960.

[35] X. Feng, Y. Wang, Y. Weng, and Y. Tong. Compact combinatorial maps in 3d. In S.-M. Hu and R. Martin, editors, *Computational Visual Media*, volume 7633 of *Lecture Notes in Computer Science*, pages 194–201. Springer Berlin / Heidelberg, 2012.

[36] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe. Design of tangent vector fields. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):56, 2007.

[37] M. S. Floater and K. Hormann. Surface Parameterization: a Tutorial and Survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer Berlin Heidelberg, 2005.

[38] T. Frankel. *The Geometry of Physics: An Introduction, Second Edition*. Cambridge University Press, 2 edition, 2003.

[39] S. Fröhlich and M. Botsch. Example-Driven Deformations Based on Discrete Shells. *Computer Graphics Forum*, 30(8):2246–2257, dec 2011.

[40] R. V. Garimella and B. K. Swartz. Curvature Estimation for Unstructured Triangulations of Surfaces. Technical Report LA-UR-03-8240, Los Alamos National Laboratory, 2003.

[41] A. Gray, E. Abbena, and S. Salamon. *Modern Differential Geometry of Curves and Surfaces with Mathematica, Third Edition*. Studies in Advanced Mathematics. Taylor & Francis, 2006.

[42] J. Gregson, A. Sheffer, and E. Zhang. All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum (SGP)*, 30:5:1407–1416, 2011.

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

[44] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom. Theory Appl.*, 14(1-3):49–65, nov 1999.

[45] A. N. Hirani. *Discrete exterior calculus*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, 2003. AAI3086864.

[46] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.*, 27:147:1–147:9, December 2008.

[47] J. Huang, Y. Tong, K. Zhou, H. Bao, and M. Desbrun. Interactive Shape Interpolation through Controllable Dynamic Deformation. *IEEE transactions on visualization and computer graphics*, aug 2010.

[48] J. Huang, Y. Tong, H. Wei, and H. Bao. Boundary aligned smooth 3d cross-frame field. *ACM Trans. Graph.*, 30:143:1–143:8, dec 2011.

[49] J. Huang, T. Jiang, Y. Wang, Y. Tong, and H. Bao. Automatic frame field guided hexahedral mesh generation. Technical report, Zhejiang University, 2012.

[50] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas. Non-rigid registration under isometric deformations. *Comput. Graph. Forum*, 27(5):1449–1457, 2008.

[51] T. A. Ivey and J. M. Landsberg. *Cartan for Beginners: Differential Geometry Via Moving Frames and Exterior Differential Systems*. American Mathematical Society, illustrate edition, 2003.

[52] V. Jain, H. Zhang, and O. van Kaick. Non-rigid spectral correspondence of triangle meshes. *International Journal of Shape Modeling*, 13(01):101–124, 2007.

[53] M. Jin, Y. Wang, S.-T. Yau, and X. Gu. Optimal global conformal surface parameterization. In *Visualization, 2004. IEEE*, pages 267–274. IEEE, 2004.

[54] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[55] F. Kälberer, M. Nieser, and K. Polthier. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384, sep 2007.

[56] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. *Computer*, pages 43–51, 2006.

[57] V. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. *ACM Transactions on Graphics (TOG)*, 30(4):79, 2011.

[58] V. Kim, W. Li, N. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Transactions on Graphics (TOG)*, 31(4):54, 2012.

[59] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[60] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH 1998*, pages 105–114, New York, NY, USA, 1998. ACM.

[61] L. Kobbelt, J. Vorsatz, and H.-P. Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Comput. Geom. Theory Appl.*, 14(1-3):5–24, 1999.

[62] A. Kovnatsky, M. Bronstein, A. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. *arXiv preprint arXiv:1210.0026*, 2012.

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

[64] B. Lévy and Y. Liu. Lp centroidal Voronoi tessellation and its applications. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):119:1–119:11, July 2010.

[65] B. Levy and R. H. Zhang. Spectral geometry processing. In *ACM SIGGRAPH Course Notes*, 2010.

[66] Y. Li, Y. Liu, W. Xu, W. Wang, and B. Guo. All-hex meshing using singularity-restricted field. *ACM Trans. Graph. (to appear)*, dec 2012.

[67] P. Lienhardt. Topological models for boundary representation : a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991.

[68] Y. Lipman and T. Funkhouser. Mobius voting for surface correspondence. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), aug 2009.

[69] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential Coordinates for Interactive Mesh Editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[70] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph.*, 24:479–487, July 2005.

[71] Y. Liu and J. Snoeyink. A Comparison of Five Implementations of 3D Delaunay Tessellation. *Combinatorial and Computational Geometry*, 52:435–453, 2005.

[72] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation-energy smoothness and fast computation. *ACM Trans. Graph.*, 28: 101:1–101:17, September 2009.

[73] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987.

[74] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[75] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds, 2002.

[76] S. Michael. kd tree nearest neighbor and range search, 2005.

[77] P. Muigg, M. Hadwiger, H. Doleisch, and E. Gröller. Interactive volume visualization of general polyhedral grids. *IEEE transactions on visualization and computer graphics*, 17 (12):2115–24, dec 2011.

[78] P. Mullen, P. Memari, F. de Goes, and M. Desbrun. Hot: Hodge-optimized triangulations. *ACM Trans. Graph.*, 30:103:1–103:12, aug 2011.

[79] P. Murdoch. The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes. *Finite Elements in Analysis and Design*, 28(2):137–149, 1997.

[80] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas. An optimization approach to improving collections of shape maps. In *Computer Graphics Forum*, volume 30, pages 1481–1491. Wiley Online Library, 2011.

[81] M. Nieser, U. Reitebuch, and K. Polthier. CubeCover - parameterization of 3d volumes. *Computer Graphics Forum (SGP)*, 30:5:1397–1406, 2011.

[82] J. R. Nieto and A. Susín. Cage based deformations: A survey. In *Deformation Models*, pages 75–99. Springer, 2013.

[83] OpenVolumeMesh. OpenVolumeMesh - A Generic and Versatile Index-Based Data Structure for Polytopal Meshes. `http://www.openvolumemesh.org/`, 2012.

[84] R. Osserman. *A Survey of Minimal Surfaces*. Phoenix Edition Series. Dover Publications, 2002.

[85] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. J. Guibas. One point isometric matching with the heat kernel. *Comput. Graph. Forum*, 29(5):1555–1564, jul 2010.

[86] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, jul 2012.

[87] M. Ovsjanikov, M. Ben-Chen, F. Chazal, and L. Guibas. Analysis and visualization of maps between shapes. *Computer Graphics Forum*, 2013 (to appear).

[88] J. Palacios and E. Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph. (SIGGRAPH)*, 26:3:55, 2007.

[89] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[90] C.-H. Peng, E. Zhang, Y. Kobayashi, and P. Wonka. Connectivity editing for quadrilateral meshes. *ACM Trans. Graph.*, 30:141:1–141:12, dec 2011.

[91] U. Pinkall and K. Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[92] S. Prat, P. Gioia, and Y. Bertrand. Connectivity compression in an arbitrary dimension. *The Visual Computer*, 21(8-10):876–885, sep 2005.

[93] R. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 225–233. Eurographics Association, 2007.

[94] R. Rustamov, M. Ovsjanikov, O. Azencot, M. Ben-Chen, F. Chazal, and L. Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph. (SIGGRAPH)*, 2013.

[95] S. P. Serna, A. Stork, and D. W. Fellner. Considerations toward a Dynamic Mesh Data Structure. In *SIGRAD Conference*, pages 83–90, 2011.

[96] J. Shepherd. *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, University of Utah, 2007.

[97] J. F. Shepherd and C. J. Tuttle. Quality improvement and feature capture in hexahedral meshes. Technical Report UUSCI-2006-029, The University of Utah, 2006.

[98] J. R. Shewchuk. General-dimensional constrained delaunay and constrained regular triangulations, i: Combinatorial properties. *Discrete Comput. Geom.*, 39(1):580–637, mar 2008.

[99] B.-Y. Shih and H. Sakurai. Automated hexahedral mesh generation by swept volume decomposition and recomposition. In *5th International Meshing Roundtable*, pages 273–280, 1996.

[100] D. Sieger and M. Botsch. Design , Implementation , and Evaluation of the Surface mesh Data Structure. In *International Meshing Roundtable*, pages 533–550, 2011.

[101] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian Surface Editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. Eurographics Association, 2004.

[102] M. L. Staten, R. A. Kerr, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: Progress update. In *In Proceedings, 15th International Meshing Roundtable*, pages 469–486, 2006.

[103] Y. Su, K. Lee, and A. S. Kumar. Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method. *Computer-Aided Design*, 36(3):203 – 215, 2004.

[104] R. Sumner and J. Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (SIGGRAPH)*, volume 23, pages 399–405. ACM, 2004.

[105] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 351–358, New York, NY, USA, 1995. ACM.

[106] T. Tautges, T. Blacker, and S. Mitchell. The Whisker Weaving Algorithm: a Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes. *International Journal for Numerical Methods in Engineering*, 39(19):3327–3349, 1996.

[107] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 201–210, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[108] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph.*, 28: 75:1–75:9, July 2009.

[109] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[110] Y. Wang, B. Liu, and Y. Tong. Linear surface reconstruction from discrete fundamental forms on triangle meshes. *Computer Graphics Forum*, pages no–no, 2012.

[111] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, and E. Grinspun. Discrete quadratic curvature energies. *Comput. Aided Geom. Des.*, 24:499–518, November 2007.

[112] S. Yamakawa and K. Shimada. Hex-dominant mesh generation with directionality control via packing rectangular solid cells. In *Proceedings of Geometric Modeling and Processing 2002*, pages 2099–2129, 2003.

[113] H. Yamauchi, S. Gumhold, R. Zayer, and H.-P. Seidel. Mesh segmentation driven by gaussian curvature. *The Visual Computer*, 21:659–668, 2005. 10.1007/s00371-005-0319-x.

[114] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh Editing with Poisson-Based Gradient Field Manipulation. *ACM TRANS. GRAPH (SIGGRAPH)*, 23:644–651, 2004.

[115] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. In *Eurographics*, volume 24:3 of *Computer Graphics Forum*, pages 601–609, 2005.

[116] W. Zeng, Y. Zeng, Y. Wang, X. Yin, X. Gu, and D. Samaras. 3D non-rigid surface matching and registration based on holomorphic differentials. In *Computer Vision–ECCV 2008*, pages 1–14. Springer, 2008.

[117] G.-X. Zhang, S.-P. Du, Y.-K. Lai, T. Ni, and S.-M. Hu. Sketch guided solid texturing. *Graphical Models*, 73(3):59–73, may 2011.

[118] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. Van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. *Computer Graphics Forum*, 27(5):1431–1439, 2008.

[119] M. Zhang, J. Huang, X. Liu, and H. Bao. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph.*, 29(4):118:1—-118:8, 2010.

[120] X. Zhao, B. Li, L. Wang, and A. Kaufman. Texture-guided volumetric deformation and visualization using 3d moving least squares. *Vis. Comput.*, 28(2):193–204, feb 2012.

[121] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph. (SIGGRAPH)*, 24 (3):496–503, 2005.

[122] G. M. Ziegler. Polyhedral surfaces of high genus. In *Discrete Differential Geometry*, volume 38 of *Oberwolfach Seminars*, pages 191–213. Birkhäuser Basel, 2008.